

```

*****
126040 Wed Feb 10 17:30:49 2016
new/usr/src/uts/common/crypto/io/dca.c
6640 dca gets the instance number a lot, never actually uses it
*****
_____unchanged_portion_omitted_____

416 /* Convenience macros */
417 #define DCA_SOFTC_FROM_CTX(ctx) ((dca_t *) (ctx)->cc_provider)
418 /* Retrieve the softc and instance number from a SPI crypto context */
419 #define DCA_SOFTC_FROM_CTX(ctx, softc, instance) {
420     (softc) = (dca_t *) (ctx)->cc_provider;
421     (instance) = ddi_get_instance((softc)->dca_dip);
422 }

418 #define DCA_MECH_FROM_CTX(ctx) \
419     (((dca_request_t *) (ctx)->cc_provider_private)->dr_ctx.ctx_cm_type)

421 static int dca_bindchains_one(dca_request_t *reqp, size_t cnt, int dr_offset,
422     caddr_t kaddr, ddi_dma_handle_t handle, uint_t flags,
423     dca_chain_t *head, int *n_chain);
424 static uint64_t dca_ena(uint64_t ena);
425 static caddr_t dca_bufdaddr_out(crypto_data_t *data);
426 static char *dca_fma_eclass_string(char *model, dca_fma_eclass_t index);
427 static int dca_check_acc_handle(dca_t *dca, ddi_acc_handle_t handle,
428     dca_fma_eclass_t eclass_index);

430 static void dca_fma_init(dca_t *dca);
431 static void dca_fma_fini(dca_t *dca);
432 static int dca_fm_error_cb(dev_info_t *dip, ddi_fm_error_t *err,
433     const void *impl_data);

436 static dca_device_t dca_devices[] = {
437     /* Broadcom vanilla variants */
438     { 0x14e4, 0x5820, "Broadcom 5820" },
439     { 0x14e4, 0x5821, "Broadcom 5821" },
440     { 0x14e4, 0x5822, "Broadcom 5822" },
441     { 0x14e4, 0x5825, "Broadcom 5825" },
442     /* Sun specific OEMd variants */
443     { 0x108e, 0x5454, "SCA" },
444     { 0x108e, 0x5455, "SCA 1000" },
445     { 0x108e, 0x5457, "SCA 500" },
446     /* subsysid should be 0x5457, but got 0x1 from HW. Assume both here. */
447     { 0x108e, 0x1, "SCA 500" },
448 };
_____unchanged_portion_omitted_____

3680 /*
3681  * Cipher (encrypt/decrypt) entry points.
3682  */

3684 /* ARGSUSED */
3685 static int
3686 dca_encrypt_init(crypto_ctx_t *ctx, crypto_mechanism_t *mechanism,
3687     crypto_key_t *key, crypto_spi_ctx_template_t ctx_template,
3688     crypto_req_handle_t req)
3689 {
3690     int error = CRYPTO_FAILED;
3691     dca_t *softc;
3692     /* LINTED E_FUNC_SET_NOT_USED */
3693     int instance;

3693     softc = DCA_SOFTC_FROM_CTX(ctx);
3700     /* extract softc and instance number from context */
3701     DCA_SOFTC_FROM_CTX(ctx, softc, instance);

```

```

3694     DBG(softc, DENTRY, "dca_encrypt_init: started");

3696     /* check mechanism */
3697     switch (mechanism->cm_type) {
3698     case DES_CBC_MECH_INFO_TYPE:
3699         error = dca_3desctxinit(ctx, mechanism, key, KM_SLEEP,
3700             DR_ENCRYPT);
3701         break;
3702     case DES3_CBC_MECH_INFO_TYPE:
3703         error = dca_3desctxinit(ctx, mechanism, key, KM_SLEEP,
3704             DR_ENCRYPT | DR_TRIPLE);
3705         break;
3706     case RSA_PKCS_MECH_INFO_TYPE:
3707     case RSA_X_509_MECH_INFO_TYPE:
3708         error = dca_rsainit(ctx, mechanism, key, KM_SLEEP);
3709         break;
3710     default:
3711         cmn_err(CE_WARN, "dca_encrypt_init: unexpected mech type "
3712             "0x%llx\n", (unsigned long long)mechanism->cm_type);
3713         error = CRYPTO_MECHANISM_INVALID;
3714     }

3716     DBG(softc, DENTRY, "dca_encrypt_init: done, err = 0x%x", error);

3718     if (error == CRYPTO_SUCCESS)
3719         dca_enlist2(&softc->dca_ctx_list, ctx->cc_provider_private,
3720             &softc->dca_ctx_list_lock);

3722     return (error);
3723 }

3725 /* ARGSUSED */
3726 static int
3727 dca_encrypt(crypto_ctx_t *ctx, crypto_data_t *plaintext,
3728     crypto_data_t *ciphertext, crypto_req_handle_t req)
3729 {
3730     int error = CRYPTO_FAILED;
3731     dca_t *softc;
3732     /* LINTED E_FUNC_SET_NOT_USED */
3733     int instance;

3733     if (!ctx || !ctx->cc_provider || !ctx->cc_provider_private)
3734         return (CRYPTO_OPERATION_NOT_INITIALIZED);

3736     softc = DCA_SOFTC_FROM_CTX(ctx);
3737     /* extract softc and instance number from context */
3738     DCA_SOFTC_FROM_CTX(ctx, softc, instance);
3739     DBG(softc, DENTRY, "dca_encrypt: started");

3739     /* handle inplace ops */
3740     if (!ciphertext) {
3741         dca_request_t *reqp = ctx->cc_provider_private;
3742         reqp->dr_flags |= DR_INPLACE;
3743         ciphertext = plaintext;
3744     }

3746     /* check mechanism */
3747     switch (DCA_MECH_FROM_CTX(ctx)) {
3748     case DES_CBC_MECH_INFO_TYPE:
3749         error = dca_3des(ctx, plaintext, ciphertext, req, DR_ENCRYPT);
3750         break;
3751     case DES3_CBC_MECH_INFO_TYPE:
3752         error = dca_3des(ctx, plaintext, ciphertext, req,
3753             DR_ENCRYPT | DR_TRIPLE);
3754         break;
3755     case RSA_PKCS_MECH_INFO_TYPE:

```

```

3756     case RSA_X_509_MECH_INFO_TYPE:
3757         error = dca_rsastart(ctx, plaintext, ciphertext, req,
3758             DCA_RSA_ENC);
3759         break;
3760     default:
3761         /* Should never reach here */
3762         cmn_err(CE_WARN, "dca_encrypt: unexpected mech type "
3763             "0x%llx\n", (unsigned long long)DCA_MECH_FROM_CTX(ctx));
3764         error = CRYPTO_MECHANISM_INVALID;
3765     }

3767     if ((error != CRYPTO_QUEUED) && (error != CRYPTO_SUCCESS) &&
3768         (error != CRYPTO_BUFFER_TOO_SMALL)) {
3769         ciphertext->cd_length = 0;
3770     }

3772     DBG(softec, DENTRY, "dca_encrypt: done, err = 0x%x", error);

3774     return (error);
3775 }

3777 /* ARGSUSED */
3778 static int
3779 dca_encrypt_update(crypto_ctx_t *ctx, crypto_data_t *plaintext,
3780     crypto_data_t *ciphertext, crypto_req_handle_t req)
3781 {
3782     int error = CRYPTO_FAILED;
3783     dca_t *softec;
3784     /* LINTED E_FUNC_SET_NOT_USED */
3785     int instance;

3786     if (!ctx || !ctx->cc_provider || !ctx->cc_provider_private)
3787         return (CRYPTO_OPERATION_NOT_INITIALIZED);

3788     softec = DCA_SOFTC_FROM_CTX(ctx);
3789     /* extract softec and instance number from context */
3790     DCA_SOFTC_FROM_CTX(ctx, softec, instance);
3791     DBG(softec, DENTRY, "dca_encrypt_update: started");

3792     /* handle inplace ops */
3793     if (!ciphertext) {
3794         dca_request_t *reqp = ctx->cc_provider_private;
3795         reqp->dr_flags |= DR_INPLACE;
3796         ciphertext = plaintext;
3797     }

3798     /* check mechanism */
3799     switch (DCA_MECH_FROM_CTX(ctx)) {
3800     case DES_CBC_MECH_INFO_TYPE:
3801         error = dca_3desupdate(ctx, plaintext, ciphertext, req,
3802             DR_ENCRYPT);
3803         break;
3804     case DES3_CBC_MECH_INFO_TYPE:
3805         error = dca_3desupdate(ctx, plaintext, ciphertext, req,
3806             DR_ENCRYPT | DR_TRIPLE);
3807         break;
3808     default:
3809         /* Should never reach here */
3810         cmn_err(CE_WARN, "dca_encrypt_update: unexpected mech type "
3811             "0x%llx\n", (unsigned long long)DCA_MECH_FROM_CTX(ctx));
3812         error = CRYPTO_MECHANISM_INVALID;
3813     }

3815     DBG(softec, DENTRY, "dca_encrypt_update: done, err = 0x%x", error);

3817     return (error);

```

```

3818 }

3820 /* ARGSUSED */
3821 static int
3822 dca_encrypt_final(crypto_ctx_t *ctx, crypto_data_t *ciphertext,
3823     crypto_req_handle_t req)
3824 {
3825     int error = CRYPTO_FAILED;
3826     dca_t *softec;
3827     /* LINTED E_FUNC_SET_NOT_USED */
3828     int instance;

3829     if (!ctx || !ctx->cc_provider || !ctx->cc_provider_private)
3830         return (CRYPTO_OPERATION_NOT_INITIALIZED);

3831     softec = DCA_SOFTC_FROM_CTX(ctx);
3832     /* extract softec and instance number from context */
3833     DCA_SOFTC_FROM_CTX(ctx, softec, instance);
3834     DBG(softec, DENTRY, "dca_encrypt_final: started");

3835     /* check mechanism */
3836     switch (DCA_MECH_FROM_CTX(ctx)) {
3837     case DES_CBC_MECH_INFO_TYPE:
3838         error = dca_3desfinal(ctx, ciphertext, DR_ENCRYPT);
3839         break;
3840     case DES3_CBC_MECH_INFO_TYPE:
3841         error = dca_3desfinal(ctx, ciphertext, DR_ENCRYPT | DR_TRIPLE);
3842         break;
3843     default:
3844         /* Should never reach here */
3845         cmn_err(CE_WARN, "dca_encrypt_final: unexpected mech type "
3846             "0x%llx\n", (unsigned long long)DCA_MECH_FROM_CTX(ctx));
3847         error = CRYPTO_MECHANISM_INVALID;
3848     }

3849     DBG(softec, DENTRY, "dca_encrypt_final: done, err = 0x%x", error);

3851     return (error);
3852 }

_____unchanged_portion_omitted_____

3906 /* ARGSUSED */
3907 static int
3908 dca_decrypt_init(crypto_ctx_t *ctx, crypto_mechanism_t *mechanism,
3909     crypto_key_t *key, crypto_spi_ctx_template_t ctx_template,
3910     crypto_req_handle_t req)
3911 {
3912     int error = CRYPTO_FAILED;
3913     dca_t *softec;
3914     /* LINTED E_FUNC_SET_NOT_USED */
3915     int instance;

3916     softec = DCA_SOFTC_FROM_CTX(ctx);
3917     /* extract softec and instance number from context */
3918     DCA_SOFTC_FROM_CTX(ctx, softec, instance);
3919     DBG(softec, DENTRY, "dca_decrypt_init: started");

3920     /* check mechanism */
3921     switch (mechanism->cm_type) {
3922     case DES_CBC_MECH_INFO_TYPE:
3923         error = dca_3desctxinit(ctx, mechanism, key, KM_SLEEP,
3924             DR_DECRYPT);
3925         break;
3926     case DES3_CBC_MECH_INFO_TYPE:
3927         error = dca_3desctxinit(ctx, mechanism, key, KM_SLEEP,
3928             DR_DECRYPT | DR_TRIPLE);

```

```

3927         break;
3928     case RSA_PKCS_MECH_INFO_TYPE:
3929     case RSA_X_509_MECH_INFO_TYPE:
3930         error = dca_rsainit(ctx, mechanism, key, KM_SLEEP);
3931         break;
3932     default:
3933         cmn_err(CE_WARN, "dca_decrypt_init: unexpected mech type "
3934             "0x%llx\n", (unsigned long long)mechanism->cm_type);
3935         error = CRYPTO_MECHANISM_INVALID;
3936     }
3937
3938     DBG(softc, DENTRY, "dca_decrypt_init: done, err = 0x%x", error);
3939
3940     if (error == CRYPTO_SUCCESS)
3941         dca_enlist2(&softc->dca_ctx_list, ctx->cc_provider_private,
3942             &softc->dca_ctx_list_lock);
3943
3944     return (error);
3945 }
3946
3947 /* ARGSUSED */
3948 static int
3949 dca_decrypt(crypto_ctx_t *ctx, crypto_data_t *ciphertext,
3950     crypto_data_t *plaintext, crypto_req_handle_t req)
3951 {
3952     int error = CRYPTO_FAILED;
3953     dca_t *softc;
3954     /* LINTED E_FUNC_SET_NOT_USED */
3955     int instance;
3956
3957     if (!ctx || !ctx->cc_provider || !ctx->cc_provider_private)
3958         return (CRYPTO_OPERATION_NOT_INITIALIZED);
3959
3960     softc = DCA_SOFTC_FROM_CTX(ctx);
3961     /* extract softc and instance number from context */
3962     DCA_SOFTC_FROM_CTX(ctx, softc, instance);
3963     DBG(softc, DENTRY, "dca_decrypt: started");
3964
3965     /* handle inplace ops */
3966     if (!plaintext) {
3967         dca_request_t *reqp = ctx->cc_provider_private;
3968         reqp->dr_flags |= DR_INPLACE;
3969         plaintext = ciphertext;
3970     }
3971
3972     /* check mechanism */
3973     switch (DCA_MECH_FROM_CTX(ctx)) {
3974     case DES_CBC_MECH_INFO_TYPE:
3975         error = dca_3des(ctx, ciphertext, plaintext, req, DR_DECRYPT);
3976         break;
3977     case DES3_CBC_MECH_INFO_TYPE:
3978         error = dca_3des(ctx, ciphertext, plaintext, req,
3979             DR_DECRYPT | DR_TRIPLE);
3980         break;
3981     case RSA_PKCS_MECH_INFO_TYPE:
3982     case RSA_X_509_MECH_INFO_TYPE:
3983         error = dca_rsastart(ctx, ciphertext, plaintext, req,
3984             DCA_RSA_DEC);
3985         break;
3986     default:
3987         /* Should never reach here */
3988         cmn_err(CE_WARN, "dca_decrypt: unexpected mech type "
3989             "0x%llx\n", (unsigned long long)DCA_MECH_FROM_CTX(ctx));
3990         error = CRYPTO_MECHANISM_INVALID;
3991     }

```

```

3989     if ((error != CRYPTO_QUEUED) && (error != CRYPTO_SUCCESS) &&
3990         (error != CRYPTO_BUFFER_TOO_SMALL)) {
3991         if (plaintext)
3992             plaintext->cd_length = 0;
3993     }
3994
3995     DBG(softc, DENTRY, "dca_decrypt: done, err = 0x%x", error);
3996
3997     return (error);
3998 }
3999
4000 /* ARGSUSED */
4001 static int
4002 dca_decrypt_update(crypto_ctx_t *ctx, crypto_data_t *ciphertext,
4003     crypto_data_t *plaintext, crypto_req_handle_t req)
4004 {
4005     int error = CRYPTO_FAILED;
4006     dca_t *softc;
4007     /* LINTED E_FUNC_SET_NOT_USED */
4008     int instance;
4009
4010     if (!ctx || !ctx->cc_provider || !ctx->cc_provider_private)
4011         return (CRYPTO_OPERATION_NOT_INITIALIZED);
4012
4013     softc = DCA_SOFTC_FROM_CTX(ctx);
4014     /* extract softc and instance number from context */
4015     DCA_SOFTC_FROM_CTX(ctx, softc, instance);
4016     DBG(softc, DENTRY, "dca_decrypt_update: started");
4017
4018     /* handle inplace ops */
4019     if (!plaintext) {
4020         dca_request_t *reqp = ctx->cc_provider_private;
4021         reqp->dr_flags |= DR_INPLACE;
4022         plaintext = ciphertext;
4023     }
4024
4025     /* check mechanism */
4026     switch (DCA_MECH_FROM_CTX(ctx)) {
4027     case DES_CBC_MECH_INFO_TYPE:
4028         error = dca_3desupdate(ctx, ciphertext, plaintext, req,
4029             DR_DECRYPT);
4030         break;
4031     case DES3_CBC_MECH_INFO_TYPE:
4032         error = dca_3desupdate(ctx, ciphertext, plaintext, req,
4033             DR_DECRYPT | DR_TRIPLE);
4034         break;
4035     default:
4036         /* Should never reach here */
4037         cmn_err(CE_WARN, "dca_decrypt_update: unexpected mech type "
4038             "0x%llx\n", (unsigned long long)DCA_MECH_FROM_CTX(ctx));
4039         error = CRYPTO_MECHANISM_INVALID;
4040     }
4041
4042     DBG(softc, DENTRY, "dca_decrypt_update: done, err = 0x%x", error);
4043
4044     return (error);
4045 }
4046
4047 /* ARGSUSED */
4048 static int
4049 dca_decrypt_final(crypto_ctx_t *ctx, crypto_data_t *plaintext,
4050     crypto_req_handle_t req)
4051 {
4052     int error = CRYPTO_FAILED;
4053     dca_t *softc;
4054     /* LINTED E_FUNC_SET_NOT_USED */

```

```

4077     int instance;

4051     if (!ctx || !ctx->cc_provider || !ctx->cc_provider_private)
4052         return (CRYPTO_OPERATION_NOT_INITIALIZED);

4054     softc = DCA_SOFTC_FROM_CTX(ctx);
4082     /* extract softc and instance number from context */
4083     DCA_SOFTC_FROM_CTX(ctx, softc, instance);
4055     DBG(softc, DENTRY, "dca_decrypt_final: started");

4057     /* check mechanism */
4058     switch (DCA_MECH_FROM_CTX(ctx)) {
4059     case DES_CBC_MECH_INFO_TYPE:
4060         error = dca_3desfinal(ctx, plaintext, DR_DECRYPT);
4061         break;
4062     case DES3_CBC_MECH_INFO_TYPE:
4063         error = dca_3desfinal(ctx, plaintext, DR_DECRYPT | DR_TRIPLE);
4064         break;
4065     default:
4066         /* Should never reach here */
4067         cmn_err(CE_WARN, "dca_decrypt_final: unexpected mech type "
4068             "0x%llx\n", (unsigned long)DCA_MECH_FROM_CTX(ctx));
4069         error = CRYPTO_MECHANISM_INVALID;
4070     }

4072     DBG(softc, DENTRY, "dca_decrypt_final: done, err = 0x%x", error);

4074     return (error);
4075 }

unchanged_portion_omitted

4129 /*
4130  * Sign entry points.
4131  */

4133 /* ARGSUSED */
4134 static int
4135 dca_sign_init(crypto_ctx_t *ctx, crypto_mechanism_t *mechanism,
4136     crypto_key_t *key, crypto_spi_ctx_template_t ctx_template,
4137     crypto_req_handle_t req)
4138 {
4139     int error = CRYPTO_FAILED;
4140     dca_t *softc;
4141     /* LINTED E_FUNC_SET_NOT_USED */
4142     int instance;

4142     softc = DCA_SOFTC_FROM_CTX(ctx);
4173     /* extract softc and instance number from context */
4174     DCA_SOFTC_FROM_CTX(ctx, softc, instance);
4143     DBG(softc, DENTRY, "dca_sign_init: started\n");

4145     if (ctx_template != NULL)
4146         return (CRYPTO_ARGUMENTS_BAD);

4148     /* check mechanism */
4149     switch (mechanism->cm_type) {
4150     case RSA_PKCS_MECH_INFO_TYPE:
4151     case RSA_X_509_MECH_INFO_TYPE:
4152         error = dca_rsainit(ctx, mechanism, key, KM_SLEEP);
4153         break;
4154     case DSA_MECH_INFO_TYPE:
4155         error = dca_dsainit(ctx, mechanism, key, KM_SLEEP,
4156             DCA_DSA_SIGN);
4157         break;
4158     default:
4159         cmn_err(CE_WARN, "dca_sign_init: unexpected mech type "

```

```

4160         "0x%llx\n", (unsigned long long)mechanism->cm_type);
4161         error = CRYPTO_MECHANISM_INVALID;
4162     }

4164     DBG(softc, DENTRY, "dca_sign_init: done, err = 0x%x", error);

4166     if (error == CRYPTO_SUCCESS)
4167         dca_enlist2(&softc->dca_ctx_list, ctx->cc_provider_private,
4168             &softc->dca_ctx_list_lock);

4170     return (error);
4171 }

4173 static int
4174 dca_sign(crypto_ctx_t *ctx, crypto_data_t *data,
4175     crypto_data_t *signature, crypto_req_handle_t req)
4176 {
4177     int error = CRYPTO_FAILED;
4178     dca_t *softc;
4179     /* LINTED E_FUNC_SET_NOT_USED */
4180     int instance;

4180     if (!ctx || !ctx->cc_provider || !ctx->cc_provider_private)
4181         return (CRYPTO_OPERATION_NOT_INITIALIZED);

4183     softc = DCA_SOFTC_FROM_CTX(ctx);
4217     /* extract softc and instance number from context */
4218     DCA_SOFTC_FROM_CTX(ctx, softc, instance);
4184     DBG(softc, DENTRY, "dca_sign: started\n");

4186     /* check mechanism */
4187     switch (DCA_MECH_FROM_CTX(ctx)) {
4188     case RSA_PKCS_MECH_INFO_TYPE:
4189     case RSA_X_509_MECH_INFO_TYPE:
4190         error = dca_rsastart(ctx, data, signature, req, DCA_RSA_SIGN);
4191         break;
4192     case DSA_MECH_INFO_TYPE:
4193         error = dca_dsa_sign(ctx, data, signature, req);
4194         break;
4195     default:
4196         cmn_err(CE_WARN, "dca_sign: unexpected mech type "
4197             "0x%llx\n", (unsigned long long)DCA_MECH_FROM_CTX(ctx));
4198         error = CRYPTO_MECHANISM_INVALID;
4199     }

4201     DBG(softc, DENTRY, "dca_sign: done, err = 0x%x", error);

4203     return (error);
4204 }

4206 /* ARGSUSED */
4207 static int
4208 dca_sign_update(crypto_ctx_t *ctx, crypto_data_t *data,
4209     crypto_req_handle_t req)
4210 {
4211     int error = CRYPTO_MECHANISM_INVALID;
4212     dca_t *softc;
4213     /* LINTED E_FUNC_SET_NOT_USED */
4214     int instance;

4214     if (!ctx || !ctx->cc_provider || !ctx->cc_provider_private)
4215         return (CRYPTO_OPERATION_NOT_INITIALIZED);

4217     softc = DCA_SOFTC_FROM_CTX(ctx);
4254     /* extract softc and instance number from context */
4255     DCA_SOFTC_FROM_CTX(ctx, softc, instance);

```

```

4218     DBG(softc, DENTRY, "dca_sign_update: started\n");
4220     cmn_err(CE_WARN, "dca_sign_update: unexpected mech type "
4221             "0x%llx\n", (unsigned long long)DCA_MECH_FROM_CTX(ctx));
4223     DBG(softc, DENTRY, "dca_sign_update: done, err = 0x%x", error);
4225     return (error);
4226 }

4228 /* ARGSUSED */
4229 static int
4230 dca_sign_final(crypto_ctx_t *ctx, crypto_data_t *signature,
4231               crypto_req_handle_t req)
4232 {
4233     int error = CRYPTO_MECHANISM_INVALID;
4234     dca_t *softc;
4235     /* LINTED E_FUNC_SET_NOT_USED */
4236     int instance;

4237     if (!ctx || !ctx->cc_provider || !ctx->cc_provider_private)
4238         return (CRYPTO_OPERATION_NOT_INITIALIZED);

4239     softc = DCA_SOFTC_FROM_CTX(ctx);
4240     /* extract softc and instance number from context */
4241     DCA_SOFTC_FROM_CTX(ctx, softc, instance);
4242     DBG(softc, DENTRY, "dca_sign_final: started\n");

4243     cmn_err(CE_WARN, "dca_sign_final: unexpected mech type "
4244             "0x%llx\n", (unsigned long long)DCA_MECH_FROM_CTX(ctx));

4245     DBG(softc, DENTRY, "dca_sign_final: done, err = 0x%x", error);

4247     return (error);
4248 }
unchanged portion omitted

4286 /* ARGSUSED */
4287 static int
4288 dca_sign_recover_init(crypto_ctx_t *ctx, crypto_mechanism_t *mechanism,
4289                      crypto_key_t *key, crypto_spi_ctx_template_t ctx_template,
4290                      crypto_req_handle_t req)
4291 {
4292     int error = CRYPTO_FAILED;
4293     dca_t *softc;
4294     /* LINTED E_FUNC_SET_NOT_USED */
4295     int instance;

4296     softc = DCA_SOFTC_FROM_CTX(ctx);
4297     /* extract softc and instance number from context */
4298     DCA_SOFTC_FROM_CTX(ctx, softc, instance);
4299     DBG(softc, DENTRY, "dca_sign_recover_init: started\n");

4300     if (ctx_template != NULL)
4301         return (CRYPTO_ARGUMENTS_BAD);

4302     /* check mechanism */
4303     switch (mechanism->cm_type) {
4304     case RSA_PKCS_MECH_INFO_TYPE:
4305     case RSA_X_509_MECH_INFO_TYPE:
4306         error = dca_rsainit(ctx, mechanism, key, KM_SLEEP);
4307         break;
4308     default:
4309         cmn_err(CE_WARN, "dca_sign_recover_init: unexpected mech type "
4310                 "0x%llx\n", (unsigned long long)mechanism->cm_type);
4311         error = CRYPTO_MECHANISM_INVALID;

```

```

4311     }
4313     DBG(softc, DENTRY, "dca_sign_recover_init: done, err = 0x%x", error);

4315     if (error == CRYPTO_SUCCESS)
4316         dca_enlist2(&softc->dca_ctx_list, ctx->cc_provider_private,
4317                   &softc->dca_ctx_list_lock);

4319     return (error);
4320 }

4322 static int
4323 dca_sign_recover(crypto_ctx_t *ctx, crypto_data_t *data,
4324                 crypto_data_t *signature, crypto_req_handle_t req)
4325 {
4326     int error = CRYPTO_FAILED;
4327     dca_t *softc;
4328     /* LINTED E_FUNC_SET_NOT_USED */
4329     int instance;

4330     if (!ctx || !ctx->cc_provider || !ctx->cc_provider_private)
4331         return (CRYPTO_OPERATION_NOT_INITIALIZED);

4332     softc = DCA_SOFTC_FROM_CTX(ctx);
4333     /* extract softc and instance number from context */
4334     DCA_SOFTC_FROM_CTX(ctx, softc, instance);
4335     DBG(softc, DENTRY, "dca_sign_recover: started\n");

4336     /* check mechanism */
4337     switch (DCA_MECH_FROM_CTX(ctx)) {
4338     case RSA_PKCS_MECH_INFO_TYPE:
4339     case RSA_X_509_MECH_INFO_TYPE:
4340         error = dca_rsastart(ctx, data, signature, req, DCA_RSA_SIGNR);
4341         break;
4342     default:
4343         cmn_err(CE_WARN, "dca_sign_recover: unexpected mech type "
4344                 "0x%llx\n", (unsigned long long)DCA_MECH_FROM_CTX(ctx));
4345         error = CRYPTO_MECHANISM_INVALID;
4346     }

4347     DBG(softc, DENTRY, "dca_sign_recover: done, err = 0x%x", error);

4349     return (error);
4350 }

4352 static int
4353 dca_sign_recover_atomic(crypto_provider_handle_t provider,
4354                        crypto_session_id_t session_id, crypto_mechanism_t *mechanism,
4355                        crypto_key_t *key, crypto_data_t *data, crypto_data_t *signature,
4356                        crypto_spi_ctx_template_t ctx_template, crypto_req_handle_t req)
4357 {
4358     int error = CRYPTO_FAILED;
4359     dca_t *softc = (dca_t *)provider;
4360     /* LINTED E_FUNC_SET_NOT_USED */
4361     int instance;

4362     instance = ddi_get_instance(softc->dca_dip);
4363     DBG(softc, DENTRY, "dca_sign_recover_atomic: started\n");

4364     if (ctx_template != NULL)
4365         return (CRYPTO_ARGUMENTS_BAD);

4366     /* check mechanism */
4367     switch (mechanism->cm_type) {
4368     case RSA_PKCS_MECH_INFO_TYPE:
4369     case RSA_X_509_MECH_INFO_TYPE:

```

```

4370         error = dca_rsaatomic(provider, session_id, mechanism, key,
4371                               data, signature, KM_SLEEP, req, DCA_RSA_SIGNR);
4372         break;
4373     default:
4374         cmn_err(CE_WARN, "dca_sign_recover_atomic: unexpected mech type"
4375              " 0x%llx\n", (unsigned long long)mechanism->cm_type);
4376         error = CRYPTO_MECHANISM_INVALID;
4377     }
4378
4379     DBG(sofct, DENTRY, "dca_sign_recover_atomic: done, err = 0x%x", error);
4380
4381     return (error);
4382 }
4383
4384 /*
4385  * Verify entry points.
4386  */
4387
4388 /* ARGSUSED */
4389 static int
4390 dca_verify_init(crypto_ctx_t *ctx, crypto_mechanism_t *mechanism,
4391               crypto_key_t *key, crypto_spi_ctx_template_t ctx_template,
4392               crypto_req_handle_t req)
4393 {
4394     int error = CRYPTO_FAILED;
4395     dca_t *sofct;
4396     /* LINTED E_FUNC_SET_NOT_USED */
4397     int instance;
4398
4399     sofct = DCA_SOFTC_FROM_CTX(ctx);
4400     /* extract sofct and instance number from context */
4401     DCA_SOFTC_FROM_CTX(ctx, sofct, instance);
4402     DBG(sofct, DENTRY, "dca_verify_init: started\n");
4403
4404     if (ctx_template != NULL)
4405         return (CRYPTO_ARGUMENTS_BAD);
4406
4407     /* check mechanism */
4408     switch (mechanism->cm_type) {
4409     case RSA_PKCS_MECH_INFO_TYPE:
4410     case RSA_X_509_MECH_INFO_TYPE:
4411         error = dca_rsainit(ctx, mechanism, key, KM_SLEEP);
4412         break;
4413     case DSA_MECH_INFO_TYPE:
4414         error = dca_dsainit(ctx, mechanism, key, KM_SLEEP,
4415                             DCA_DSA_VRFY);
4416         break;
4417     default:
4418         cmn_err(CE_WARN, "dca_verify_init: unexpected mech type "
4419              "0x%llx\n", (unsigned long long)mechanism->cm_type);
4420         error = CRYPTO_MECHANISM_INVALID;
4421     }
4422
4423     DBG(sofct, DENTRY, "dca_verify_init: done, err = 0x%x", error);
4424
4425     if (error == CRYPTO_SUCCESS)
4426         dca_enlist2(&sofct->dca_ctx_list, ctx->cc_provider_private,
4427                  &sofct->dca_ctx_list_lock);
4428
4429     return (error);
4430 }
4431
4432 static int
4433 dca_verify(crypto_ctx_t *ctx, crypto_data_t *data, crypto_data_t *signature,
4434            crypto_req_handle_t req)

```

```

4432     int error = CRYPTO_FAILED;
4433     dca_t *sofct;
4434     /* LINTED E_FUNC_SET_NOT_USED */
4435     int instance;
4436
4437     if (!ctx || !ctx->cc_provider || !ctx->cc_provider_private)
4438         return (CRYPTO_OPERATION_NOT_INITIALIZED);
4439
4440     sofct = DCA_SOFTC_FROM_CTX(ctx);
4441     /* extract sofct and instance number from context */
4442     DCA_SOFTC_FROM_CTX(ctx, sofct, instance);
4443     DBG(sofct, DENTRY, "dca_verify: started\n");
4444
4445     /* check mechanism */
4446     switch (DCA_MECH_FROM_CTX(ctx)) {
4447     case RSA_PKCS_MECH_INFO_TYPE:
4448     case RSA_X_509_MECH_INFO_TYPE:
4449         error = dca_rsastart(ctx, signature, data, req, DCA_RSA_VRFY);
4450         break;
4451     case DSA_MECH_INFO_TYPE:
4452         error = dca_dsa_verify(ctx, data, signature, req);
4453         break;
4454     default:
4455         cmn_err(CE_WARN, "dca_verify: unexpected mech type "
4456              "0x%llx\n", (unsigned long long)DCA_MECH_FROM_CTX(ctx));
4457         error = CRYPTO_MECHANISM_INVALID;
4458     }
4459
4460     DBG(sofct, DENTRY, "dca_verify: done, err = 0x%x", error);
4461
4462     return (error);
4463 }
4464
4465 /* ARGSUSED */
4466 static int
4467 dca_verify_update(crypto_ctx_t *ctx, crypto_data_t *data,
4468                  crypto_req_handle_t req)
4469 {
4470     int error = CRYPTO_MECHANISM_INVALID;
4471     dca_t *sofct;
4472     /* LINTED E_FUNC_SET_NOT_USED */
4473     int instance;
4474
4475     if (!ctx || !ctx->cc_provider || !ctx->cc_provider_private)
4476         return (CRYPTO_OPERATION_NOT_INITIALIZED);
4477
4478     sofct = DCA_SOFTC_FROM_CTX(ctx);
4479     /* extract sofct and instance number from context */
4480     DCA_SOFTC_FROM_CTX(ctx, sofct, instance);
4481     DBG(sofct, DENTRY, "dca_verify_update: started\n");
4482
4483     cmn_err(CE_WARN, "dca_verify_update: unexpected mech type "
4484          "0x%llx\n", (unsigned long long)DCA_MECH_FROM_CTX(ctx));
4485
4486     DBG(sofct, DENTRY, "dca_verify_update: done, err = 0x%x", error);
4487
4488     return (error);
4489 }
4490
4491 /* ARGSUSED */
4492 static int
4493 dca_verify_final(crypto_ctx_t *ctx, crypto_data_t *signature,
4494                 crypto_req_handle_t req)
4495 {
4496     int error = CRYPTO_MECHANISM_INVALID;
4497     dca_t *sofct;

```

```

4549  /* LINTED E_FUNC_SET_NOT_USED */
4550  int instance;

4491  if (!ctx || !ctx->cc_provider || !ctx->cc_provider_private)
4492      return (CRYPTO_OPERATION_NOT_INITIALIZED);

4494  softc = DCA_SOFTC_FROM_CTX(ctx);
4555  /* extract softc and instance number from context */
4556  DCA_SOFTC_FROM_CTX(ctx, softc, instance);
4495  DBG(softc, DENTRY, "dca_verify_final: started\n");

4497  cmn_err(CE_WARN, "dca_verify_final: unexpected mech type "
4498          "0x%llx\n", (unsigned long long)DCA_MECH_FROM_CTX(ctx));

4500  DBG(softc, DENTRY, "dca_verify_final: done, err = 0x%x", error);

4502  return (error);
4503 }
  unchanged_portion_omitted_

4541 /* ARGSUSED */
4542 static int
4543 dca_verify_recover_init(crypto_ctx_t *ctx, crypto_mechanism_t *mechanism,
4544                       crypto_key_t *key, crypto_spi_ctx_template_t ctx_template,
4545                       crypto_req_handle_t req)
4546 {
4547     int error = CRYPTO_MECHANISM_INVALID;
4548     dca_t *softc;
4611  /* LINTED E_FUNC_SET_NOT_USED */
4612  int instance;

4550  softc = DCA_SOFTC_FROM_CTX(ctx);
4614  /* extract softc and instance number from context */
4615  DCA_SOFTC_FROM_CTX(ctx, softc, instance);
4551  DBG(softc, DENTRY, "dca_verify_recover_init: started\n");

4553  if (ctx_template != NULL)
4554      return (CRYPTO_ARGUMENTS_BAD);

4556  /* check mechanism */
4557  switch (mechanism->cm_type) {
4558  case RSA_PKCS_MECH_INFO_TYPE:
4559  case RSA_X_509_MECH_INFO_TYPE:
4560      error = dca_rsainit(ctx, mechanism, key, KM_SLEEP);
4561      break;
4562  default:
4563      cmn_err(CE_WARN, "dca_verify_recover_init: unexpected mech type "
4564              "0x%llx\n", (unsigned long long)mechanism->cm_type);
4565  }

4567  DBG(softc, DENTRY, "dca_verify_recover_init: done, err = 0x%x", error);

4569  if (error == CRYPTO_SUCCESS)
4570      dca_enlist2(&softc->dca_ctx_list, ctx->cc_provider_private,
4571                &softc->dca_ctx_list_lock);

4573  return (error);
4574 }

4576 static int
4577 dca_verify_recover(crypto_ctx_t *ctx, crypto_data_t *signature,
4578                   crypto_data_t *data, crypto_req_handle_t req)
4579 {
4580     int error = CRYPTO_MECHANISM_INVALID;
4581     dca_t *softc;
4647  /* LINTED E_FUNC_SET_NOT_USED */

```

```

4648  int instance;

4583  if (!ctx || !ctx->cc_provider || !ctx->cc_provider_private)
4584      return (CRYPTO_OPERATION_NOT_INITIALIZED);

4586  softc = DCA_SOFTC_FROM_CTX(ctx);
4653  /* extract softc and instance number from context */
4654  DCA_SOFTC_FROM_CTX(ctx, softc, instance);
4587  DBG(softc, DENTRY, "dca_verify_recover: started\n");

4589  /* check mechanism */
4590  switch (DCA_MECH_FROM_CTX(ctx)) {
4591  case RSA_PKCS_MECH_INFO_TYPE:
4592  case RSA_X_509_MECH_INFO_TYPE:
4593      error = dca_rsastart(ctx, signature, data, req, DCA_RSA_VRFYR);
4594      break;
4595  default:
4596      cmn_err(CE_WARN, "dca_verify_recover: unexpected mech type "
4597              "0x%llx\n", (unsigned long long)DCA_MECH_FROM_CTX(ctx));
4598  }

4600  DBG(softc, DENTRY, "dca_verify_recover: done, err = 0x%x", error);

4602  return (error);
4603 }
  unchanged_portion_omitted_

4638 /*
4639  * Random number entry points.
4640  */

4642 /* ARGSUSED */
4643 static int
4644 dca_generate_random(crypto_provider_handle_t provider,
4645                   crypto_session_id_t session_id,
4646                   uchar_t *buf, size_t len, crypto_req_handle_t req)
4647 {
4648     int error = CRYPTO_FAILED;
4649     dca_t *softc = (dca_t *)provider;
4718  /* LINTED E_FUNC_SET_NOT_USED */
4719  int instance;

4721  instance = ddi_get_instance(softc->dca_dip);
4651  DBG(softc, DENTRY, "dca_generate_random: started");

4653  error = dca_rng(softc, buf, len, req);

4655  DBG(softc, DENTRY, "dca_generate_random: done, err = 0x%x", error);

4657  return (error);
4658 }

4660 /*
4661  * Context management entry points.
4662  */

4664 int
4665 dca_free_context(crypto_ctx_t *ctx)
4666 {
4667     int error = CRYPTO_SUCCESS;
4668     dca_t *softc;
4740  /* LINTED E_FUNC_SET_NOT_USED */
4741  int instance;

4670  softc = DCA_SOFTC_FROM_CTX(ctx);
4743  /* extract softc and instance number from context */

```

```
4744     DCA_SOFTC_FROM_CTX(ctx, softc, instance);
4671     DBG(softc, DENTRY, "dca_free_context: entered");

4673     if (ctx->cc_provider_private == NULL)
4674         return (error);

4676     dca_rmlist2(ctx->cc_provider_private, &softc->dca_ctx_list_lock);

4678     error = dca_free_context_low(ctx);

4680     DBG(softc, DENTRY, "dca_free_context: done, err = 0x%x", error);

4682     return (error);
4683 }
_____unchanged_portion_omitted_____
```


new/usr/src/uts/intel/dca/Makefile

1

```
*****
2401 Wed Feb 10 17:30:50 2016
new/usr/src/uts/intel/dca/Makefile
6640 dca gets the instance number a lot, never actually uses it
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 #
23 # Copyright 2006 Sun Microsystems, Inc. All rights reserved.
24 # Use is subject to license terms.
25 #
26 #
27 #
28 # This makefile drives the production of the DCA kCF provider.
29 #
30 # intel implementation architecture dependent
31 #
32 #
33 #
34 # Path to the base of the uts directory tree (usually /usr/src/uts).
35 #
36 UTBASE = ../../
37 #
38 #
39 # Define the module and object file sets.
40 #
41 MODULE = dca
42 OBJECTS = $(DCA_OBJS:%=$(OBJS_DIR)/%)
43 LINTS = $(DCA_OBJS:%.o=$(LINTS_DIR)/%.ln)
44 ROOTMODULE = $(ROOT_DRV_DIR)/$(MODULE)
45 CONF_SRC_DIR = $(UTSBASE)/common/crypto/io
46 #
47 #
48 # Include common rules.
49 #
50 include $(UTSBASE)/intel/Makefile.intel
51 #
52 # set signing mode
53 ELFSIGN_MOD = $(ELFSIGN_CRYPT)
54 #
55 #
56 # Define targets
57 #
58 ALL_TARGET = $(BINARY) $(SRC_CONFFILE)
59 LINT_TARGET = $(MODULE).lint
60 INSTALL_TARGET = $(BINARY) $(ROOTMODULE) $(ROOTLINK) $(ROOT_CONFFILE)
```

new/usr/src/uts/intel/dca/Makefile

2

```
62 # C99 mode is needed for dca
63 CFLAGS += $(C99_ENABLE)
64 #
65 #
66 # For now, disable these lint checks; maintainers should endeavor
67 # to investigate and remove these for maximum lint coverage.
68 # Please do not carry these forward to new Makefiles.
69 #
70 LINTTAGS += -erroff=E_BAD_PTR_CAST_ALIGN
71 LINTTAGS += -erroff=E_PTRDIFF_OVERFLOW
72 LINTTAGS += -erroff=E_ASSIGN_NARROW_CONV
73 #
74 CERRWARN += -_gcc=-Wno-parentheses
75 CERRWARN += -_gcc=-Wno-unused-variable
76 #
77 # Default build targets.
78 #
79 .KEEP_STATE:
80 #
81 def: $(DEF_DEPS)
82 #
83 all: $(ALL_DEPS)
84 #
85 clean: $(CLEAN_DEPS)
86 #
87 clobber: $(CLOBBER_DEPS)
88 #
89 lint: $(LINT_DEPS)
90 #
91 modlintlib: $(MODLINTLIB_DEPS)
92 #
93 clean.lint: $(CLEAN_LINT_DEPS)
94 #
95 install: $(INSTALL_DEPS)
96 #
97 $(ROOTLINK): $(ROOT_CRYPT_DIR) $(ROOTMODULE)
98 -$(RM) $@; ln $(ROOTMODULE) $@
99 #
100 #
101 # Include common targets.
102 #
103 include $(UTSBASE)/intel/Makefile.targ
```