FreeBSD kernel crypto code Reference Manual

Generated by Doxygen 1.4.7

Sat Feb 24 11:04:51 2007

Contents

Chapter 1

FreeBSD kernel crypto code Main Page

IMPORTANT: This API documentation may contain both functions which are public and functions that are for internal use only. Since we have not reviewed every part of the documentation yet, *some internal functions are not marked as such*. Until we finish reviewing the API documentation and add appropriate comments to functions which are only for internal use, you should take this into account. In case you want to use a function of this kernel subsystem in another kernel subsystem you should search for precedence of use outside this subsystem. If the function is not used outside this subsystem you should ask on the mailinglists about it, else you risk breaking something.

Chapter 2

FreeBSD kernel crypto code Directory Hierarchy

2.1 FreeBSD kernel crypto code Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 3

FreeBSD kernel crypto code Data Structure Index

3.1 FreeBSD kernel crypto code Data Structures

Here are the data structures with brief descriptions:

Chapter 4

FreeBSD kernel crypto code File Index

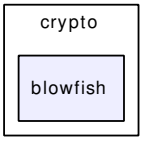
4.1 FreeBSD kernel crypto code File List

Here is a list of all files with brief descriptions:

Chapter 5

FreeBSD kernel crypto code Directory Documentation

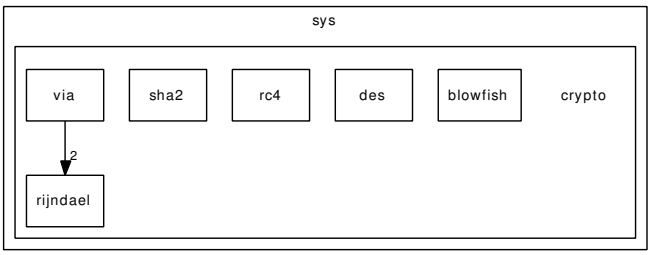
5.1 /usr/src/sys/crypto/blowfish/ Directory Reference



Files

- file bf_ecb.c
- file bf_enc.c
- file bf_locl.h
- file bf_pi.h
- file bf_skey.c
- file blowfish.h

5.2 /usr/src/sys/crypto/ Directory Reference



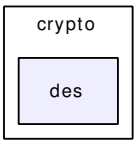
Directories

- directory blowfish
- directory des
- directory rc4
- directory rijndael
- directory sha2
- directory via

Files

- file sha1.c
- file sha1.h

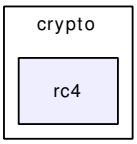
5.3 /usr/src/sys/crypto/des/ Directory Reference



Files

- file des.h
- file des_ecb.c
- file des_enc.c
- file des_locl.h
- file des_setkey.c
- file podd.h
- file sk.h
- file spr.h

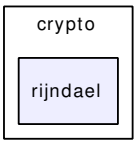
5.4 /usr/src/sys/crypto/rc4/ Directory Reference



Files

- file rc4.c
- file rc4.h

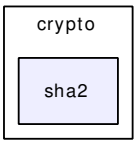
5.5 /usr/src/sys/crypto/rijndael/ Directory Reference



Files

- file rijndael-alg-fst.c
- file rijndael-api-fst.c
- file rijndael-api-fst.h
- file rijndael-api.c
- file rijndael.h
- file rijndael_local.h
- file test00.c

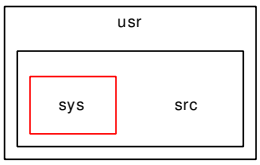
5.6 /usr/src/sys/crypto/sha2/ Directory Reference



Files

- file sha2.c
- file sha2.h

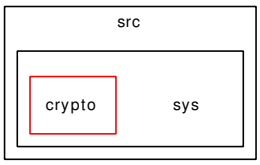
5.7 /usr/src/ Directory Reference



Directories

- directory sys

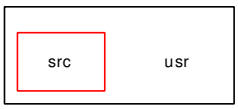
5.8 /usr/src/sys/ Directory Reference



Directories

- directory crypto

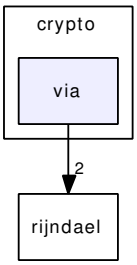
5.9 /usr/ Directory Reference



Directories

- directory src

5.10 /usr/src/sys/crypto/via/ Directory Reference



Files

- file padlock.c
- file padlock.h
- file padlock_cipher.c
- file padlock_hash.c

Chapter 6

FreeBSD kernel crypto code Data Structure Documentation

6.1 _SHA256_CTX Struct Reference

`#include <sha2.h>`

Data Fields

- u_int32_t state [8]
- u_int64_t bitcount
- u_int8_t buffer [SHA256_BLOCK_LENGTH]

6.1.1 Detailed Description

Definition at line 100 of file sha2.h.

6.1.2 Field Documentation

6.1.2.1 u_int64_t _SHA256_CTX::bitcount

Definition at line 102 of file sha2.h.

Referenced by SHA256_Final(), SHA256_Init(), and SHA256_Update().

6.1.2.2 u_int8_t _SHA256_CTX::buffer[SHA256_BLOCK_LENGTH]

Definition at line 103 of file sha2.h.

Referenced by SHA256_Final(), SHA256_Init(), SHA256_Transform(), and SHA256_Update().

6.1.2.3 u_int32_t _SHA256_CTX::state[8]

Definition at line 101 of file sha2.h.

Referenced by SHA256_Final(), SHA256_Init(), and SHA256_Transform().

The documentation for this struct was generated from the following file:

- /usr/src/sys/crypto/sha2/sha2.h

6.2 _SHA512_CTX Struct Reference

`#include <sha2.h>`

Data Fields

- u_int64_t state [8]
- u_int64_t bitcount [2]
- u_int8_t buffer [SHA512_BLOCK_LENGTH]

6.2.1 Detailed Description

Definition at line 105 of file sha2.h.

6.2.2 Field Documentation

6.2.2.1 u_int64_t _SHA512_CTX::bitcount[2]

Definition at line 107 of file sha2.h.

Referenced by SHA384_Init(), SHA512_Init(), SHA512_Last(), and SHA512_Update().

6.2.2.2 u_int8_t _SHA512_CTX::buffer[SHA512_BLOCK_LENGTH]

Definition at line 108 of file sha2.h.

Referenced by SHA384_Init(), SHA512_Init(), SHA512_Last(), SHA512_Transform(), and SHA512_-Update().

6.2.2.3 u_int64_t _SHA512_CTX::state[8]

Definition at line 106 of file sha2.h.

Referenced by SHA384_Init(), SHA512_Init(), and SHA512_Transform().

The documentation for this struct was generated from the following file:

- /usr/src/sys/crypto/sha2/sha2.h

6.3 bf_key_st Struct Reference

`#include <blowfish.h>`

Data Fields

- BF_LONG P [BF_ROUNDS+2]
- BF_LONG S [4 ∗256]

6.3.1 Detailed Description

Definition at line 78 of file blowfish.h.

6.3.2 Field Documentation

6.3.2.1 BF_LONG bf_key_st::P[BF_ROUNDS+2]

Definition at line 79 of file blowfish.h.

Referenced by BF_decrypt(), and BF_encrypt().

6.3.2.2 BF_LONG bf_key_st::S[4 ∗256]

Definition at line 80 of file blowfish.h.

Referenced by BF_decrypt(), and BF_encrypt().

The documentation for this struct was generated from the following file:

- /usr/src/sys/crypto/blowfish/blowfish.h

6.4 cipherInstance Struct Reference

`#include <rijndael-api-fst.h>`

Data Fields

- u_int8_t mode
- u_int8_t IV [RIJNDAEL_MAX_IV_SIZE]

6.4.1 Detailed Description

Definition at line 52 of file rijndael-api-fst.h.

6.4.2 Field Documentation

6.4.2.1 u_int8_t cipherInstance::IV[RIJNDAEL_MAX_IV_SIZE]

Definition at line 54 of file rijndael-api-fst.h.

Referenced by rijndael_blockDecrypt(), rijndael_blockEncrypt(), rijndael_cipherInit(), rijndael_padDecrypt(), and rijndael_padEncrypt().

6.4.2.2 u_int8_t cipherInstance::mode

Definition at line 53 of file rijndael-api-fst.h.

Referenced by rijndael_blockDecrypt(), rijndael_blockEncrypt(), rijndael_cipherInit(), rijndael_padDecrypt(), and rijndael_padEncrypt().

The documentation for this struct was generated from the following file:

- /usr/src/sys/crypto/rijndael/rijndael-api-fst.h

6.5 des_ks_struct Struct Reference

`#include <des.h>`

Data Fields

- union {
  des_cblock cblock
  DES_LONG deslong [2]
  } ks

- int weak_key

6.5.1 Detailed Description

Definition at line 62 of file des.h.

6.5.2 Field Documentation

6.5.2.1 des_cblock des_ks_struct::cblock

Definition at line 65 of file des.h.

6.5.2.2 DES_LONG des_ks_struct::deslong[2]

Definition at line 68 of file des.h.

6.5.2.3 union { ... } des_ks_struct::ks

6.5.2.4 int des_ks_struct::weak_key

Definition at line 70 of file des.h.

The documentation for this struct was generated from the following file:

- /usr/src/sys/crypto/des/des.h

6.6 keyInstance Struct Reference

`#include <rijndael-api-fst.h>`

Data Fields

- u_int8_t direction
- int keyLen
- char keyMaterial [RIJNDAEL_MAX_KEY_SIZE+1]
- int Nr
- u_int32_t rk [4 ∗(RIJNDAEL_MAXNR+1)]
- u_int32_t ek [4 ∗(RIJNDAEL_MAXNR+1)]

6.6.1 Detailed Description

Definition at line 42 of file rijndael-api-fst.h.

6.6.2 Field Documentation

6.6.2.1 u_int8_t keyInstance::direction

Definition at line 43 of file rijndael-api-fst.h.

Referenced by rijndael_blockDecrypt(), rijndael_blockEncrypt(), rijndael_makeKey(), rijndael_padDecrypt(), and rijndael_padEncrypt().

6.6.2.2 u_int32_t keyInstance::ek[4 ∗(RIJNDAEL_MAXNR+1)]

Definition at line 48 of file rijndael-api-fst.h.

Referenced by rijndael_blockDecrypt(), rijndael_blockEncrypt(), and rijndael_makeKey().

6.6.2.3 int keyInstance::keyLen

Definition at line 44 of file rijndael-api-fst.h.

Referenced by rijndael_makeKey().

6.6.2.4 char keyInstance::keyMaterial[RIJNDAEL_MAX_KEY_SIZE+1]

Definition at line 45 of file rijndael-api-fst.h.

Referenced by rijndael_makeKey().

6.6.2.5 int keyInstance::Nr

Definition at line 46 of file rijndael-api-fst.h.

Referenced by rijndael_blockDecrypt(), rijndael_blockEncrypt(), rijndael_makeKey(), rijndael_padDecrypt(), and rijndael_padEncrypt().

6.6.2.6 u_int32_t keyInstance::rk[4 ∗(RIJNDAEL_MAXNR+1)]

Definition at line 47 of file rijndael-api-fst.h.

Referenced by rijndael_blockDecrypt(), rijndael_blockEncrypt(), rijndael_makeKey(), rijndael_padDecrypt(), and rijndael_padEncrypt().

The documentation for this struct was generated from the following file:

- /usr/src/sys/crypto/rijndael/rijndael-api-fst.h

6.7 padlock_cw Union Reference

`#include <padlock.h>`

Data Fields

- uint64_t raw
- struct {
  u_int round_count: 4
  u_int algorithm_type: 3
  u_int key_generation: 1
  u_int intermediate: 1
  u_int direction: 1
  u_int key_size: 2
  u_int filler0: 20
  u_int filler1: 32
  u_int filler2: 32
  u_int filler3: 32
  } __field

6.7.1 Detailed Description

Definition at line 35 of file padlock.h.

6.7.2 Field Documentation

6.7.2.1 struct { ... } padlock_cw::__field

6.7.2.2 u_int padlock_cw::algorithm_type

Definition at line 39 of file padlock.h.

6.7.2.3 u_int padlock_cw::direction

Definition at line 42 of file padlock.h.

6.7.2.4 u_int padlock_cw::filler0

Definition at line 44 of file padlock.h.

6.7.2.5 u_int padlock_cw::filler1

Definition at line 45 of file padlock.h.

6.7.2.6 u_int padlock_cw::filler2

Definition at line 46 of file padlock.h.

6.7.2.7 u_int padlock_cw::filler3

Definition at line 47 of file padlock.h.

6.7.2.8 u_int padlock_cw::intermediate

Definition at line 41 of file padlock.h.

6.7.2.9 u_int padlock_cw::key_generation

Definition at line 40 of file padlock.h.

6.7.2.10 u_int padlock_cw::key_size

Definition at line 43 of file padlock.h.

6.7.2.11 uint64_t padlock_cw::raw

Definition at line 36 of file padlock.h.

6.7.2.12 u_int padlock_cw::round_count

Definition at line 38 of file padlock.h.

The documentation for this union was generated from the following file:

- /usr/src/sys/crypto/via/padlock.h

6.8 padlock_session Struct Reference

`#include <padlock.h>`

Public Member Functions

- padlock_cw ses_cw __aligned (16)
- uint8_t ses_iv[16] __aligned (16)
- TAILQ_ENTRY (padlock_session) ses_next

Data Fields

- auth_hash * ses_axf
- uint8_t * ses_ictx
- uint8_t * ses_octx
- int ses_mlen
- int ses_used
- uint32_t ses_id

6.8.1 Detailed Description

Definition at line 61 of file padlock.h.

6.8.2 Member Function Documentation

6.8.2.1 uint8_t ses_iv [16] padlock_session::__aligned (16)

6.8.2.2 union padlock_cw ses_cw padlock_session::__aligned (16)

6.8.2.3 padlock_session::TAILQ_ENTRY (padlock_session)

6.8.3 Field Documentation

6.8.3.1 struct auth_hash* padlock_session::ses_axf

Definition at line 66 of file padlock.h.

Referenced by padlock_authcompute(), padlock_hash_free(), padlock_hash_key_setup(), padlock_hash_-setup(), and padlock_process().

6.8.3.2 uint8_t* padlock_session::ses_ictx

Definition at line 67 of file padlock.h.

Referenced by padlock_authcompute(), padlock_hash_free(), padlock_hash_key_setup(), padlock_hash_-setup(), and padlock_process().

6.8.3.3 uint32_t padlock_session::ses_id

Definition at line 71 of file padlock.h.

Referenced by padlock_freesession(), and padlock_process().

6.8.3.4 int padlock_session::ses_mlen

Definition at line 69 of file padlock.h.

Referenced by padlock_authcompute(), and padlock_hash_setup().

6.8.3.5 uint8_t* padlock_session::ses_octx

Definition at line 68 of file padlock.h.

Referenced by padlock_authcompute(), padlock_hash_free(), padlock_hash_key_setup(), padlock_hash_-setup(), and padlock_process().

6.8.3.6 int padlock_session::ses_used

Definition at line 70 of file padlock.h.

Referenced by padlock_destroy().

The documentation for this struct was generated from the following file:

- /usr/src/sys/crypto/via/padlock.h

6.9 padlock_sha_ctx Struct Reference

Data Fields

- uint8_t * psc_buf
- int psc_offset
- int psc_size

6.9.1 Detailed Description

Definition at line 68 of file padlock_hash.c.

6.9.2 Field Documentation

6.9.2.1 uint8_t* padlock_sha_ctx::psc_buf

Definition at line 69 of file padlock_hash.c.

Referenced by padlock_copy_ctx(), padlock_sha1_final(), padlock_sha256_final(), padlock_sha_free(), padlock_sha_init(), and padlock_sha_update().

6.9.2.2 int padlock_sha_ctx::psc_offset

Definition at line 70 of file padlock_hash.c.

Referenced by padlock_copy_ctx(), padlock_sha1_final(), padlock_sha256_final(), padlock_sha_free(), padlock_sha_init(), and padlock_sha_update().

6.9.2.3 int padlock_sha_ctx::psc_size

Definition at line 71 of file padlock_hash.c.

Referenced by padlock_copy_ctx(), padlock_sha_free(), padlock_sha_init(), and padlock_sha_update().

The documentation for this struct was generated from the following file:

- /usr/src/sys/crypto/via/padlock_hash.c

6.10 padlock_softc Struct Reference

Data Fields

- int32_t sc_cid
- uint32_t sc_sid

6.10.1 Detailed Description

Definition at line 55 of file padlock.c.

6.10.2 Field Documentation

6.10.2.1 int32_t padlock_softc::sc_cid

Definition at line 56 of file padlock.c.

Referenced by padlock_destroy(), and padlock_init().

6.10.2.2 uint32_t padlock_softc::sc_sid

Definition at line 57 of file padlock.c.

Referenced by padlock_init().

The documentation for this struct was generated from the following file:

- /usr/src/sys/crypto/via/padlock.c

6.11 rc4_state Struct Reference

`#include <rc4.h>`

Data Fields

- u_char perm [256]
- u_char index1
- u_char index2

6.11.1 Detailed Description

Definition at line 43 of file rc4.h.

6.11.2 Field Documentation

6.11.2.1 u_char rc4_state::index1

Definition at line 45 of file rc4.h.

Referenced by rc4_crypt(), and rc4_init().

6.11.2.2 u_char rc4_state::index2

Definition at line 46 of file rc4.h.

Referenced by rc4_crypt(), and rc4_init().

6.11.2.3 u_char rc4_state::perm[256]

Definition at line 44 of file rc4.h.

Referenced by rc4_crypt(), and rc4_init().

The documentation for this struct was generated from the following file:

- /usr/src/sys/crypto/rc4/rc4.h

6.12 rijndael_ctx Struct Reference

`#include <rijndael.h>`

Data Fields

- int decrypt
- int Nr
- uint32_t ek [4 ∗(RIJNDAEL_MAXNR+1)]
- uint32_t dk [4 ∗(RIJNDAEL_MAXNR+1)]

6.12.1 Detailed Description

Definition at line 37 of file rijndael.h.

6.12.2 Field Documentation

6.12.2.1 int rijndael_ctx::decrypt

Definition at line 38 of file rijndael.h.

6.12.2.2 uint32_t rijndael_ctx::dk[4 ∗(RIJNDAEL_MAXNR+1)]

Definition at line 41 of file rijndael.h.

Referenced by rijndael_decrypt(), and rijndael_set_key().

6.12.2.3 uint32_t rijndael_ctx::ek[4 ∗(RIJNDAEL_MAXNR+1)]

Definition at line 40 of file rijndael.h.

Referenced by rijndael_encrypt(), and rijndael_set_key().

6.12.2.4 int rijndael_ctx::Nr

Definition at line 39 of file rijndael.h.

Referenced by rijndael_decrypt(), rijndael_encrypt(), and rijndael_set_key().

The documentation for this struct was generated from the following file:

- /usr/src/sys/crypto/rijndael/rijndael.h

6.13 sha1_ctxt Struct Reference

`#include <sha1.h>`

Data Fields

- union {
  u_int8_t b8 [20]
  u_int32_t b32 [5]
  } h

- union {
  u_int8_t b8 [8]
  u_int64_t b64 [1]
  } c

- union {
  u_int8_t b8 [64]
  u_int32_t b32 [16]
  } m

- u_int8_t count

6.13.1 Detailed Description

Definition at line 41 of file sha1.h.

6.13.2 Field Documentation

6.13.2.1 u_int32_t sha1_ctxt::b32[16]

Definition at line 44 of file sha1.h.

6.13.2.2 u_int64_t sha1_ctxt::b64[1]

Definition at line 48 of file sha1.h.

6.13.2.3 u_int8_t sha1_ctxt::b8[64]

Definition at line 43 of file sha1.h.

Referenced by sha1_step().

6.13.2.4 union { ... } sha1_ctxt::c

6.13.2.5 u_int8_t sha1_ctxt::count

Definition at line 54 of file sha1.h.

6.13.2.6 union { ... } sha1_ctxt::h

6.13.2.7 union { ... } sha1_ctxt::m

Referenced by sha1_step().

The documentation for this struct was generated from the following file:

- /usr/src/sys/crypto/sha1.h

Chapter 7

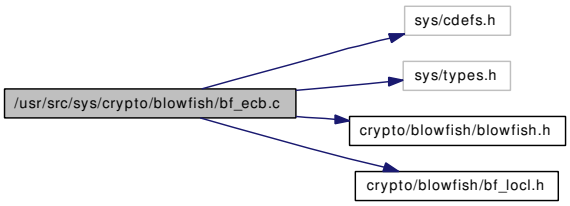
FreeBSD kernel crypto code File Documentation

7.1 notreviewed.dox File Reference

7.2 /usr/src/sys/crypto/blowfish/bf_ecb.c File Reference

```
#include <sys/cdefs.h>
#include <sys/types.h>
#include <crypto/blowfish/blowfish.h>
#include <crypto/blowfish/bf_locl.h>
```

Include dependency graph for bf_ecb.c:



Functions

- __FBSDID ("$FreeBSD: src/sys/crypto/blowfish/bf_ecb.c,v 1.1 2003/10/13 19:26:08 ume Exp $")
- void BF_ecb_encrypt (const unsigned char *in, unsigned char *out, BF_KEY *key, int encrypt)

7.2.1 Function Documentation

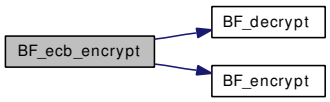
7.2.1.1 __FBSDID ("$FreeBSD: src/sys/crypto/blowfish/bf_ecb. *c*, v 1.1 2003/10/13 19:26:08 ume Exp $")

7.2.1.2 void BF_ecb_encrypt (const unsigned char * *in*, unsigned char * *out*, BF_KEY * *key*, int *encrypt*)

Definition at line 71 of file bf_ecb.c.

References BF_decrypt(), BF_encrypt(), BF_LONG, l2n, and n2l.

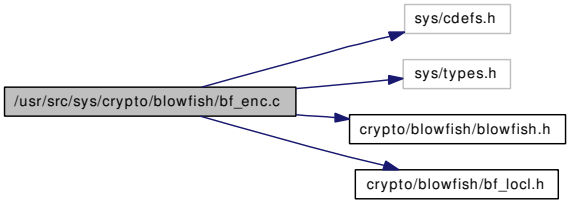
Here is the call graph for this function:



7.3 /usr/src/sys/crypto/blowfish/bf_enc.c File Reference

```
#include <sys/cdefs.h>
#include <sys/types.h>
#include <crypto/blowfish/blowfish.h>
#include <crypto/blowfish/bf_locl.h>
```

Include dependency graph for bf_enc.c:



Functions

- __FBSDID ("$FreeBSD: src/sys/crypto/blowfish/bf_enc.c,v 1.6 2003/06/10 21:38:38 obrien Exp $")
- If you set BF_ROUNDS to some value other than you will have to modify the code void BF_encrypt (BF_LONG *data, BF_KEY *key)
- void BF_decrypt (BF_LONG *data, BF_KEY *key)

Variables

- If you set BF_ROUNDS to some value other than or

7.3.1 Function Documentation

7.3.1.1 __FBSDID ("$FreeBSD: src/sys/crypto/blowfish/bf_enc. *c*, v 1.6 2003/06/10 21:38:38 obrien Exp $")

7.3.1.2 void BF_decrypt (BF_LONG * *data*, BF_KEY * *key*)

Definition at line 123 of file bf_enc.c.

References BF_ENC, BF_LONG, BF_ROUNDS, bf_key_st::P, and bf_key_st::S.

Referenced by BF_ecb_encrypt().

7.3.1.3 If you set BF_ROUNDS to some value other than you will have to modify the code void BF_encrypt (BF_LONG * *data*, BF_KEY * *key*)

Definition at line 81 of file bf_enc.c.

References BF_ENC, BF_LONG, BF_ROUNDS, bf_key_st::P, and bf_key_st::S.

Referenced by BF_ecb_encrypt(), and BF_set_key().

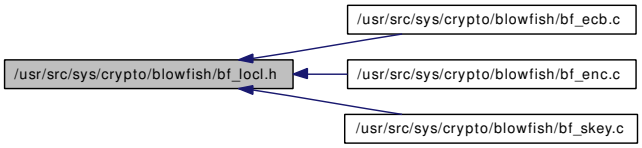
7.3.2 Variable Documentation

7.3.2.1 If you set BF_ROUNDS to some value other than or

Definition at line 75 of file bf_enc.c.

7.4 /usr/src/sys/crypto/blowfish/bf_locl.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define c2l(c, l)
- #define c2ln(c, l1, l2, n)
- #define l2c(l, c)
- #define l2cn(l1, l2, c, n)
- #define n2ln(c, l1, l2, n)
- #define l2nn(l1, l2, c, n)
- #define n2l(c, l)
- #define l2n(l, c)
- #define BF_M 0x3fc
- #define BF_0 22L
- #define BF_1 14L
- #define BF_2 6L
- #define BF_3 2L
- #define BF_ENC(LL, R, S, P)

7.4.1 Define Documentation

7.4.1.1 #define BF_0 22L

Definition at line 175 of file bf_locl.h.

7.4.1.2 #define BF_1 14L

Definition at line 176 of file bf_locl.h.

7.4.1.3 #define BF_2 6L

Definition at line 177 of file bf_locl.h.

7.4.1.4 #define BF_3 2L

Definition at line 178 of file bf_locl.h.

7.4.1.5 #define BF_ENC(LL, R, S, P)

Value:

```
LL^=P; \
        LL^=((( S[         (R>>24L)       ] + \
                S[0x0100+((R>>16L)&0xff)])^ \
                S[0x0200+((R>> 8L)&0xff)])+ \
                S[0x0300+((R     )&0xff)])&0xffffffff;
```

Definition at line 218 of file bf_locl.h.

Referenced by BF_decrypt(), and BF_encrypt().

7.4.1.6 #define BF_M 0x3fc

Definition at line 174 of file bf_locl.h.

7.4.1.7 #define c2l(c, l)

Value:

```
(l =((BF_LONG)(*((c)++)))    , \
                         l|=((BF_LONG)(*((c)++)))<< 8L, \
                         l|=((BF_LONG)(*((c)++)))<<16L, \
                         l|=((BF_LONG)(*((c)++)))<<24L)
```

Definition at line 70 of file bf_locl.h.

Referenced by des_ecb3_encrypt(), des_ecb_encrypt(), and des_set_key_unchecked().

7.4.1.8 #define c2ln(c, l1, l2, n)

Value:

```
{ \
                        c+=n; \
                        l1=l2=0; \
                        switch (n) { \
                        case 8: l2 =((BF_LONG)(*(--(c))))<<24L; \
                        case 7: l2|=((BF_LONG)(*(--(c))))<<16L; \
                        case 6: l2|=((BF_LONG)(*(--(c))))<< 8L; \
                        case 5: l2|=((BF_LONG)(*(--(c))));     \
                        case 4: l1 =((BF_LONG)(*(--(c))))<<24L; \
                        case 3: l1|=((BF_LONG)(*(--(c))))<<16L; \
                        case 2: l1|=((BF_LONG)(*(--(c))))<< 8L; \
                        case 1: l1|=((BF_LONG)(*(--(c))));     \
                                } \
                        }
```

Definition at line 77 of file bf_locl.h.

7.4.1.9 #define l2c(l, c)

Value:

```
(*((c)++)=(unsigned char)(((l)     )&0xff), \
                         *((c)++)=(unsigned char)(((l)>> 8L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>16L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>24L)&0xff))
```

Definition at line 93 of file bf_locl.h.

Referenced by des_ecb3_encrypt(), and des_ecb_encrypt().

7.4.1.10 #define l2cn(l1, l2, c, n)

Value:

```
{ \
                        c+=n; \
                        switch (n) { \
                        case 8: *(--(c))=(unsigned char)(((l2)>>24L)&0xff); \
                        case 7: *(--(c))=(unsigned char)(((l2)>>16L)&0xff); \
                        case 6: *(--(c))=(unsigned char)(((l2)>> 8L)&0xff); \
                        case 5: *(--(c))=(unsigned char)(((l2)     )&0xff); \
                        case 4: *(--(c))=(unsigned char)(((l1)>>24L)&0xff); \
                        case 3: *(--(c))=(unsigned char)(((l1)>>16L)&0xff); \
                        case 2: *(--(c))=(unsigned char)(((l1)>> 8L)&0xff); \
                        case 1: *(--(c))=(unsigned char)(((l1)     )&0xff); \
                                } \
                        }
```

Definition at line 100 of file bf_locl.h.

7.4.1.11 #define l2n(l, c)

Value:

```
(*((c)++)=(unsigned char)(((l)>>24L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>16L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>> 8L)&0xff), \
                         *((c)++)=(unsigned char)(((l)     )&0xff))
```

Definition at line 152 of file bf_locl.h.

Referenced by BF_ecb_encrypt().

7.4.1.12 #define l2nn(l1, l2, c, n)

Value:

```
{ \
                        c+=n; \
                        switch (n) { \
                        case 8: *(--(c))=(unsigned char)(((l2)    )&0xff); \
                        case 7: *(--(c))=(unsigned char)(((l2)>> 8)&0xff); \
                        case 6: *(--(c))=(unsigned char)(((l2)>>16)&0xff); \
                        case 5: *(--(c))=(unsigned char)(((l2)>>24)&0xff); \
                        case 4: *(--(c))=(unsigned char)(((l1)    )&0xff); \
                        case 3: *(--(c))=(unsigned char)(((l1)>> 8)&0xff); \
                        case 2: *(--(c))=(unsigned char)(((l1)>>16)&0xff); \
                        case 1: *(--(c))=(unsigned char)(((l1)>>24)&0xff); \
                                } \
                        }
```

Definition at line 131 of file bf_locl.h.

7.4.1.13 #define n2l(c, l)

Value:

```
(l =((BF_LONG)(*((c)++)))<<24L, \
                          l|=((BF_LONG)(*((c)++)))<<16L, \
                          l|=((BF_LONG)(*((c)++)))<< 8L, \
                          l|=((BF_LONG)(*((c)++))))
```

Definition at line 146 of file bf_locl.h.

Referenced by BF_ecb_encrypt().

7.4.1.14 #define n2ln(c, l1, l2, n)

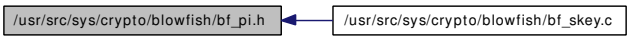
Value:

```
{ \
                        c+=n; \
                        l1=l2=0; \
                        switch (n) { \
                        case 8: l2 =((BF_LONG)(*(--(c))))    ; \
                        case 7: l2|=((BF_LONG)(*(--(c))))<< 8; \
                        case 6: l2|=((BF_LONG)(*(--(c))))<<16; \
                        case 5: l2|=((BF_LONG)(*(--(c))))<<24; \
                        case 4: l1 =((BF_LONG)(*(--(c))))    ; \
                        case 3: l1|=((BF_LONG)(*(--(c))))<< 8; \
                        case 2: l1|=((BF_LONG)(*(--(c))))<<16; \
                        case 1: l1|=((BF_LONG)(*(--(c))))<<24; \
                                } \
                        }
```

Definition at line 115 of file bf_locl.h.

7.5 /usr/src/sys/crypto/blowfish/bf_pi.h File Reference

This graph shows which files directly or indirectly include this file:



Variables

- static const BF_KEY bf_init

7.5.1 Variable Documentation

7.5.1.1 const BF_KEY bf_init [static]

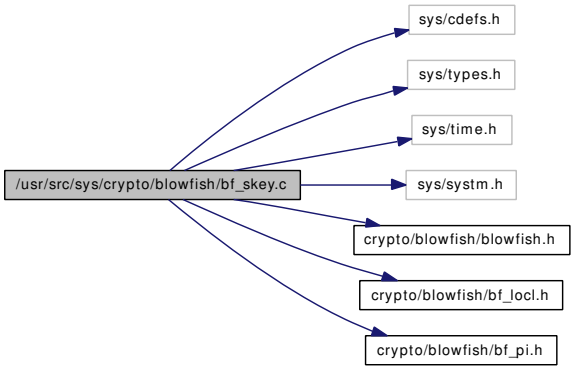
Definition at line 62 of file bf_pi.h.

Referenced by BF_set_key().

7.6 /usr/src/sys/crypto/blowfish/bf_skey.c File Reference

```
#include <sys/cdefs.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/systm.h>
#include <crypto/blowfish/blowfish.h>
#include <crypto/blowfish/bf_locl.h>
#include <crypto/blowfish/bf_pi.h>
```

Include dependency graph for bf_skey.c:



Functions

- __FBSDID ("$FreeBSD: src/sys/crypto/blowfish/bf_skey.c,v 1.6 2003/06/10 21:38:38 obrien Exp $")
- void BF_set_key (BF_KEY *key, int len, unsigned char *data)

7.6.1 Function Documentation

7.6.1.1 __FBSDID ("$FreeBSD: src/sys/crypto/blowfish/bf_skey. *c*, v 1.6 2003/06/10 21:38:38 obrien Exp $")

7.6.1.2 void BF_set_key (BF_KEY * *key*, int *len*, unsigned char * *data*)

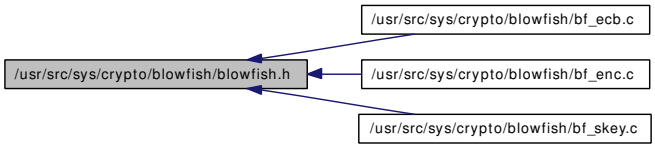
Definition at line 73 of file bf_skey.c.

References BF_encrypt(), bf_init, BF_LONG, and BF_ROUNDS.

Here is the call graph for this function:

7.7 /usr/src/sys/crypto/blowfish/blowfish.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct bf_key_st

Defines

- #define BF_ENCRYPT 1
- #define BF_DECRYPT 0
- #define BF_LONG u_int32_t
- #define BF_ROUNDS 16
- #define BF_BLOCK 8

Typedefs

- typedef bf_key_st BF_KEY

Functions

- void BF_set_key (BF_KEY *, int, unsigned char *)
- void BF_encrypt (BF_LONG *, BF_KEY *)
- void BF_decrypt (BF_LONG *, BF_KEY *)
- void BF_ecb_encrypt (const unsigned char *, unsigned char *, BF_KEY *, int)

7.7.1 Define Documentation

7.7.1.1 #define BF_BLOCK 8

Definition at line 76 of file blowfish.h.

7.7.1.2 #define BF_DECRYPT 0

Definition at line 70 of file blowfish.h.

7.7.1.3 #define BF_ENCRYPT 1

Definition at line 69 of file blowfish.h.

7.7.1.4 #define BF_LONG u_int32_t

Definition at line 73 of file blowfish.h.

Referenced by BF_decrypt(), BF_ecb_encrypt(), BF_encrypt(), and BF_set_key().

7.7.1.5 #define BF_ROUNDS 16

Definition at line 75 of file blowfish.h.

Referenced by BF_decrypt(), BF_encrypt(), and BF_set_key().

7.7.2 Typedef Documentation

7.7.2.1 typedef struct bf_key_st BF_KEY

7.7.3 Function Documentation

7.7.3.1 void BF_decrypt (BF_LONG *, BF_KEY *)

Definition at line 123 of file bf_enc.c.

References BF_ENC, BF_LONG, BF_ROUNDS, bf_key_st::P, and bf_key_st::S.

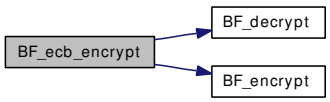
Referenced by BF_ecb_encrypt().

7.7.3.2 void BF_ecb_encrypt (const unsigned char *, unsigned char *, BF_KEY *, int)

Definition at line 71 of file bf_ecb.c.

References BF_decrypt(), BF_encrypt(), BF_LONG, l2n, and n2l.

Here is the call graph for this function:



7.7.3.3 void BF_encrypt (BF_LONG *, BF_KEY *)

Definition at line 81 of file bf_enc.c.

References BF_ENC, BF_LONG, BF_ROUNDS, bf_key_st::P, and bf_key_st::S.

Referenced by BF_ecb_encrypt(), and BF_set_key().

7.7.3.4 void BF_set_key (BF_KEY *, int, unsigned char *)

Definition at line 73 of file bf_skey.c.

References BF_encrypt(), bf_init, BF_LONG, and BF_ROUNDS.

Here is the call graph for this function:


BF_set_key
BF_encrypt

7.8 /usr/src/sys/crypto/des/des.h File Reference

This graph shows which files directly or indirectly include this file:

Data Structures

- struct des_ks_struct

Defines

- #define DES_LONG u_int32_t
- #define DES_KEY_SZ (sizeof(des_cblock))
- #define DES_SCHEDULE_SZ (sizeof(des_key_schedule))
- #define DES_ENCRYPT 1
- #define DES_DECRYPT 0
- #define DES_CBC_MODE 0
- #define DES_PCBC_MODE 1

Typedefs

- typedef unsigned char des_cblock [8]
- typedef des_ks_struct des_key_schedule [16]

Functions

- char * des_options (void)
- void des_ecb_encrypt (des_cblock *, des_cblock *, des_key_schedule, int)
- void des_encrypt1 (DES_LONG *, des_key_schedule, int)
- void des_encrypt2 (DES_LONG *, des_key_schedule, int)
- void des_encrypt3 (DES_LONG *, des_key_schedule, des_key_schedule, des_key_schedule)
- void des_decrypt3 (DES_LONG *, des_key_schedule, des_key_schedule, des_key_schedule)
- void des_ecb3_encrypt (des_cblock *, des_cblock *, des_key_schedule, des_key_schedule, des_key_schedule, int)
- void des_ncbc_encrypt (const unsigned char *, unsigned char *, long, des_key_schedule, des_cblock *, int)
- void des_ede3_cbc_encrypt (const unsigned char *, unsigned char *, long, des_key_schedule, des_key_schedule, des_key_schedule, des_cblock *, int)
- void des_set_odd_parity (des_cblock *)
- void des_fixup_key_parity (des_cblock *)
- int des_is_weak_key (des_cblock *)
- int des_set_key (des_cblock *, des_key_schedule)
- int des_key_sched (des_cblock *, des_key_schedule)

- int des_set_key_checked (des_cblock *, des_key_schedule)
- void des_set_key_unchecked (des_cblock *, des_key_schedule)
- int des_check_key_parity (des_cblock *)

Variables

- int des_check_key

7.8.1 Define Documentation

7.8.1.1 #define DES_CBC_MODE 0

Definition at line 79 of file des.h.

7.8.1.2 #define DES_DECRYPT 0

Definition at line 77 of file des.h.

Referenced by des_decrypt3(), and des_encrypt3().

7.8.1.3 #define DES_ENCRYPT 1

Definition at line 76 of file des.h.

Referenced by des_decrypt3(), and des_encrypt3().

7.8.1.4 #define DES_KEY_SZ (sizeof(des_cblock))

Definition at line 73 of file des.h.

Referenced by des_check_key_parity(), and des_set_odd_parity().

7.8.1.5 #define DES_LONG u_int32_t

Definition at line 59 of file des.h.

Referenced by des_decrypt3(), des_ecb3_encrypt(), des_ecb_encrypt(), des_encrypt1(), des_encrypt2(), des_encrypt3(), des_options(), and des_set_key_unchecked().

7.8.1.6 #define DES_PCBC_MODE 1

Definition at line 80 of file des.h.

7.8.1.7 #define DES_SCHEDULE_SZ (sizeof(des_key_schedule))

Definition at line 74 of file des.h.

7.8.2 Typedef Documentation

7.8.2.1 typedef unsigned char des_cblock[8]

Definition at line 61 of file des.h.

7.8.2.2 typedef struct des_ks_struct des_key_schedule[16]

7.8.3 Function Documentation

7.8.3.1 int des_check_key_parity (des_cblock *)

Definition at line 78 of file des_setkey.c.

References DES_KEY_SZ, and odd_parity.

Referenced by des_set_key_checked().

7.8.3.2 void des_decrypt3 (DES_LONG *, des_key_schedule, des_key_schedule, des_key_schedule)

Definition at line 279 of file des_enc.c.

References DES_DECRYPT, DES_ENCRYPT, des_encrypt2(), DES_LONG, FP, and IP.

Referenced by des_ecb3_encrypt().

Here is the call graph for this function:

7.8.3.3 void des_ecb3_encrypt (des_cblock *, des_cblock *, des_key_schedule, des_key_schedule, des_key_schedule, int)

Definition at line 116 of file des_ecb.c.

References c2l, des_decrypt3(), des_encrypt3(), DES_LONG, and l2c.

Here is the call graph for this function:

7.8.3.4 void des_ecb_encrypt (des_cblock *, des_cblock *, des_key_schedule, int)

Definition at line 100 of file des_ecb.c.

References c2l, des_encrypt1(), DES_LONG, and l2c.

Here is the call graph for this function:

7.8.3.5 void des_ede3_cbc_encrypt (const unsigned char *, unsigned char *, long, des_key_schedule, des_key_schedule, des_key_schedule, des_cblock *, int)

7.8.3.6 void des_encrypt1 (DES_LONG *, des_key_schedule, int)

Definition at line 70 of file des_enc.c.

References D_ENCRYPT, DES_LONG, des_SPtrans, FP, IP, and ROTATE.

Referenced by des_ecb_encrypt().

7.8.3.7 void des_encrypt2 (DES_LONG *, des_key_schedule, int)

Definition at line 168 of file des_enc.c.

References D_ENCRYPT, DES_LONG, des_SPtrans, and ROTATE.

Referenced by des_decrypt3(), and des_encrypt3().

7.8.3.8 void des_encrypt3 (DES_LONG *, des_key_schedule, des_key_schedule, des_key_schedule)

Definition at line 259 of file des_enc.c.

References DES_DECRYPT, DES_ENCRYPT, des_encrypt2(), DES_LONG, FP, and IP.

Referenced by des_ecb3_encrypt().

Here is the call graph for this function:

7.8.3.9 void des_fixup_key_parity (des_cblock *)

Definition at line 233 of file des_setkey.c.

References des_set_odd_parity().

Here is the call graph for this function:

7.8.3.10 int des_is_weak_key (des_cblock *)

Definition at line 120 of file des_setkey.c.

References NUM_WEAK_KEY, and weak_keys.

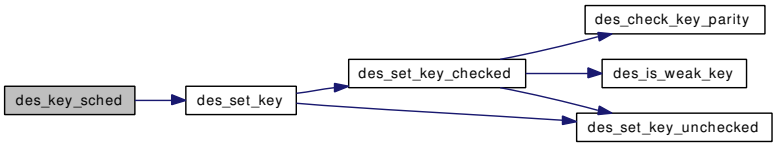
Referenced by des_set_key_checked().

7.8.3.11 int des_key_sched (des_cblock *, des_key_schedule)

Definition at line 228 of file des_setkey.c.

References des_set_key().

Here is the call graph for this function:



7.8.3.12 void des_ncbc_encrypt (const unsigned char *, unsigned char *, long, des_key_schedule, des_cblock *, int)

7.8.3.13 char* des_options (void)

Definition at line 62 of file des_ecb.c.

References DES_LONG.

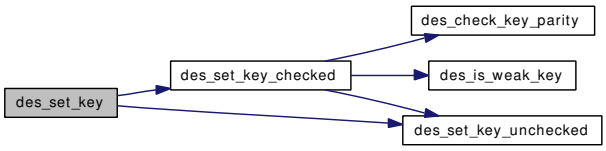
7.8.3.14 int des_set_key (des_cblock *, des_key_schedule)

Definition at line 145 of file des_setkey.c.

References des_check_key, des_set_key_checked(), and des_set_key_unchecked().

Referenced by des_key_sched().

Here is the call graph for this function:



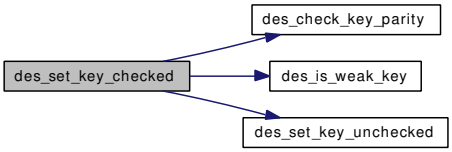
7.8.3.15 int des_set_key_checked (des_cblock *, des_key_schedule)

Definition at line 162 of file des_setkey.c.

References des_check_key_parity(), des_is_weak_key(), and des_set_key_unchecked().

Referenced by des_set_key().

Here is the call graph for this function:



7.8.3.16 void des_set_key_unchecked (des_cblock *, des_key_schedule)

Definition at line 172 of file des_setkey.c.

References c2l, DES_LONG, des_skb, HPERM_OP, ITERATIONS, PERM_OP, and ROTATE.

Referenced by des_set_key(), and des_set_key_checked().

7.8.3.17 void des_set_odd_parity (des_cblock *)

Definition at line 70 of file des_setkey.c.

References DES_KEY_SZ, and odd_parity.

Referenced by des_fixup_key_parity().

7.8.4 Variable Documentation

7.8.4.1 int des_check_key

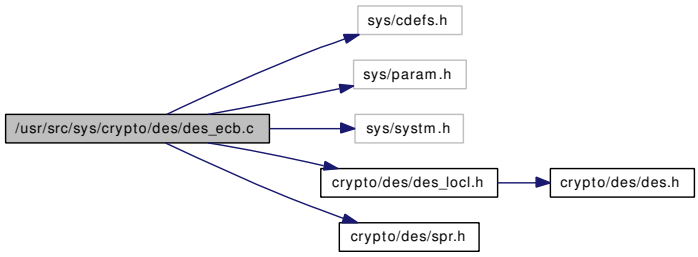
Definition at line 68 of file des_setkey.c.

Referenced by des_set_key().

7.9 /usr/src/sys/crypto/des/des_ecb.c File Reference

```
#include <sys/cdefs.h>
#include <sys/param.h>
#include <sys/systm.h>
#include <crypto/des/des_locl.h>
#include <crypto/des/spr.h>
```

Include dependency graph for des_ecb.c:



Functions

- __FBSDID ("$FreeBSD: src/sys/crypto/des/des_ecb.c,v 1.6 2004/06/14 00:38:54 obrien Exp $")
- char * des_options (void)
- void des_ecb_encrypt (des_cblock *input, des_cblock *output, des_key_schedule ks, int enc)
- void des_ecb3_encrypt (des_cblock *input, des_cblock *output, des_key_schedule ks1, des_key_schedule ks2, des_key_schedule ks3, int enc)

7.9.1 Function Documentation

7.9.1.1 __FBSDID ("$FreeBSD: src/sys/crypto/des/des_ecb. *c*, v 1.6 2004/06/14 00:38:54 obrien Exp $")

7.9.1.2 void des_ecb3_encrypt (des_cblock * *input*, des_cblock * *output*, des_key_schedule *ks1*, des_key_schedule *ks2*, des_key_schedule *ks3*, int *enc*)

Definition at line 116 of file des_ecb.c.

References c2l, des_decrypt3(), des_encrypt3(), DES_LONG, and l2c.

Here is the call graph for this function:

7.9.1.3 void des_ecb_encrypt (des_cblock * *input*, des_cblock * *output*, des_key_schedule *ks*, int *enc*)

Definition at line 100 of file des_ecb.c.

References c2l, des_encrypt1(), DES_LONG, and l2c.

Here is the call graph for this function:

7.9.1.4 char* des_options (void)

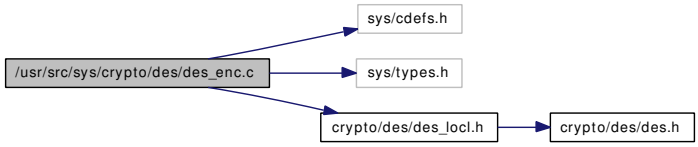
Definition at line 62 of file des_ecb.c.

References DES_LONG.

7.10 /usr/src/sys/crypto/des/des_enc.c File Reference

```
#include <sys/cdefs.h>
#include <sys/types.h>
#include <crypto/des/des_locl.h>
```

Include dependency graph for des_enc.c:



Functions

- __FBSDID ("$FreeBSD: src/sys/crypto/des/des_enc.c,v 1.2 2004/06/14 00:38:54 obrien Exp $")
- void des_encrypt1 (DES_LONG *data, des_key_schedule ks, int enc)
- void des_encrypt2 (DES_LONG *data, des_key_schedule ks, int enc)
- void des_encrypt3 (DES_LONG *data, des_key_schedule ks1, des_key_schedule ks2, des_key_schedule ks3)
- void des_decrypt3 (DES_LONG *data, des_key_schedule ks1, des_key_schedule ks2, des_key_schedule ks3)

Variables

- const DES_LONG des_SPtrans [8][64]

7.10.1 Function Documentation

7.10.1.1 __FBSDID ("$FreeBSD: src/sys/crypto/des/des_enc. *c*, v 1.2 2004/06/14 00:38:54 obrien Exp $")

7.10.1.2 void des_decrypt3 (DES_LONG * *data*, des_key_schedule *ks1*, des_key_schedule *ks2*, des_key_schedule *ks3*)

Definition at line 279 of file des_enc.c.

References DES_DECRYPT, DES_ENCRYPT, des_encrypt2(), DES_LONG, FP, and IP.

Referenced by des_ecb3_encrypt().

Here is the call graph for this function:

7.10.1.3 void des_encrypt1 (DES_LONG * *data*, des_key_schedule *ks*, int *enc*)

Definition at line 70 of file des_enc.c.

References D_ENCRYPT, DES_LONG, des_SPtrans, FP, IP, and ROTATE.

Referenced by des_ecb_encrypt().

7.10.1.4 void des_encrypt2 (DES_LONG * *data*, des_key_schedule *ks*, int *enc*)

Definition at line 168 of file des_enc.c.

References D_ENCRYPT, DES_LONG, des_SPtrans, and ROTATE.

Referenced by des_decrypt3(), and des_encrypt3().

7.10.1.5 void des_encrypt3 (DES_LONG * *data*, des_key_schedule *ks1*, des_key_schedule *ks2*, des_key_schedule *ks3*)

Definition at line 259 of file des_enc.c.

References DES_DECRYPT, DES_ENCRYPT, des_encrypt2(), DES_LONG, FP, and IP.

Referenced by des_ecb3_encrypt().

Here is the call graph for this function:

7.10.2 Variable Documentation

7.10.2.1 const DES_LONG des_SPtrans[8][64]

Definition at line 62 of file spr.h.

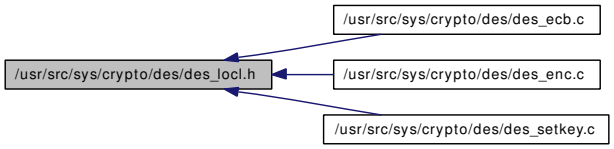
Referenced by des_encrypt1(), and des_encrypt2().

7.11 /usr/src/sys/crypto/des/des_locl.h File Reference

`#include <crypto/des/des.h>`

Include dependency graph for des_locl.h:

This graph shows which files directly or indirectly include this file:



Defines

- #define ITERATIONS 16
- #define HALF_ITERATIONS 8
- #define MAXWRITE (1024∗16)
- #define BSIZE (MAXWRITE+4)
- #define c2l(c, l)
- #define c2ln(c, l1, l2, n)
- #define l2c(l, c)
- #define HDRSIZE 4
- #define n2l(c, l)
- #define l2n(l, c)
- #define l2cn(l1, l2, c, n)
- #define ROTATE(a, n) (((a)>>(n))+((a)<<(32-(n))))
- #define LOAD_DATA_tmp(a, b, c, d, e, f) LOAD_DATA(a,b,c,d,e,f,g)
- #define LOAD_DATA(R, S, u, t, E0, E1, tmp)
- #define D_ENCRYPT(LL, R, S)
- #define PERM_OP(a, b, t, n, m)
- #define IP(l, r)
- #define FP(l, r)

7.11.1 Define Documentation

7.11.1.1 #define BSIZE (MAXWRITE+4)

Definition at line 67 of file des_locl.h.

7.11.1.2 #define c2l(c, l)

Value:

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

Definition at line 69 of file des_locl.h.

7.11.1.3 #define c2ln(c, l1, l2, n)

Value:

```
{ \
                        c+=n; \
                        l1=l2=0; \
                        switch (n) { \
                        case 8: l2 =((DES_LONG)(*(--(c))))<<24L; \
                        case 7: l2|=((DES_LONG)(*(--(c))))<<16L; \
                        case 6: l2|=((DES_LONG)(*(--(c))))<< 8L; \
                        case 5: l2|=((DES_LONG)(*(--(c))));     \
                        case 4: l1 =((DES_LONG)(*(--(c))))<<24L; \
                        case 3: l1|=((DES_LONG)(*(--(c))))<<16L; \
                        case 2: l1|=((DES_LONG)(*(--(c))))<< 8L; \
                        case 1: l1|=((DES_LONG)(*(--(c))));     \
                                } \
                        }
```

Definition at line 75 of file des_locl.h.

7.11.1.4 #define D_ENCRYPT(LL, R, S)

Value:

```
{\
        LOAD_DATA_tmp(R,S,u,t,E0,E1); \
        t=ROTATE(t,4); \
        LL^=\
                des_SPtrans[0][(u>> 2L)&0x3f]^ \
                des_SPtrans[2][(u>>10L)&0x3f]^ \
                des_SPtrans[4][(u>>18L)&0x3f]^ \
                des_SPtrans[6][(u>>26L)&0x3f]^ \
                des_SPtrans[1][(t>> 2L)&0x3f]^ \
                des_SPtrans[3][(t>>10L)&0x3f]^ \
                des_SPtrans[5][(t>>18L)&0x3f]^ \
                des_SPtrans[7][(t>>26L)&0x3f]; }
```

Definition at line 289 of file des_locl.h.

Referenced by des_encrypt1(), and des_encrypt2().

7.11.1.5 #define FP(l, r)

Value:

```
{ \
        register DES_LONG tt; \
        PERM_OP(l,r,tt, 1,0x55555555L); \
        PERM_OP(r,l,tt, 8,0x00ff00ffL); \
        PERM_OP(l,r,tt, 2,0x33333333L); \
```

```
        PERM_OP(r,l,tt,16,0x0000ffffL); \
        PERM_OP(l,r,tt, 4,0x0f0f0f0fL); \
        }
```

Definition at line 355 of file des_locl.h.

Referenced by des_decrypt3(), des_encrypt1(), and des_encrypt3().

7.11.1.6 #define HALF_ITERATIONS 8

Definition at line 63 of file des_locl.h.

7.11.1.7 #define HDRSIZE 4

Definition at line 97 of file des_locl.h.

7.11.1.8 #define IP(l, r)

Value:

```
{ \
        register DES_LONG tt; \
        PERM_OP(r,l,tt, 4,0x0f0f0f0fL); \
        PERM_OP(l,r,tt,16,0x0000ffffL); \
        PERM_OP(r,l,tt, 2,0x33333333L); \
        PERM_OP(l,r,tt, 8,0x00ff00ffL); \
        PERM_OP(r,l,tt, 1,0x55555555L); \
        }
```

Definition at line 345 of file des_locl.h.

Referenced by des_decrypt3(), des_encrypt1(), and des_encrypt3().

7.11.1.9 #define ITERATIONS 16

Definition at line 62 of file des_locl.h.

Referenced by des_set_key_unchecked().

7.11.1.10 #define l2c(l, c)

Value:

```
(*((c)++)=(unsigned char)(((l)     )&0xff), \
                         *((c)++)=(unsigned char)(((l)>> 8L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>16L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>24L)&0xff))
```

Definition at line 90 of file des_locl.h.

7.11.1.11 #define l2cn(l1, l2, c, n)

Value:

```
{ \
                        c+=n; \
                        switch (n) { \
                        case 8: *(--(c))=(unsigned char)(((l2)>>24L)&0xff); \
                        case 7: *(--(c))=(unsigned char)(((l2)>>16L)&0xff); \
                        case 6: *(--(c))=(unsigned char)(((l2)>> 8L)&0xff); \
                        case 5: *(--(c))=(unsigned char)(((l2)     )&0xff); \
                        case 4: *(--(c))=(unsigned char)(((l1)>>24L)&0xff); \
                        case 3: *(--(c))=(unsigned char)(((l1)>>16L)&0xff); \
                        case 2: *(--(c))=(unsigned char)(((l1)>> 8L)&0xff); \
                        case 1: *(--(c))=(unsigned char)(((l1)     )&0xff); \
                                } \
                        }
```

Definition at line 110 of file des_locl.h.

7.11.1.12 #define l2n(l, c)

Value:

```
(*((c)++)=(unsigned char)(((l)>>24L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>16L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>> 8L)&0xff), \
                         *((c)++)=(unsigned char)(((l)     )&0xff))
```

Definition at line 104 of file des_locl.h.

7.11.1.13 #define LOAD_DATA(R, S, u, t, E0, E1, tmp)

Value:

```
u=R^s[S  ]; \
        t=R^s[S+1]
```

Definition at line 127 of file des_locl.h.

7.11.1.14 #define LOAD_DATA_tmp(a, b, c, d, e, f) LOAD_DATA(a,b,c,d,e,f,g)

Definition at line 126 of file des_locl.h.

7.11.1.15 #define MAXWRITE (1024*16)

Definition at line 66 of file des_locl.h.

7.11.1.16 #define n2l(c, l)

Value:

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

Definition at line 99 of file des_locl.h.

7.11.1.17 #define PERM_OP(a, b, t, n, m)

Value:

```
((t)=((((a)>>(n))^(b))&(m)),\
         (b)^=(t),\
         (a)^=((t)<<(n)))
```

Definition at line 341 of file des_locl.h.

Referenced by des_set_key_unchecked().

7.11.1.18 #define ROTATE(a, n) (((a)>>(n))+((a)<<(32-(n))))

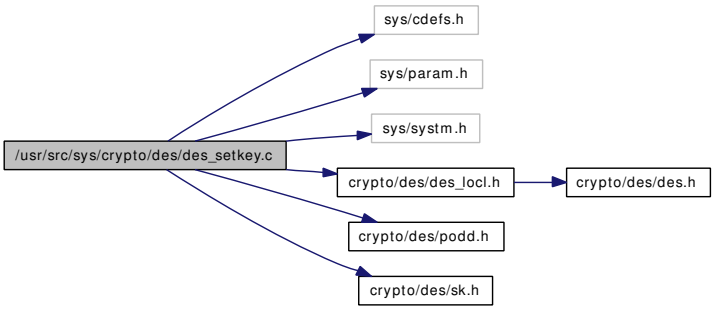
Definition at line 124 of file des_locl.h.

Referenced by des_encrypt1(), des_encrypt2(), and des_set_key_unchecked().

7.12 /usr/src/sys/crypto/des/des_setkey.c File Reference

```
#include <sys/cdefs.h>
#include <sys/param.h>
#include <sys/systm.h>
#include <crypto/des/des_locl.h>
#include <crypto/des/podd.h>
#include <crypto/des/sk.h>
```

Include dependency graph for des_setkey.c:



Defines

- #define NUM_WEAK_KEY 16
- #define HPERM_OP(a, t, n, m)

Functions

- __FBSDID ("$FreeBSD: src/sys/crypto/des/des_setkey.c,v 1.7 2004/06/14 00:38:16 obrien Exp $")
- void des_set_odd_parity (des_cblock *key)
- int des_check_key_parity (des_cblock *key)
- int des_is_weak_key (des_cblock *key)
- int des_set_key (des_cblock *key, des_key_schedule schedule)
- int des_set_key_checked (des_cblock *key, des_key_schedule schedule)
- void des_set_key_unchecked (des_cblock *key, des_key_schedule schedule)
- int des_key_sched (des_cblock *key, des_key_schedule schedule)
- void des_fixup_key_parity (des_cblock *key)

Variables

- int des_check_key = 0
- static des_cblock weak_keys [NUM_WEAK_KEY]

7.12.1 Define Documentation

7.12.1.1 #define HPERM_OP(a, t, n, m)

Value:

```
((t)=((((a)<<(16-(n)))^(a))&(m)),\
        (a)=(a)^(t)^(t>>(16-(n))))
```

Definition at line 142 of file des_setkey.c.

Referenced by des_set_key_unchecked().

7.12.1.2 #define NUM_WEAK_KEY 16

Definition at line 99 of file des_setkey.c.

Referenced by des_is_weak_key().

7.12.2 Function Documentation

7.12.2.1 __FBSDID ("$FreeBSD: src/sys/crypto/des/des_setkey. *c*, v 1.7 2004/06/14 00:38:16 obrien Exp $")

7.12.2.2 int des_check_key_parity (des_cblock * *key*)

Definition at line 78 of file des_setkey.c.

References DES_KEY_SZ, and odd_parity.

Referenced by des_set_key_checked().

7.12.2.3 void des_fixup_key_parity (des_cblock * *key*)

Definition at line 233 of file des_setkey.c.

References des_set_odd_parity().

Here is the call graph for this function:

7.12.2.4 int des_is_weak_key (des_cblock * *key*)

Definition at line 120 of file des_setkey.c.

References NUM_WEAK_KEY, and weak_keys.

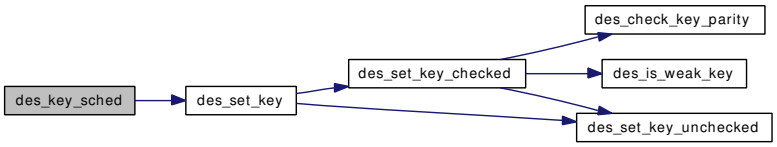
Referenced by des_set_key_checked().

7.12.2.5 int des_key_sched (des_cblock * *key*, des_key_schedule *schedule*)

Definition at line 228 of file des_setkey.c.

References des_set_key().

Here is the call graph for this function:



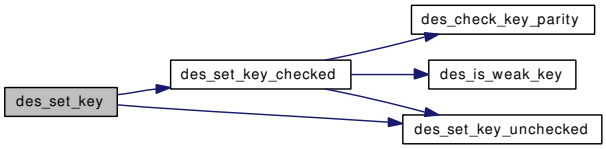
7.12.2.6 int des_set_key (des_cblock * *key*, des_key_schedule *schedule*)

Definition at line 145 of file des_setkey.c.

References des_check_key, des_set_key_checked(), and des_set_key_unchecked().

Referenced by des_key_sched().

Here is the call graph for this function:



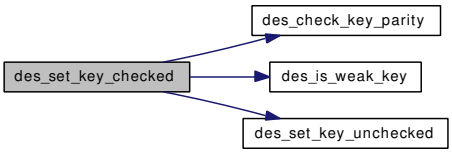
7.12.2.7 int des_set_key_checked (des_cblock * *key*, des_key_schedule *schedule*)

Definition at line 162 of file des_setkey.c.

References des_check_key_parity(), des_is_weak_key(), and des_set_key_unchecked().

Referenced by des_set_key().

Here is the call graph for this function:



7.12.2.8 void des_set_key_unchecked (des_cblock * *key*, des_key_schedule *schedule*)

Definition at line 172 of file des_setkey.c.

References c2l, DES_LONG, des_skb, HPERM_OP, ITERATIONS, PERM_OP, and ROTATE.

Referenced by des_set_key(), and des_set_key_checked().

7.12.2.9 void des_set_odd_parity (des_cblock * *key*)

Definition at line 70 of file des_setkey.c.

References DES_KEY_SZ, and odd_parity.

Referenced by des_fixup_key_parity().

7.12.3 Variable Documentation

7.12.3.1 int des_check_key = 0

Definition at line 68 of file des_setkey.c.

Referenced by des_set_key().

7.12.3.2 des_cblock weak_keys[NUM_WEAK_KEY] [static]

Initial value:

```
{

        {0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01},
        {0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE},
        {0x1F,0x1F,0x1F,0x1F,0x0E,0x0E,0x0E,0x0E},
        {0xE0,0xE0,0xE0,0xE0,0xF1,0xF1,0xF1,0xF1},

        {0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE},
        {0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01},
        {0x1F,0xE0,0x1F,0xE0,0x0E,0xF1,0x0E,0xF1},
        {0xE0,0x1F,0xE0,0x1F,0xF1,0x0E,0xF1,0x0E},
        {0x01,0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1},
        {0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1,0x01},
        {0x1F,0xFE,0x1F,0xFE,0x0E,0xFE,0x0E,0xFE},
        {0xFE,0x1F,0xFE,0x1F,0xFE,0x0E,0xFE,0x0E},
        {0x01,0x1F,0x01,0x1F,0x01,0x0E,0x01,0x0E},
        {0x1F,0x01,0x1F,0x01,0x0E,0x01,0x0E,0x01},
        {0xE0,0xFE,0xE0,0xFE,0xF1,0xFE,0xF1,0xFE},
        {0xFE,0xE0,0xFE,0xE0,0xFE,0xF1,0xFE,0xF1}}
```

Definition at line 100 of file des_setkey.c.

Referenced by des_is_weak_key().

7.13 /usr/src/sys/crypto/des/podd.h File Reference

This graph shows which files directly or indirectly include this file:

Variables

- static const unsigned char odd_parity [256]

7.13.1 Variable Documentation

7.13.1.1 const unsigned char odd_parity[256] [static]

Initial value:

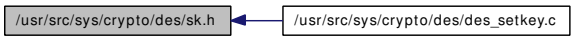
```
{
  1,  1,  2,  2,  4,  4,  7,  7,  8,  8, 11, 11, 13, 13, 14, 14,
 16, 16, 19, 19, 21, 21, 22, 22, 25, 25, 26, 26, 28, 28, 31, 31,
 32, 32, 35, 35, 37, 37, 38, 38, 41, 41, 42, 42, 44, 44, 47, 47,
 49, 49, 50, 50, 52, 52, 55, 55, 56, 56, 59, 59, 61, 61, 62, 62,
 64, 64, 67, 67, 69, 69, 70, 70, 73, 73, 74, 74, 76, 76, 79, 79,
 81, 81, 82, 82, 84, 84, 87, 87, 88, 88, 91, 91, 93, 93, 94, 94,
 97, 97, 98, 98,100,100,103,103,104,104,107,107,109,109,110,110,
112,112,115,115,117,117,118,118,121,121,122,122,124,124,127,127,
128,128,131,131,133,133,134,134,137,137,138,138,140,140,143,143,
145,145,146,146,148,148,151,151,152,152,155,155,157,157,158,158,
161,161,162,162,164,164,167,167,168,168,171,171,173,173,174,174,
176,176,179,179,181,181,182,182,185,185,186,186,188,188,191,191,
193,193,194,194,196,196,199,199,200,200,203,203,205,205,206,206,
208,208,211,211,213,213,214,214,217,217,218,218,220,220,223,223,
224,224,227,227,229,229,230,230,233,233,234,234,236,236,239,239,
241,241,242,242,244,244,247,247,248,248,251,251,253,253,254,254}
```

Definition at line 51 of file podd.h.

Referenced by des_check_key_parity(), and des_set_odd_parity().

7.14 /usr/src/sys/crypto/des/sk.h File Reference

This graph shows which files directly or indirectly include this file:



Variables

- static const DES_LONG des_skb [8][64]

7.14.1 Variable Documentation

7.14.1.1 const DES_LONG des_skb[8][64] [static]

Definition at line 51 of file sk.h.

Referenced by des_set_key_unchecked().

7.15 /usr/src/sys/crypto/des/spr.h File Reference

This graph shows which files directly or indirectly include this file:

Variables

- const DES_LONG des_SPtrans [8][64]

7.15.1 Variable Documentation

7.15.1.1 const DES_LONG des_SPtrans[8][64]

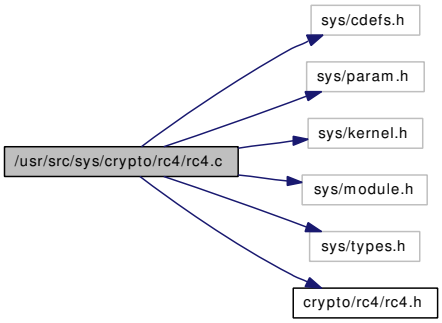
Definition at line 62 of file spr.h.

Referenced by des_encrypt1(), and des_encrypt2().

7.16 /usr/src/sys/crypto/rc4/rc4.c File Reference

```
#include <sys/cdefs.h>
#include <sys/param.h>
#include <sys/kernel.h>
#include <sys/module.h>
#include <sys/types.h>
#include <crypto/rc4/rc4.h>
```

Include dependency graph for rc4.c:



Functions

- __FBSDID ("$FreeBSD: src/sys/crypto/rc4/rc4.c,v 1.5 2003/06/10 21:43:12 obrien Exp $")
- static __inline void swap_bytes (u_char *a, u_char *b)
- void rc4_init (struct rc4_state *const state, const u_char *key, int keylen)
- void rc4_crypt (struct rc4_state *const state, const u_char *inbuf, u_char *outbuf, int buflen)
- static int rc4_modevent (module_t mod, int type, void *unused)
- DECLARE_MODULE (rc4, rc4_mod, SI_SUB_DRIVERS, SI_ORDER_FIRST)
- MODULE_VERSION (rc4, 1)

Variables

- static moduledata_t rc4_mod

7.16.1 Function Documentation

7.16.1.1 __FBSDID ("$FreeBSD: src/sys/crypto/rc4/rc4. *c*, v 1.5 2003/06/10 21:43:12 obrien Exp $")

7.16.1.2 DECLARE_MODULE (rc4, rc4_mod, SI_SUB_DRIVERS, SI_ORDER_FIRST)

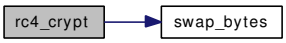
7.16.1.3 MODULE_VERSION (rc4, 1)

7.16.1.4 void rc4_crypt (struct rc4_state *const *state*, const u_char * *inbuf*, u_char * *outbuf*, int *buflen*)

Definition at line 86 of file rc4.c.

References rc4_state::index1, rc4_state::index2, rc4_state::perm, and swap_bytes().

Here is the call graph for this function:

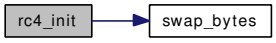


7.16.1.5 void rc4_init (struct rc4_state *const *state*, const u_char * *key*, int *keylen*)

Definition at line 61 of file rc4.c.

References rc4_state::index1, rc4_state::index2, rc4_state::perm, and swap_bytes().

Here is the call graph for this function:



7.16.1.6 static int rc4_modevent (module_t *mod*, int *type*, void * *unused*) [static]

Definition at line 109 of file rc4.c.

7.16.1.7 static __inline void swap_bytes (u_char * *a*, u_char * *b*) [static]

Definition at line 47 of file rc4.c.

Referenced by rc4_crypt(), and rc4_init().

7.16.2 Variable Documentation

7.16.2.1 moduledata_t rc4_mod [static]

Initial value:

```
{
        "rc4",
```

```
        rc4_modevent,
        0
}
```

Definition at line 120 of file rc4.c.

7.17 /usr/src/sys/crypto/rc4/rc4.h File Reference

This graph shows which files directly or indirectly include this file:

Data Structures

- struct rc4_state

Functions

- void rc4_init (struct rc4_state *state, const u_char *key, int keylen)
- void rc4_crypt (struct rc4_state *state, const u_char *inbuf, u_char *outbuf, int buflen)

7.17.1 Function Documentation

7.17.1.1 void rc4_crypt (struct rc4_state * *state*, const u_char * *inbuf*, u_char * *outbuf*, int *buflen*)

Definition at line 86 of file rc4.c.

References rc4_state::index1, rc4_state::index2, rc4_state::perm, and swap_bytes().

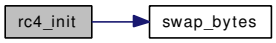
Here is the call graph for this function:

7.17.1.2 void rc4_init (struct rc4_state * *state*, const u_char * *key*, int *keylen*)

Definition at line 61 of file rc4.c.

References rc4_state::index1, rc4_state::index2, rc4_state::perm, and swap_bytes().

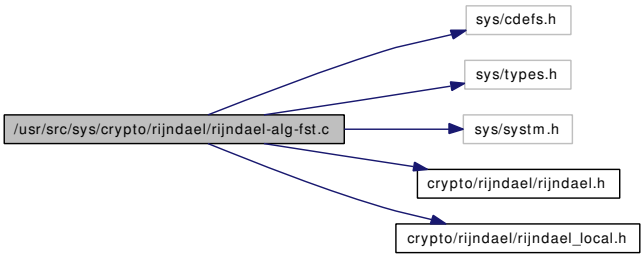
Here is the call graph for this function:



7.18 /usr/src/sys/crypto/rijndael/rijndael-alg-fst.c File Reference

```
#include <sys/cdefs.h>
#include <sys/types.h>
#include <sys/systm.h>
#include <crypto/rijndael/rijndael.h>
#include <crypto/rijndael/rijndael_local.h>
```

Include dependency graph for rijndael-alg-fst.c:



Defines

- #define SWAP(x) (_lrotl(x, 8) & 0x00ff00ff | _lrotr(x, 8) & 0xff00ff00)
- #define GETU32(pt) (((u32)(pt)[0] << 24) ^ ((u32)(pt)[1] << 16) ^ ((u32)(pt)[2] << 8) ^ ((u32)(pt)[3]))
- #define PUTU32(ct, st) { (ct)[0] = (u8)((st) >> 24); (ct)[1] = (u8)((st) >> 16); (ct)[2] = (u8)((st) >> 8); (ct)[3] = (u8)(st); }

Functions

- __FBSDID ("$FreeBSD: src/sys/crypto/rijndael/rijndael-alg-fst.c,v 1.11 2005/08/17 07:59:07 pjd Exp $")
- int rijndaelKeySetupEnc (u32 rk[ ], const u8 cipherKey[ ], int keyBits)
- int rijndaelKeySetupDec (u32 rk[ ], const u8 cipherKey[ ], int keyBits)
- void rijndaelEncrypt (const u32 rk[ ], int Nr, const u8 pt[16], u8 ct[16])
- void rijndaelDecrypt (const u32 rk[ ], int Nr, const u8 ct[16], u8 pt[16])

Variables

- static const u32 Te0 [256]
- static const u32 Te1 [256]
- static const u32 Te2 [256]
- static const u32 Te3 [256]
- static const u32 Te4 [256]
- static const u32 Td0 [256]
- static const u32 Td1 [256]
- static const u32 Td2 [256]
- static const u32 Td3 [256]
- static const u32 Td4 [256]
- static const u32 rcon [ ]

7.18.1 Define Documentation

7.18.1.1 #define GETU32(pt) (((u32)(pt)[0] << 24) ^ ((u32)(pt)[1] << 16) ^ ((u32)(pt)[2] << 8) ^ ((u32)(pt)[3]))

Definition at line 727 of file rijndael-alg-fst.c.

Referenced by rijndaelDecrypt(), rijndaelEncrypt(), and rijndaelKeySetupEnc().

7.18.1.2 #define PUTU32(ct, st) { (ct)[0] = (u8)((st) >> 24); (ct)[1] = (u8)((st) >> 16); (ct)[2] = (u8)((st) >> 8); (ct)[3] = (u8)(st); }

Definition at line 728 of file rijndael-alg-fst.c.

Referenced by rijndaelDecrypt(), and rijndaelEncrypt().

7.18.1.3 #define SWAP(x) (_lrotl(x, 8) & 0x00ff00ff | _lrotr(x, 8) & 0xff00ff00)

Definition at line 725 of file rijndael-alg-fst.c.

7.18.2 Function Documentation

7.18.2.1 __FBSDID ("$FreeBSD: src/sys/crypto/rijndael/rijndael-alg-fst. *c*, v 1.11 2005/08/17 07:59:07 pjd Exp $")

rijndael-alg-fst.c

Version:

3.0 (December 2000)

Optimised ANSI C code for the Rijndael cipher (now AES)

Author:

Vincent Rijmen <vincent.rijmen@esat.kuleuven.ac.be>
Antoon Bosselaers <antoon.bosselaers@esat.kuleuven.ac.be>
Paulo Barreto <paulo.barreto@terra.com.br>

7.18.2.2 void rijndaelDecrypt (const u32 *rk*[ ], int *Nr*, const u8 *ct*[16], u8 *pt*[16])

Definition at line 1044 of file rijndael-alg-fst.c.

References GETU32, PUTU32, Td0, Td1, Td2, Td3, and Td4.

Referenced by rijndael_blockDecrypt(), rijndael_decrypt(), and rijndael_padDecrypt().

7.18.2.3 void rijndaelEncrypt (const u32 *rk*[ ], int *Nr*, const u8 *pt*[16], u8 *ct*[16])

Definition at line 863 of file rijndael-alg-fst.c.

References GETU32, PUTU32, Te0, Te1, Te2, Te3, and Te4.

Referenced by rijndael_blockDecrypt(), rijndael_blockEncrypt(), rijndael_encrypt(), and rijndael_padEncrypt().

7.18.2.4 int rijndaelKeySetupDec (u32 *rk*[ ], const u8 *cipherKey*[ ], int *keyBits*)

Expand the cipher key into the decryption key schedule.

Returns:

the number of rounds for the given cipher key size.

Definition at line 823 of file rijndael-alg-fst.c.

References rijndaelKeySetupEnc(), Td0, Td1, Td2, Td3, and Te4.

Referenced by padlock_cipher_key_setup(), rijndael_makeKey(), and rijndael_set_key().

Here is the call graph for this function:

7.18.2.5 int rijndaelKeySetupEnc (u32 *rk*[ ], const u8 *cipherKey*[ ], int *keyBits*)

Expand the cipher key into the encryption key schedule.

Returns:

the number of rounds for the given cipher key size.

Definition at line 735 of file rijndael-alg-fst.c.

References GETU32, rcon, and Te4.

Referenced by padlock_cipher_key_setup(), rijndael_makeKey(), rijndael_set_key(), and rijndaelKeySetupDec().

7.18.3 Variable Documentation

7.18.3.1 const u32 rcon[ ] [static]

Initial value:

```
 {
        0x01000000, 0x02000000, 0x04000000, 0x08000000,
        0x10000000, 0x20000000, 0x40000000, 0x80000000,
        0x1B000000, 0x36000000,
}
```

Definition at line 719 of file rijndael-alg-fst.c.

Referenced by rijndaelKeySetupEnc().

7.18.3.2 const u32 Td0[256] [static]

Definition at line 388 of file rijndael-alg-fst.c.

Referenced by rijndaelDecrypt(), and rijndaelKeySetupDec().

7.18.3.3 const u32 Td1[256] [static]

Definition at line 454 of file rijndael-alg-fst.c.

Referenced by rijndaelDecrypt(), and rijndaelKeySetupDec().

7.18.3.4 const u32 Td2[256] [static]

Definition at line 520 of file rijndael-alg-fst.c.

Referenced by rijndaelDecrypt(), and rijndaelKeySetupDec().

7.18.3.5 const u32 Td3[256] [static]

Definition at line 587 of file rijndael-alg-fst.c.

Referenced by rijndaelDecrypt(), and rijndaelKeySetupDec().

7.18.3.6 const u32 Td4[256] [static]

Definition at line 653 of file rijndael-alg-fst.c.

Referenced by rijndaelDecrypt().

7.18.3.7 const u32 Te0[256] [static]

Definition at line 57 of file rijndael-alg-fst.c.

Referenced by rijndaelEncrypt().

7.18.3.8 const u32 Te1[256] [static]

Definition at line 123 of file rijndael-alg-fst.c.

Referenced by rijndaelEncrypt().

7.18.3.9 const u32 Te2[256] [static]

Definition at line 189 of file rijndael-alg-fst.c.

Referenced by rijndaelEncrypt().

7.18.3.10 const u32 Te3[256] [static]

Definition at line 255 of file rijndael-alg-fst.c.

Referenced by rijndaelEncrypt().

7.18.3.11 const u32 Te4[256] [static]

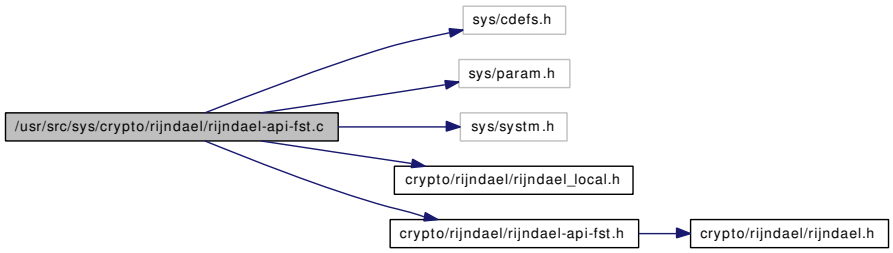
Definition at line 322 of file rijndael-alg-fst.c.

Referenced by rijndaelEncrypt(), rijndaelKeySetupDec(), and rijndaelKeySetupEnc().

7.19 /usr/src/sys/crypto/rijndael/rijndael-api-fst.c File Reference

```
#include <sys/cdefs.h>
#include <sys/param.h>
#include <sys/systm.h>
#include <crypto/rijndael/rijndael_local.h>
#include <crypto/rijndael/rijndael-api-fst.h>
```

Include dependency graph for rijndael-api-fst.c:



Defines

- #define TRUE 1

Typedefs

- typedef u_int8_t BYTE

Functions

- __FBSDID ("$FreeBSD: src/sys/crypto/rijndael/rijndael-api-fst.c,v 1.12 2005/03/11 16:26:10 ume Exp $")
- int rijndael_makeKey (keyInstance *key, BYTE direction, int keyLen, char *keyMaterial)
- int rijndael_cipherInit (cipherInstance *cipher, BYTE mode, char *IV)
- int rijndael_blockEncrypt (cipherInstance *cipher, keyInstance *key, BYTE *input, int inputLen, BYTE *outBuffer)
- int rijndael_padEncrypt (cipherInstance *cipher, keyInstance *key, BYTE *input, int inputOctets, BYTE *outBuffer)
- int rijndael_blockDecrypt (cipherInstance *cipher, keyInstance *key, BYTE *input, int inputLen, BYTE *outBuffer)
- int rijndael_padDecrypt (cipherInstance *cipher, keyInstance *key, BYTE *input, int inputOctets, BYTE *outBuffer)

7.19.1 Define Documentation

7.19.1.1 #define TRUE 1

Definition at line 32 of file rijndael-api-fst.c.

Referenced by rijndael_cipherInit(), and rijndael_makeKey().

7.19.2 Typedef Documentation

7.19.2.1 typedef u_int8_t BYTE

Definition at line 35 of file rijndael-api-fst.c.

7.19.3 Function Documentation

7.19.3.1 __FBSDID ("$FreeBSD: src/sys/crypto/rijndael/rijndael-api-fst. *c*, v 1.12 2005/03/11 16:26:10 ume Exp $")

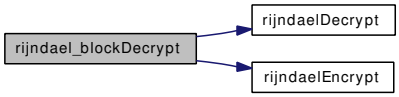
7.19.3.2 int rijndael_blockDecrypt (cipherInstance * *cipher*, keyInstance * *key*, BYTE * *input*, int *inputLen*, BYTE * *outBuffer*)

Definition at line 263 of file rijndael-api-fst.c.

References BAD_CIPHER_STATE, DIR_ENCRYPT, keyInstance::direction, keyInstance::ek, cipherInstance::IV, cipherInstance::mode, MODE_CBC, MODE_CFB1, MODE_ECB, keyInstance::Nr, rijndaelDecrypt(), rijndaelEncrypt(), and keyInstance::rk.

Referenced by main().

Here is the call graph for this function:

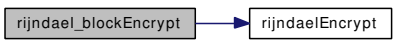


7.19.3.3 int rijndael_blockEncrypt (cipherInstance * *cipher*, keyInstance * *key*, BYTE * *input*, int *inputLen*, BYTE * *outBuffer*)

Definition at line 85 of file rijndael-api-fst.c.

References BAD_CIPHER_STATE, DIR_DECRYPT, keyInstance::direction, keyInstance::ek, cipherInstance::IV, cipherInstance::mode, MODE_CBC, MODE_CFB1, MODE_ECB, keyInstance::Nr, rijndaelEncrypt(), and keyInstance::rk.

Here is the call graph for this function:



7.19.3.4 int rijndael_cipherInit (cipherInstance * *cipher*, BYTE *mode*, char * *IV*)

Definition at line 71 of file rijndael-api-fst.c.

References BAD_CIPHER_MODE, cipherInstance::IV, cipherInstance::mode, MODE_CBC, MODE_CFB1, MODE_ECB, RIJNDAEL_MAX_IV_SIZE, and TRUE.

Referenced by main().

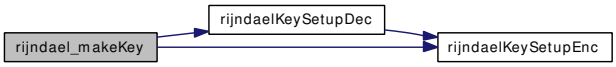
7.19.3.5 int rijndael_makeKey (keyInstance * *key*, BYTE *direction*, int *keyLen*, char * *keyMaterial*)

Definition at line 37 of file rijndael-api-fst.c.

References BAD_KEY_DIR, BAD_KEY_INSTANCE, BAD_KEY_MAT, DIR_DECRYPT, DIR_ENCRYPT, keyInstance::direction, keyInstance::ek, keyInstance::keyLen, keyInstance::keyMaterial, keyInstance::Nr, RIJNDAEL_MAXKB, rijndaelKeySetupDec(), rijndaelKeySetupEnc(), keyInstance::rk, and TRUE.

Referenced by main().

Here is the call graph for this function:



7.19.3.6 int rijndael_padDecrypt (cipherInstance * *cipher*, keyInstance * *key*, BYTE * *input*, int *inputOctets*, BYTE * *outBuffer*)

Definition at line 362 of file rijndael-api-fst.c.

References BAD_CIPHER_STATE, BAD_DATA, DIR_ENCRYPT, keyInstance::direction, cipherInstance::IV, cipherInstance::mode, MODE_CBC, MODE_ECB, keyInstance::Nr, rijndaelDecrypt(), and keyInstance::rk.

Here is the call graph for this function:

7.19.3.7 int rijndael_padEncrypt (cipherInstance * *cipher*, keyInstance * *key*, BYTE * *input*, int *inputOctets*, BYTE * *outBuffer*)

Encrypt data partitioned in octets, using RFC 2040-like padding.

Parameters:

input data to be encrypted (octet sequence)

inputOctets input length in octets (not bits)

outBuffer encrypted output data

Returns:

length in octets (not bits) of the encrypted output buffer.

Definition at line 200 of file rijndael-api-fst.c.

References BAD_CIPHER_STATE, DIR_DECRYPT, keyInstance::direction, cipherInstance::IV, cipherInstance::mode, MODE_CBC, MODE_ECB, keyInstance::Nr, rijndaelEncrypt(), and keyInstance::rk.

Here is the call graph for this function:


rijndael_padEncrypt
rijndaelEncrypt

7.20 /usr/src/sys/crypto/rijndael/rijndael-api-fst.h File Reference

`#include <crypto/rijndael/rijndael.h>`

Include dependency graph for rijndael-api-fst.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct keyInstance
- struct cipherInstance

Defines

- #define DIR_ENCRYPT 0
- #define DIR_DECRYPT 1
- #define MODE_ECB 1
- #define MODE_CBC 2
- #define MODE_CFB1 3
- #define BITSPERBLOCK 128
- #define BAD_KEY_DIR -1
- #define BAD_KEY_MAT -2
- #define BAD_KEY_INSTANCE -3
- #define BAD_CIPHER_MODE -4
- #define BAD_CIPHER_STATE -5
- #define BAD_BLOCK_LENGTH -6
- #define BAD_CIPHER_INSTANCE -7
- #define BAD_DATA -8
- #define BAD_OTHER -9
- #define RIJNDAEL_MAX_KEY_SIZE 64
- #define RIJNDAEL_MAX_IV_SIZE 16

Functions

- int rijndael_makeKey (keyInstance *, u_int8_t, int, char *)
- int rijndael_cipherInit (cipherInstance *, u_int8_t, char *)
- int rijndael_blockEncrypt (cipherInstance *, keyInstance *, u_int8_t *, int, u_int8_t *)
- int rijndael_padEncrypt (cipherInstance *, keyInstance *, u_int8_t *, int, u_int8_t *)
- int rijndael_blockDecrypt (cipherInstance *, keyInstance *, u_int8_t *, int, u_int8_t *)
- int rijndael_padDecrypt (cipherInstance *, keyInstance *, u_int8_t *, int, u_int8_t *)

7.20.1 Define Documentation

7.20.1.1 #define BAD_BLOCK_LENGTH -6

Definition at line 30 of file rijndael-api-fst.h.

7.20.1.2 #define BAD_CIPHER_INSTANCE -7

Definition at line 31 of file rijndael-api-fst.h.

7.20.1.3 #define BAD_CIPHER_MODE -4

Definition at line 28 of file rijndael-api-fst.h.

Referenced by rijndael_cipherInit().

7.20.1.4 #define BAD_CIPHER_STATE -5

Definition at line 29 of file rijndael-api-fst.h.

Referenced by rijndael_blockDecrypt(), rijndael_blockEncrypt(), rijndael_padDecrypt(), and rijndael_padEncrypt().

7.20.1.5 #define BAD_DATA -8

Definition at line 32 of file rijndael-api-fst.h.

Referenced by rijndael_padDecrypt().

7.20.1.6 #define BAD_KEY_DIR -1

Definition at line 25 of file rijndael-api-fst.h.

Referenced by rijndael_makeKey().

7.20.1.7 #define BAD_KEY_INSTANCE -3

Definition at line 27 of file rijndael-api-fst.h.

Referenced by rijndael_makeKey().

7.20.1.8 #define BAD_KEY_MAT -2

Definition at line 26 of file rijndael-api-fst.h.

Referenced by rijndael_makeKey().

7.20.1.9 #define BAD_OTHER -9

Definition at line 33 of file rijndael-api-fst.h.

7.20.1.10 #define BITSPERBLOCK 128

Definition at line 22 of file rijndael-api-fst.h.

7.20.1.11 #define DIR_DECRYPT 1

Definition at line 18 of file rijndael-api-fst.h.

Referenced by main(), rijndael_blockEncrypt(), rijndael_makeKey(), and rijndael_padEncrypt().

7.20.1.12 #define DIR_ENCRYPT 0

Definition at line 17 of file rijndael-api-fst.h.

Referenced by rijndael_blockDecrypt(), rijndael_makeKey(), and rijndael_padDecrypt().

7.20.1.13 #define MODE_CBC 2

Definition at line 20 of file rijndael-api-fst.h.

Referenced by main(), rijndael_blockDecrypt(), rijndael_blockEncrypt(), rijndael_cipherInit(), rijndael_padDecrypt(), and rijndael_padEncrypt().

7.20.1.14 #define MODE_CFB1 3

Definition at line 21 of file rijndael-api-fst.h.

Referenced by rijndael_blockDecrypt(), rijndael_blockEncrypt(), and rijndael_cipherInit().

7.20.1.15 #define MODE_ECB 1

Definition at line 19 of file rijndael-api-fst.h.

Referenced by rijndael_blockDecrypt(), rijndael_blockEncrypt(), rijndael_cipherInit(), rijndael_padDecrypt(), and rijndael_padEncrypt().

7.20.1.16 #define RIJNDAEL_MAX_IV_SIZE 16

Definition at line 37 of file rijndael-api-fst.h.

Referenced by rijndael_cipherInit().

7.20.1.17 #define RIJNDAEL_MAX_KEY_SIZE 64

Definition at line 36 of file rijndael-api-fst.h.

7.20.2 Function Documentation

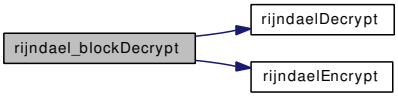
7.20.2.1 int rijndael_blockDecrypt (cipherInstance *, keyInstance *, u_int8_t *, int, u_int8_t *)

Definition at line 263 of file rijndael-api-fst.c.

References BAD_CIPHER_STATE, DIR_ENCRYPT, keyInstance::direction, keyInstance::ek, cipherInstance::IV, cipherInstance::mode, MODE_CBC, MODE_CFB1, MODE_ECB, keyInstance::Nr, rijndaelDecrypt(), rijndaelEncrypt(), and keyInstance::rk.

Referenced by main().

Here is the call graph for this function:

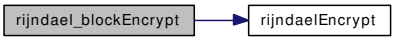


7.20.2.2 int rijndael_blockEncrypt (cipherInstance *, keyInstance *, u_int8_t *, int, u_int8_t *)

Definition at line 85 of file rijndael-api-fst.c.

References BAD_CIPHER_STATE, DIR_DECRYPT, keyInstance::direction, keyInstance::ek, cipherInstance::IV, cipherInstance::mode, MODE_CBC, MODE_CFB1, MODE_ECB, keyInstance::Nr, rijndaelEncrypt(), and keyInstance::rk.

Here is the call graph for this function:



7.20.2.3 int rijndael_cipherInit (cipherInstance *, u_int8_t, char *)

Definition at line 71 of file rijndael-api-fst.c.

References BAD_CIPHER_MODE, cipherInstance::IV, cipherInstance::mode, MODE_CBC, MODE_CFB1, MODE_ECB, RIJNDAEL_MAX_IV_SIZE, and TRUE.

Referenced by main().

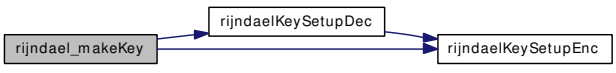
7.20.2.4 int rijndael_makeKey (keyInstance *, u_int8_t, int, char *)

Definition at line 37 of file rijndael-api-fst.c.

References BAD_KEY_DIR, BAD_KEY_INSTANCE, BAD_KEY_MAT, DIR_DECRYPT, DIR_ENCRYPT, keyInstance::direction, keyInstance::ek, keyInstance::keyLen, keyInstance::keyMaterial, keyInstance::Nr, RIJNDAEL_MAXKB, rijndaelKeySetupDec(), rijndaelKeySetupEnc(), keyInstance::rk, and TRUE.

Referenced by main().

Here is the call graph for this function:



7.20.2.5 int rijndael_padDecrypt (cipherInstance *, keyInstance *, u_int8_t *, int, u_int8_t *)

Definition at line 362 of file rijndael-api-fst.c.

References BAD_CIPHER_STATE, BAD_DATA, DIR_ENCRYPT, keyInstance::direction, cipherInstance::IV, cipherInstance::mode, MODE_CBC, MODE_ECB, keyInstance::Nr, rijndaelDecrypt(), and keyInstance::rk.

Here is the call graph for this function:

7.20.2.6 int rijndael_padEncrypt (cipherInstance * *cipher*, keyInstance * *key*, BYTE * *input*, int *inputOctets*, BYTE * *outBuffer*)

Encrypt data partitioned in octets, using RFC 2040-like padding.

Parameters:

input data to be encrypted (octet sequence)

inputOctets input length in octets (not bits)

outBuffer encrypted output data

Returns:

length in octets (not bits) of the encrypted output buffer.

Definition at line 200 of file rijndael-api-fst.c.

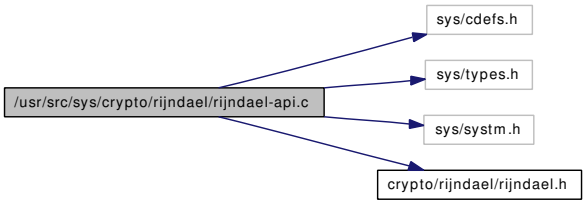
References BAD_CIPHER_STATE, DIR_DECRYPT, keyInstance::direction, cipherInstance::IV, cipherInstance::mode, MODE_CBC, MODE_ECB, keyInstance::Nr, rijndaelEncrypt(), and keyInstance::rk.

Here is the call graph for this function:

7.21 /usr/src/sys/crypto/rijndael/rijndael-api.c File Reference

```
#include <sys/cdefs.h>
#include <sys/types.h>
#include <sys/systm.h>
#include <crypto/rijndael/rijndael.h>
```

Include dependency graph for rijndael-api.c:



Functions

- __FBSDID ("$FreeBSD: src/sys/crypto/rijndael/rijndael-api.c,v 1.1 2003/11/12 04:22:37 ume Exp $")
- void rijndael_set_key (rijndael_ctx *ctx, const u_char *key, int bits)
- void rijndael_decrypt (const rijndael_ctx *ctx, const u_char *src, u_char *dst)
- void rijndael_encrypt (const rijndael_ctx *ctx, const u_char *src, u_char *dst)

7.21.1 Function Documentation

7.21.1.1 __FBSDID ("$FreeBSD: src/sys/crypto/rijndael/rijndael-api. *c*, v 1.1 2003/11/12 04:22:37 ume Exp $")

7.21.1.2 void rijndael_decrypt (const rijndael_ctx * *ctx*, const u_char * *src*, u_char * *dst*)

Definition at line 46 of file rijndael-api.c.

References rijndael_ctx::dk, rijndael_ctx::Nr, and rijndaelDecrypt().

Here is the call graph for this function:

7.21.1.3 void rijndael_encrypt (const rijndael_ctx * *ctx*, const u_char * *src*, u_char * *dst*)

Definition at line 53 of file rijndael-api.c.

References rijndael_ctx::ek, rijndael_ctx::Nr, and rijndaelEncrypt().

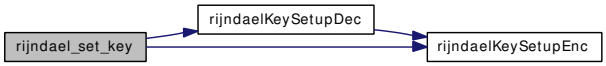
Here is the call graph for this function:

7.21.1.4 void rijndael_set_key (rijndael_ctx * *ctx*, const u_char * *key*, int *bits*)

Definition at line 38 of file rijndael-api.c.

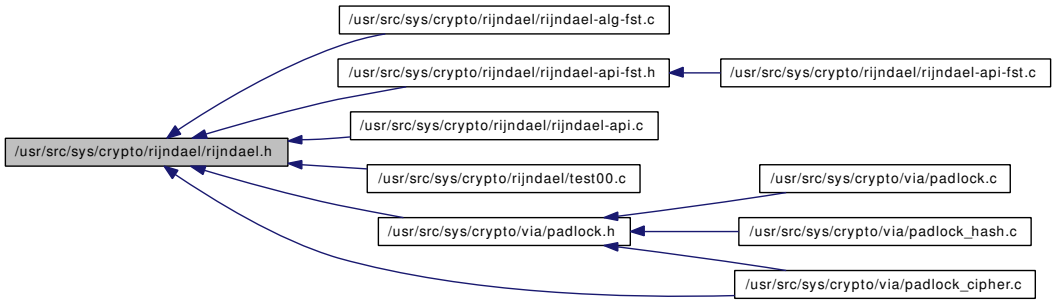
References rijndael_ctx::dk, rijndael_ctx::ek, rijndael_ctx::Nr, rijndaelKeySetupDec(), and rijndaelKeySetupEnc().

Here is the call graph for this function:



7.22 /usr/src/sys/crypto/rijndael/rijndael.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct rijndael_ctx

Defines

- #define RIJNDAEL_MAXKC (256/32)
- #define RIJNDAEL_MAXKB (256/8)
- #define RIJNDAEL_MAXNR 14

Functions

- void rijndael_set_key (rijndael_ctx *, const u_char *, int)
- void rijndael_decrypt (const rijndael_ctx *, const u_char *, u_char *)
- void rijndael_encrypt (const rijndael_ctx *, const u_char *, u_char *)
- int rijndaelKeySetupEnc (u_int32_t[ ], const u_int8_t[ ], int)
- int rijndaelKeySetupDec (u_int32_t[ ], const u_int8_t[ ], int)
- void rijndaelEncrypt (const u_int32_t[ ], int, const u_int8_t[16], u_int8_t[16])
- void rijndaelDecrypt (const u_int32_t[ ], int, const u_int8_t[16], u_int8_t[16])

7.22.1 Define Documentation

7.22.1.1 #define RIJNDAEL_MAXKB (256/8)

Definition at line 34 of file rijndael.h.

Referenced by rijndael_makeKey().

7.22.1.2 #define RIJNDAEL_MAXKC (256/32)

rijndael-alg-fst.h

Version:

3.0 (December 2000)

Optimised ANSI C code for the Rijndael cipher (now AES)

Author:

Vincent Rijmen <vincent.rijmen@esat.kuleuven.ac.be>
Antoon Bosselaers <antoon.bosselaers@esat.kuleuven.ac.be>
Paulo Barreto <paulo.barreto@terra.com.br>

Definition at line 33 of file rijndael.h.

7.22.1.3 #define RIJNDAEL_MAXNR 14

Definition at line 35 of file rijndael.h.

Referenced by padlock_cipher_key_setup().

7.22.2 Function Documentation

7.22.2.1 void rijndael_decrypt (const rijndael_ctx *, const u_char *, u_char *)

Definition at line 46 of file rijndael-api.c.

References rijndael_ctx::dk, rijndael_ctx::Nr, and rijndaelDecrypt().

Here is the call graph for this function:

7.22.2.2 void rijndael_encrypt (const rijndael_ctx *, const u_char *, u_char *)

Definition at line 53 of file rijndael-api.c.

References rijndael_ctx::ek, rijndael_ctx::Nr, and rijndaelEncrypt().

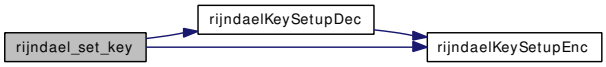
Here is the call graph for this function:

7.22.2.3 void rijndael_set_key (rijndael_ctx *, const u_char *, int)

Definition at line 38 of file rijndael-api.c.

References rijndael_ctx::dk, rijndael_ctx::ek, rijndael_ctx::Nr, rijndaelKeySetupDec(), and rijndaelKeySetupEnc().

Here is the call graph for this function:



7.22.2.4 void rijndaelDecrypt (const *u_int32_t*[ ], int, const *u_int8_t*[16], u_int8_t[16])

Definition at line 1044 of file rijndael-alg-fst.c.

References GETU32, PUTU32, Td0, Td1, Td2, Td3, and Td4.

Referenced by rijndael_blockDecrypt(), rijndael_decrypt(), and rijndael_padDecrypt().

7.22.2.5 void rijndaelEncrypt (const *u_int32_t*[ ], int, const *u_int8_t*[16], u_int8_t[16])

Definition at line 863 of file rijndael-alg-fst.c.

References GETU32, PUTU32, Te0, Te1, Te2, Te3, and Te4.

Referenced by rijndael_blockDecrypt(), rijndael_blockEncrypt(), rijndael_encrypt(), and rijndael_padEncrypt().

7.22.2.6 int rijndaelKeySetupDec (u32 *rk*[ ], const u8 *cipherKey*[ ], int *keyBits*)

Expand the cipher key into the decryption key schedule.

Returns:

the number of rounds for the given cipher key size.

Definition at line 823 of file rijndael-alg-fst.c.

References rijndaelKeySetupEnc(), Td0, Td1, Td2, Td3, and Te4.

Referenced by padlock_cipher_key_setup(), rijndael_makeKey(), and rijndael_set_key().

Here is the call graph for this function:

7.22.2.7 int rijndaelKeySetupEnc (u32 *rk*[ ], const u8 *cipherKey*[ ], int *keyBits*)

Expand the cipher key into the encryption key schedule.

Returns:

the number of rounds for the given cipher key size.

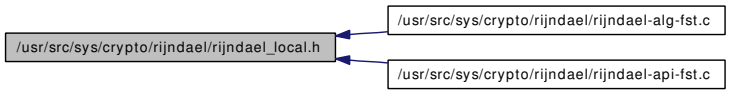
Definition at line 735 of file rijndael-alg-fst.c.

References GETU32, rcon, and Te4.

Referenced by padlock_cipher_key_setup(), rijndael_makeKey(), rijndael_set_key(), and rijndaelKeySetupDec().

7.23 /usr/src/sys/crypto/rijndael/rijndael_local.h File Reference

This graph shows which files directly or indirectly include this file:



Typedefs

- typedef u_int8_t u8
- typedef u_int16_t u16
- typedef u_int32_t u32

7.23.1 Typedef Documentation

7.23.1.1 typedef u_int16_t u16

Definition at line 6 of file rijndael_local.h.

7.23.1.2 typedef u_int32_t u32

Definition at line 7 of file rijndael_local.h.

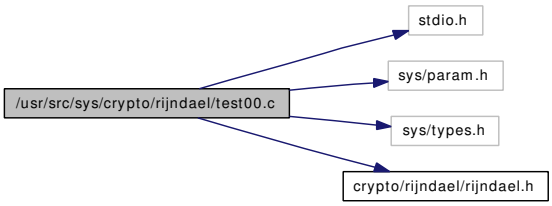
7.23.1.3 typedef u_int8_t u8

Definition at line 5 of file rijndael_local.h.

7.24 /usr/src/sys/crypto/rijndael/test00.c File Reference

```
#include <stdio.h>
#include <sys/param.h>
#include <sys/types.h>
#include <crypto/rijndael/rijndael.h>
```

Include dependency graph for test00.c:



Defines

- #define LL 32

Functions

- int main (int argc, char **argv)

7.24.1 Define Documentation

7.24.1.1 #define LL 32

Definition at line 39 of file test00.c.

Referenced by main().

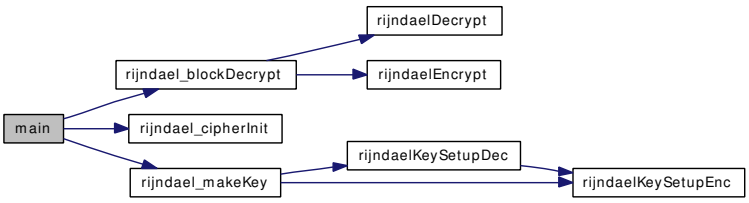
7.24.2 Function Documentation

7.24.2.1 int main (int *argc*, char ** *argv*)

Definition at line 41 of file test00.c.

References DIR_DECRYPT, LL, MODE_CBC, rijndael_blockDecrypt(), rijndael_cipherInit(), and rijndael_makeKey().

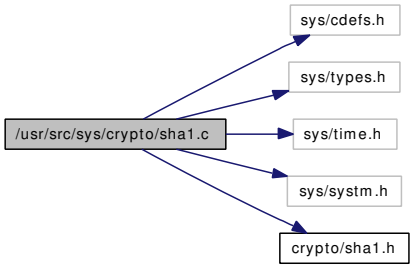
Here is the call graph for this function:


rijndaelDecrypt
rijndael_blockDecrypt
rijndaelEncrypt
main
rijndael_cipherInit
rijndaelKeySetupDec
rijndael_makeKey
rijndaelKeySetupEnc

7.25 /usr/src/sys/crypto/sha1.c File Reference

```
#include <sys/cdefs.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/systm.h>
#include <crypto/sha1.h>
```

Include dependency graph for sha1.c:



Defines

- #define K(t) _K[(t) / 20]
- #define F0(b, c, d) (((b) & (c)) | ((~(b)) & (d)))
- #define F1(b, c, d) (((b) ^ (c)) ^ (d))
- #define F2(b, c, d) (((b) & (c)) | ((b) & (d)) | ((c) & (d)))
- #define F3(b, c, d) (((b) ^ (c)) ^ (d))
- #define S(n, x) (((x) << (n)) | ((x) >> (32 - n)))
- #define H(n) (ctxt → h.b32[(n)])
- #define COUNT (ctxt → count)
- #define BCOUNT (ctxt → c.b64[0] / 8)
- #define W(n) (ctxt → m.b32[(n)])
- #define PUTBYTE(x)
- #define PUTPAD(x)

Functions

- __FBSDID ("$FreeBSD: src/sys/crypto/sha1.c,v 1.9 2003/06/10 21:36:57 obrien Exp $")
- static void sha1_step (struct sha1_ctxt *)
- void sha1_init (struct sha1_ctxt *ctxt)
- void sha1_pad (struct sha1_ctxt *ctxt)
- void sha1_loop (struct sha1_ctxt *ctxt, const u_int8_t *input, size_t len)
- void sha1_result (struct sha1_ctxt *ctxt, caddr_t digest0)

Variables

- static u_int32_t _K [ ] = { 0x5a827999, 0x6ed9eba1, 0x8f1bbcdc, 0xca62c1d6 }

7.25.1 Define Documentation

7.25.1.1 #define BCOUNT (ctxt → c.b64[0] / 8)

Definition at line 69 of file sha1.c.

7.25.1.2 #define COUNT (ctxt → count)

Definition at line 68 of file sha1.c.

Referenced by sha1_loop(), and sha1_pad().

7.25.1.3 #define F0(b, c, d) (((b) & (c)) | ((∼(b)) & (d)))

Definition at line 60 of file sha1.c.

Referenced by sha1_step().

7.25.1.4 #define F1(b, c, d) (((b) ^ (c)) ^ (d))

Definition at line 61 of file sha1.c.

Referenced by sha1_step().

7.25.1.5 #define F2(b, c, d) (((b) & (c)) | ((b) & (d)) | ((c) & (d)))

Definition at line 62 of file sha1.c.

Referenced by sha1_step().

7.25.1.6 #define F3(b, c, d) (((b) ^ (c)) ^ (d))

Definition at line 63 of file sha1.c.

Referenced by sha1_step().

7.25.1.7 #define H(n) (ctxt → h.b32[(n)])

Definition at line 67 of file sha1.c.

Referenced by sha1_init(), and sha1_step().

7.25.1.8 #define K(t) _K[(t) / 20]

Definition at line 58 of file sha1.c.

Referenced by sha1_step().

7.25.1.9 #define PUTBYTE(x)

Value:

```
{ \
        ctxt->m.b8[(COUNT % 64)] = (x);         \
        COUNT++;                                \
        COUNT %= 64;                            \
        ctxt->c.b64[0] += 8;                    \
        if (COUNT % 64 == 0)                    \
                sha1_step(ctxt);                \
     }
```

Definition at line 72 of file sha1.c.

7.25.1.10 #define PUTPAD(x)

Value:

```
{ \
        ctxt->m.b8[(COUNT % 64)] = (x);         \
        COUNT++;                                \
        COUNT %= 64;                            \
        if (COUNT % 64 == 0)                    \
                sha1_step(ctxt);                \
     }
```

Definition at line 81 of file sha1.c.

Referenced by sha1_pad().

7.25.1.11 #define S(n, x) (((x) << (n)) | ((x) >> (32 - n)))

Definition at line 65 of file sha1.c.

Referenced by sha1_step().

7.25.1.12 #define W(n) (ctxt → m.b32[(n)])

Definition at line 70 of file sha1.c.

Referenced by sha1_step().

7.25.2 Function Documentation

7.25.2.1 __FBSDID ("$FreeBSD: src/sys/crypto/sha1. *c*, v 1.9 2003/06/10 21:36:57 obrien Exp $")

7.25.2.2 void sha1_init (struct sha1_ctxt ∗ *ctxt*)

Definition at line 177 of file sha1.c.

References H.

7.25.2.3 void sha1_loop (struct sha1_ctxt ∗ *ctxt*, const u_int8_t ∗ *input*, size_t *len*)

Definition at line 224 of file sha1.c.

References COUNT, and sha1_step().

Here is the call graph for this function:

7.25.2.4 void sha1_pad (struct sha1_ctxt * *ctxt*)

Definition at line 189 of file sha1.c.

References COUNT, PUTPAD, and sha1_step().

Referenced by sha1_result().

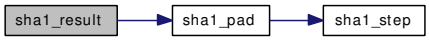
Here is the call graph for this function:

7.25.2.5 void sha1_result (struct sha1_ctxt * *ctxt*, caddr_t *digest0*)

Definition at line 252 of file sha1.c.

References sha1_pad().

Here is the call graph for this function:



7.25.2.6 static void sha1_step (struct sha1_ctxt *) [static]

Definition at line 92 of file sha1.c.

References sha1_ctxt::b8, F0, F1, F2, F3, H, K, sha1_ctxt::m, S, and W.

Referenced by sha1_loop(), and sha1_pad().

7.25.3 Variable Documentation

7.25.3.1 u_int32_t _K[ ] = { 0x5a827999, 0x6ed9eba1, 0x8f1bbcdc, 0xca62c1d6 } [static]

Definition at line 57 of file sha1.c.

7.26 /usr/src/sys/crypto/sha1.h File Reference

This graph shows which files directly or indirectly include this file:

Data Structures

- struct sha1_ctxt

Defines

- #define SHA1Init(x) sha1_init((x))
- #define SHA1Update(x, y, z) sha1_loop((x), (y), (z))
- #define SHA1Final(x, y) sha1_result((y), (x))
- #define SHA1_RESULTLEN (160/8)

Typedefs

- typedef sha1_ctxt SHA1_CTX

Functions

- void sha1_init (struct sha1_ctxt *)
- void sha1_pad (struct sha1_ctxt *)
- void sha1_loop (struct sha1_ctxt *, const u_int8_t *, size_t)
- void sha1_result (struct sha1_ctxt *, caddr_t)

7.26.1 Define Documentation

7.26.1.1 #define SHA1_RESULTLEN (160/8)

Definition at line 70 of file sha1.h.

7.26.1.2 #define SHA1Final(x, y) sha1_result((y), (x))

Definition at line 67 of file sha1.h.

7.26.1.3 #define SHA1Init(x) sha1_init((x))

Definition at line 65 of file sha1.h.

7.26.1.4 #define SHA1Update(x, y, z) sha1_loop((x), (y), (z))

Definition at line 66 of file sha1.h.

7.26.2 Typedef Documentation

7.26.2.1 typedef struct sha1_ctxt SHA1_CTX

Definition at line 64 of file sha1.h.

7.26.3 Function Documentation

7.26.3.1 void sha1_init (struct sha1_ctxt *)

Definition at line 177 of file sha1.c.

References H.

7.26.3.2 void sha1_loop (struct sha1_ctxt *, const u_int8_t *, size_t)

Definition at line 224 of file sha1.c.

References COUNT, and sha1_step().

Here is the call graph for this function:

7.26.3.3 void sha1_pad (struct sha1_ctxt *)

Definition at line 189 of file sha1.c.

References COUNT, PUTPAD, and sha1_step().

Referenced by sha1_result().

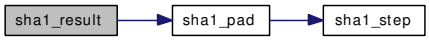
Here is the call graph for this function:

7.26.3.4 void sha1_result (struct sha1_ctxt *, caddr_t)

Definition at line 252 of file sha1.c.

References sha1_pad().

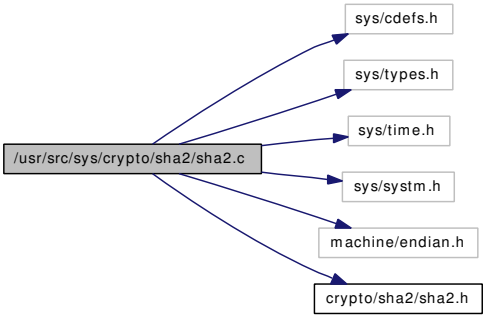
Here is the call graph for this function:



7.27 /usr/src/sys/crypto/sha2/sha2.c File Reference

```
#include <sys/cdefs.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/systm.h>
#include <machine/endian.h>
#include <crypto/sha2/sha2.h>
```

Include dependency graph for sha2.c:



Defines

- #define assert(x)
- #define SHA256_SHORT_BLOCK_LENGTH (SHA256_BLOCK_LENGTH - 8)
- #define SHA384_SHORT_BLOCK_LENGTH (SHA384_BLOCK_LENGTH - 16)
- #define SHA512_SHORT_BLOCK_LENGTH (SHA512_BLOCK_LENGTH - 16)
- #define REVERSE32(w, x)
- #define REVERSE64(w, x)
- #define ADDINC128(w, n)
- #define R(b, x) ((x) >> (b))
- #define S32(b, x) (((x) >> (b)) | ((x) << (32 - (b))))
- #define S64(b, x) (((x) >> (b)) | ((x) << (64 - (b))))
- #define Ch(x, y, z) (((x) & (y)) ^ ((~(x)) & (z)))
- #define Maj(x, y, z) (((x) & (y)) ^ ((x) & (z)) ^ ((y) & (z)))
- #define Sigma0_256(x) (S32(2, (x)) ^ S32(13, (x)) ^ S32(22, (x)))
- #define Sigma1_256(x) (S32(6, (x)) ^ S32(11, (x)) ^ S32(25, (x)))
- #define sigma0_256(x) (S32(7, (x)) ^ S32(18, (x)) ^ R(3 , (x)))
- #define sigma1_256(x) (S32(17, (x)) ^ S32(19, (x)) ^ R(10, (x)))
- #define Sigma0_512(x) (S64(28, (x)) ^ S64(34, (x)) ^ S64(39, (x)))
- #define Sigma1_512(x) (S64(14, (x)) ^ S64(18, (x)) ^ S64(41, (x)))
- #define sigma0_512(x) (S64(1, (x)) ^ S64(8, (x)) ^ R(7, (x)))
- #define sigma1_512(x) (S64(19, (x)) ^ S64(61, (x)) ^ R(6, (x)))

Typedefs

- typedef u_int8_t sha2_byte
- typedef u_int32_t sha2_word32
- typedef u_int64_t sha2_word64

Functions

- __FBSDID ("$FreeBSD: src/sys/crypto/sha2/sha2.c,v 1.9 2006/10/22 02:19:33 kevlo Exp $")
- void SHA512_Last (SHA512_CTX *)
- void SHA256_Transform (SHA256_CTX *, const sha2_word32 *)
- void SHA512_Transform (SHA512_CTX *, const sha2_word64 *)
- void SHA256_Init (SHA256_CTX *context)
- void SHA256_Update (SHA256_CTX *context, const sha2_byte *data, size_t len)
- void SHA256_Final (sha2_byte digest[ ], SHA256_CTX *context)
- char * SHA256_End (SHA256_CTX *context, char buffer[ ])
- char * SHA256_Data (const sha2_byte *data, size_t len, char digest[SHA256_DIGEST_STRING_-LENGTH])
- void SHA512_Init (SHA512_CTX *context)
- void SHA512_Update (SHA512_CTX *context, const sha2_byte *data, size_t len)
- void SHA512_Final (sha2_byte digest[ ], SHA512_CTX *context)
- char * SHA512_End (SHA512_CTX *context, char buffer[ ])
- char * SHA512_Data (const sha2_byte *data, size_t len, char digest[SHA512_DIGEST_STRING_-LENGTH])
- void SHA384_Init (SHA384_CTX *context)
- void SHA384_Update (SHA384_CTX *context, const sha2_byte *data, size_t len)
- void SHA384_Final (sha2_byte digest[ ], SHA384_CTX *context)
- char * SHA384_End (SHA384_CTX *context, char buffer[ ])
- char * SHA384_Data (const sha2_byte *data, size_t len, char digest[SHA384_DIGEST_STRING_-LENGTH])

Variables

- static const sha2_word32 K256 [64]
- static const sha2_word32 sha256_initial_hash_value [8]
- static const sha2_word64 K512 [80]
- static const sha2_word64 sha384_initial_hash_value [8]
- static const sha2_word64 sha512_initial_hash_value [8]
- static const char * sha2_hex_digits = "0123456789abcdef"

7.27.1 Define Documentation

7.27.1.1 #define ADDINC128(w, n)

Value:

```
{ \
        (w)[0] += (sha2_word64)(n); \
        if ((w)[0] < (n)) { \
                (w)[1]++; \
        } \
}
```

Definition at line 165 of file sha2.c.

Referenced by SHA512_Update().

7.27.1.2 #define assert(x)

Definition at line 71 of file sha2.c.

Referenced by SHA256_End(), SHA256_Final(), SHA256_Update(), SHA384_End(), SHA384_Final(), SHA512_End(), SHA512_Final(), and SHA512_Update().

7.27.1.3 #define Ch(x, y, z) (((x) & (y)) ^ ((~(x)) & (z)))

Definition at line 189 of file sha2.c.

Referenced by SHA256_Transform(), and SHA512_Transform().

7.27.1.4 #define Maj(x, y, z) (((x) & (y)) ^ ((x) & (z)) ^ ((y) & (z)))

Definition at line 190 of file sha2.c.

Referenced by SHA256_Transform(), and SHA512_Transform().

7.27.1.5 #define R(b, x) ((x) >> (b))

Definition at line 182 of file sha2.c.

7.27.1.6 #define REVERSE32(w, x)

Value:

```
{ \
        sha2_word32 tmp = (w); \
        tmp = (tmp >> 16) | (tmp << 16); \
        (x) = ((tmp & 0xff00ff00UL) >> 8) | ((tmp & 0x00ff00ffUL) << 8); \
}
```

Definition at line 145 of file sha2.c.

Referenced by SHA256_Final(), and SHA256_Transform().

7.27.1.7 #define REVERSE64(w, x)

Value:

```
{ \
        sha2_word64 tmp = (w); \
        tmp = (tmp >> 32) | (tmp << 32); \
        tmp = ((tmp & 0xff00ff00ff00ff00ULL) >> 8) | \
              ((tmp & 0x00ff00ff00ff00ffULL) << 8); \
        (x) = ((tmp & 0xffff0000ffff0000ULL) >> 16) | \
              ((tmp & 0x0000ffff0000ffffULL) << 16); \
}
```

Definition at line 150 of file sha2.c.

Referenced by SHA256_Final(), SHA384_Final(), SHA512_Final(), SHA512_Last(), and SHA512_-Transform().

7.27.1.8 #define S32(b, x) (((x) >> (b)) | ((x) << (32 - (b))))

Definition at line 184 of file sha2.c.

7.27.1.9 #define S64(b, x) (((x) >> (b)) | ((x) << (64 - (b))))

Definition at line 186 of file sha2.c.

7.27.1.10 #define SHA256_SHORT_BLOCK_LENGTH (SHA256_BLOCK_LENGTH - 8)

Definition at line 138 of file sha2.c.

Referenced by SHA256_Final().

7.27.1.11 #define SHA384_SHORT_BLOCK_LENGTH (SHA384_BLOCK_LENGTH - 16)

Definition at line 139 of file sha2.c.

7.27.1.12 #define SHA512_SHORT_BLOCK_LENGTH (SHA512_BLOCK_LENGTH - 16)

Definition at line 140 of file sha2.c.

Referenced by SHA512_Last().

7.27.1.13 #define sigma0_256(x) (S32(7, (x)) ^ S32(18, (x)) ^ R(3 , (x)))

Definition at line 195 of file sha2.c.

7.27.1.14 #define Sigma0_256(x) (S32(2, (x)) ^ S32(13, (x)) ^ S32(22, (x)))

Definition at line 193 of file sha2.c.

Referenced by SHA256_Transform().

7.27.1.15 #define sigma0_512(x) (S64(1, (x)) ^ S64(8, (x)) ^ R(7, (x)))

Definition at line 201 of file sha2.c.

7.27.1.16 #define Sigma0_512(x) (S64(28, (x)) ^ S64(34, (x)) ^ S64(39, (x)))

Definition at line 199 of file sha2.c.

Referenced by SHA512_Transform().

7.27.1.17 #define sigma1_256(x) (S32(17, (x)) ^ S32(19, (x)) ^ R(10, (x)))

Definition at line 196 of file sha2.c.

7.27.1.18 #define Sigma1_256(x) (S32(6, (x)) ^ S32(11, (x)) ^ S32(25, (x)))

Definition at line 194 of file sha2.c.

Referenced by SHA256_Transform().

7.27.1.19 #define sigma1_512(x) (S64(19, (x)) ^ S64(61, (x)) ^ R(6, (x)))

Definition at line 202 of file sha2.c.

7.27.1.20 #define Sigma1_512(x) (S64(14, (x)) ^ S64(18, (x)) ^ S64(41, (x)))

Definition at line 200 of file sha2.c.

Referenced by SHA512_Transform().

7.27.2 Typedef Documentation

7.27.2.1 typedef u_int8_t sha2_byte

Definition at line 129 of file sha2.c.

7.27.2.2 typedef u_int32_t sha2_word32

Definition at line 130 of file sha2.c.

7.27.2.3 typedef u_int64_t sha2_word64

Definition at line 131 of file sha2.c.

7.27.3 Function Documentation

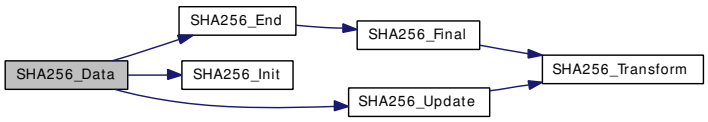
7.27.3.1 __FBSDID ("$FreeBSD: src/sys/crypto/sha2/sha2. *c*, v 1.9 2006/10/22 02:19:33 kevlo Exp $")

7.27.3.2 char* SHA256_Data (const sha2_byte * *data*, size_t *len*, char *digest*[SHA256_DIGEST_STRING_LENGTH])

Definition at line 641 of file sha2.c.

References SHA256_End(), SHA256_Init(), and SHA256_Update().

Here is the call graph for this function:



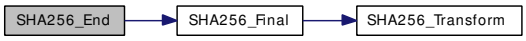
7.27.3.3 char* SHA256_End (SHA256_CTX * *context*, char *buffer*[ ])

Definition at line 618 of file sha2.c.

References assert, SHA256_DIGEST_LENGTH, SHA256_Final(), and sha2_hex_digits.

Referenced by SHA256_Data().

Here is the call graph for this function:



7.27.3.4 void SHA256_Final (sha2_byte *digest*[ ], SHA256_CTX * *context*)

Definition at line 555 of file sha2.c.

References assert, _SHA256_CTX::bitcount, _SHA256_CTX::buffer, REVERSE32, REVERSE64, SHA256_BLOCK_LENGTH, SHA256_DIGEST_LENGTH, SHA256_SHORT_BLOCK_LENGTH, SHA256_Transform(), and _SHA256_CTX::state.

Referenced by SHA256_End().

Here is the call graph for this function:

7.27.3.5 void SHA256_Init (SHA256_CTX * *context*)

Definition at line 323 of file sha2.c.

References _SHA256_CTX::bitcount, _SHA256_CTX::buffer, SHA256_BLOCK_LENGTH, SHA256_-DIGEST_LENGTH, sha256_initial_hash_value, and _SHA256_CTX::state.

Referenced by SHA256_Data().

7.27.3.6 void SHA256_Transform (SHA256_CTX *, const sha2_word32 *)

Definition at line 427 of file sha2.c.

References _SHA256_CTX::buffer, Ch, K256, Maj, REVERSE32, Sigma0_256, Sigma1_256, and _-SHA256_CTX::state.

Referenced by SHA256_Final(), and SHA256_Update().

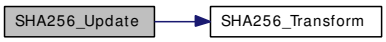
7.27.3.7 void SHA256_Update (SHA256_CTX * *context*, const sha2_byte * *data*, size_t *len*)

Definition at line 507 of file sha2.c.

References assert, _SHA256_CTX::bitcount, _SHA256_CTX::buffer, SHA256_BLOCK_LENGTH, and SHA256_Transform().

Referenced by SHA256_Data().

Here is the call graph for this function:

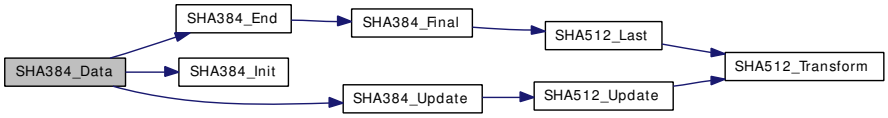


7.27.3.8 char* SHA384_Data (const sha2_byte * *data*, size_t *len*, char *digest*[SHA384_DIGEST_STRING_LENGTH])

Definition at line 1045 of file sha2.c.

References SHA384_End(), SHA384_Init(), and SHA384_Update().

Here is the call graph for this function:



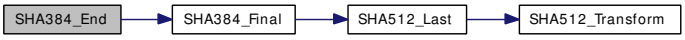
7.27.3.9 char* SHA384_End (SHA384_CTX * *context*, char *buffer*[ ])

Definition at line 1022 of file sha2.c.

References assert, sha2_hex_digits, SHA384_DIGEST_LENGTH, and SHA384_Final().

Referenced by SHA384_Data().

Here is the call graph for this function:



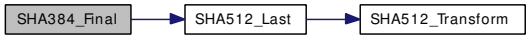
7.27.3.10 void SHA384_Final (sha2_byte *digest*[ ], SHA384_CTX * *context*)

Definition at line 993 of file sha2.c.

References assert, REVERSE64, SHA384_DIGEST_LENGTH, and SHA512_Last().

Referenced by SHA384_End().

Here is the call graph for this function:



7.27.3.11 void SHA384_Init (SHA384_CTX * *context*)

Definition at line 980 of file sha2.c.

References _SHA512_CTX::bitcount, _SHA512_CTX::buffer, SHA384_BLOCK_LENGTH, sha384_initial_hash_value, SHA512_DIGEST_LENGTH, and _SHA512_CTX::state.

Referenced by SHA384_Data().

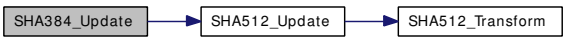
7.27.3.12 void SHA384_Update (SHA384_CTX * *context*, const sha2_byte * *data*, size_t *len*)

Definition at line 989 of file sha2.c.

References SHA512_Update().

Referenced by SHA384_Data().

Here is the call graph for this function:

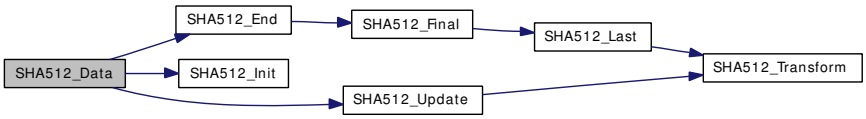


7.27.3.13 char* SHA512_Data (const sha2_byte * *data*, size_t *len*, char *digest*[SHA512_DIGEST_STRING_LENGTH])

Definition at line 970 of file sha2.c.

References SHA512_End(), SHA512_Init(), and SHA512_Update().

Here is the call graph for this function:



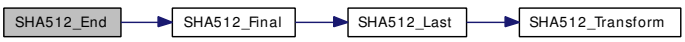
7.27.3.14 char* SHA512_End (SHA512_CTX * *context*, char *buffer*[ ])

Definition at line 947 of file sha2.c.

References assert, sha2_hex_digits, SHA512_DIGEST_LENGTH, and SHA512_Final().

Referenced by SHA512_Data().

Here is the call graph for this function:



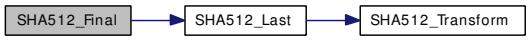
7.27.3.15 void SHA512_Final (sha2_byte *digest*[ ], SHA512_CTX * *context*)

Definition at line 918 of file sha2.c.

References assert, REVERSE64, SHA512_DIGEST_LENGTH, and SHA512_Last().

Referenced by SHA512_End().

Here is the call graph for this function:



7.27.3.16 void SHA512_Init (SHA512_CTX * *context*)

Definition at line 651 of file sha2.c.

References _SHA512_CTX::bitcount, _SHA512_CTX::buffer, SHA512_BLOCK_LENGTH, SHA512_-DIGEST_LENGTH, sha512_initial_hash_value, and _SHA512_CTX::state.

Referenced by SHA512_Data().

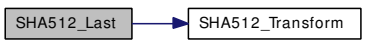
7.27.3.17 void SHA512_Last (SHA512_CTX *)

Definition at line 877 of file sha2.c.

References _SHA512_CTX::bitcount, _SHA512_CTX::buffer, REVERSE64, SHA512_BLOCK_-LENGTH, SHA512_SHORT_BLOCK_LENGTH, and SHA512_Transform().

Referenced by SHA384_Final(), and SHA512_Final().

Here is the call graph for this function:



7.27.3.18 void SHA512_Transform (SHA512_CTX *, const sha2_word64 *)

Definition at line 751 of file sha2.c.

References _SHA512_CTX::buffer, Ch, K512, Maj, REVERSE64, Sigma0_512, Sigma1_512, and _-SHA512_CTX::state.

Referenced by SHA512_Last(), and SHA512_Update().

7.27.3.19 void SHA512_Update (SHA512_CTX * *context*, const sha2_byte * *data*, size_t *len*)

Definition at line 829 of file sha2.c.

References ADDINC128, assert, _SHA512_CTX::bitcount, _SHA512_CTX::buffer, SHA512_BLOCK_-LENGTH, and SHA512_Transform().

Referenced by SHA384_Update(), and SHA512_Data().

Here is the call graph for this function:

7.27.4 Variable Documentation

7.27.4.1 const sha2_word32 K256[64] [static]

Initial value:

```
{
        0x428a2f98UL, 0x71374491UL, 0xb5c0fbcfUL, 0xe9b5dba5UL,
        0x3956c25bUL, 0x59f111f1UL, 0x923f82a4UL, 0xab1c5ed5UL,
        0xd807aa98UL, 0x12835b01UL, 0x243185beUL, 0x550c7dc3UL,
        0x72be5d74UL, 0x80deb1feUL, 0x9bdc06a7UL, 0xc19bf174UL,
        0xe49b69c1UL, 0xefbe4786UL, 0x0fc19dc6UL, 0x240ca1ccUL,
        0x2de92c6fUL, 0x4a7484aaUL, 0x5cb0a9dcUL, 0x76f988daUL,
        0x983e5152UL, 0xa831c66dUL, 0xb00327c8UL, 0xbf597fc7UL,
        0xc6e00bf3UL, 0xd5a79147UL, 0x06ca6351UL, 0x14292967UL,
        0x27b70a85UL, 0x2e1b2138UL, 0x4d2c6dfcUL, 0x53380d13UL,
        0x650a7354UL, 0x766a0abbUL, 0x81c2c92eUL, 0x92722c85UL,
        0xa2bfe8a1UL, 0xa81a664bUL, 0xc24b8b70UL, 0xc76c51a3UL,
        0xd192e819UL, 0xd6990624UL, 0xf40e3585UL, 0x106aa070UL,
        0x19a4c116UL, 0x1e376c08UL, 0x2748774cUL, 0x34b0bcb5UL,
        0x391c0cb3UL, 0x4ed8aa4aUL, 0x5b9cca4fUL, 0x682e6ff3UL,
        0x748f82eeUL, 0x78a5636fUL, 0x84c87814UL, 0x8cc70208UL,
        0x90befffaUL, 0xa4506cebUL, 0xbef9a3f7UL, 0xc67178f2UL
}
```

Definition at line 216 of file sha2.c.

Referenced by SHA256_Transform().

7.27.4.2 const sha2_word64 K512[80] [static]

Definition at line 248 of file sha2.c.

Referenced by SHA512_Transform().

7.27.4.3 const sha2_word32 sha256_initial_hash_value[8] [static]

Initial value:

```
{
        0x6a09e667UL,
        0xbb67ae85UL,
        0x3c6ef372UL,
        0xa54ff53aUL,
        0x510e527fUL,
        0x9b05688cUL,
        0x1f83d9abUL,
        0x5be0cd19UL
}
```

Definition at line 236 of file sha2.c.

Referenced by SHA256_Init().

7.27.4.4 const char* sha2_hex_digits = "0123456789abcdef" [static]

Definition at line 319 of file sha2.c.

Referenced by SHA256_End(), SHA384_End(), and SHA512_End().

7.27.4.5 const sha2_word64 sha384_initial_hash_value[8] [static]

Initial value:

```
{
        0xcbbb9d5dc1059ed8ULL,
        0x629a292a367cd507ULL,
        0x9159015a3070dd17ULL,
        0x152fecd8f70e5939ULL,
        0x67332667ffc00b31ULL,
        0x8eb44a8768581511ULL,
        0xdb0c2e0d64f98fa7ULL,
        0x47b5481dbefa4fa4ULL
}
```

Definition at line 292 of file sha2.c.

Referenced by SHA384_Init().

7.27.4.6 const sha2_word64 sha512_initial_hash_value[8] [static]

Initial value:

```
{
        0x6a09e667f3bcc908ULL,
        0xbb67ae8584caa73bULL,
        0x3c6ef372fe94f82bULL,
        0xa54ff53a5f1d36f1ULL,
        0x510e527fade682d1ULL,
        0x9b05688c2b3e6c1fULL,
        0x1f83d9abfb41bd6bULL,
        0x5be0cd19137e2179ULL
}
```

Definition at line 304 of file sha2.c.

Referenced by SHA512_Init().

7.28 /usr/src/sys/crypto/sha2/sha2.h File Reference

This graph shows which files directly or indirectly include this file:

Data Structures

- struct _SHA256_CTX
- struct _SHA512_CTX

Defines

- #define SHA256_BLOCK_LENGTH 64
- #define SHA256_DIGEST_LENGTH 32
- #define SHA256_DIGEST_STRING_LENGTH (SHA256_DIGEST_LENGTH * 2 + 1)
- #define SHA384_BLOCK_LENGTH 128
- #define SHA384_DIGEST_LENGTH 48
- #define SHA384_DIGEST_STRING_LENGTH (SHA384_DIGEST_LENGTH * 2 + 1)
- #define SHA512_BLOCK_LENGTH 128
- #define SHA512_DIGEST_LENGTH 64
- #define SHA512_DIGEST_STRING_LENGTH (SHA512_DIGEST_LENGTH * 2 + 1)

Typedefs

- typedef _SHA256_CTX SHA256_CTX
- typedef _SHA512_CTX SHA512_CTX
- typedef SHA512_CTX SHA384_CTX

Functions

- void SHA256_Init (SHA256_CTX *)
- void SHA256_Update (SHA256_CTX *, const u_int8_t *, size_t)
- void SHA256_Final (u_int8_t[SHA256_DIGEST_LENGTH], SHA256_CTX *)
- char * SHA256_End (SHA256_CTX *, char[SHA256_DIGEST_STRING_LENGTH])
- char * SHA256_Data (const u_int8_t *, size_t, char[SHA256_DIGEST_STRING_LENGTH])
- void SHA384_Init (SHA384_CTX *)
- void SHA384_Update (SHA384_CTX *, const u_int8_t *, size_t)
- void SHA384_Final (u_int8_t[SHA384_DIGEST_LENGTH], SHA384_CTX *)
- char * SHA384_End (SHA384_CTX *, char[SHA384_DIGEST_STRING_LENGTH])
- char * SHA384_Data (const u_int8_t *, size_t, char[SHA384_DIGEST_STRING_LENGTH])
- void SHA512_Init (SHA512_CTX *)
- void SHA512_Update (SHA512_CTX *, const u_int8_t *, size_t)
- void SHA512_Final (u_int8_t[SHA512_DIGEST_LENGTH], SHA512_CTX *)
- char * SHA512_End (SHA512_CTX *, char[SHA512_DIGEST_STRING_LENGTH])
- char * SHA512_Data (const u_int8_t *, size_t, char[SHA512_DIGEST_STRING_LENGTH])

7.28.1 Define Documentation

7.28.1.1 #define SHA256_BLOCK_LENGTH 64

Definition at line 48 of file sha2.h.

Referenced by SHA256_Final(), SHA256_Init(), and SHA256_Update().

7.28.1.2 #define SHA256_DIGEST_LENGTH 32

Definition at line 49 of file sha2.h.

Referenced by SHA256_End(), SHA256_Final(), and SHA256_Init().

7.28.1.3 #define SHA256_DIGEST_STRING_LENGTH (SHA256_DIGEST_LENGTH * 2 + 1)

Definition at line 50 of file sha2.h.

7.28.1.4 #define SHA384_BLOCK_LENGTH 128

Definition at line 51 of file sha2.h.

Referenced by SHA384_Init().

7.28.1.5 #define SHA384_DIGEST_LENGTH 48

Definition at line 52 of file sha2.h.

Referenced by SHA384_End(), and SHA384_Final().

7.28.1.6 #define SHA384_DIGEST_STRING_LENGTH (SHA384_DIGEST_LENGTH * 2 + 1)

Definition at line 53 of file sha2.h.

7.28.1.7 #define SHA512_BLOCK_LENGTH 128

Definition at line 54 of file sha2.h.

Referenced by SHA512_Init(), SHA512_Last(), and SHA512_Update().

7.28.1.8 #define SHA512_DIGEST_LENGTH 64

Definition at line 55 of file sha2.h.

Referenced by SHA384_Init(), SHA512_End(), SHA512_Final(), and SHA512_Init().

7.28.1.9 #define SHA512_DIGEST_STRING_LENGTH (SHA512_DIGEST_LENGTH * 2 + 1)

Definition at line 56 of file sha2.h.

7.28.2 Typedef Documentation

7.28.2.1 typedef struct _SHA256_CTX SHA256_CTX

7.28.2.2 typedef SHA512_CTX SHA384_CTX

Definition at line 113 of file sha2.h.

7.28.2.3 typedef struct _SHA512_CTX SHA512_CTX

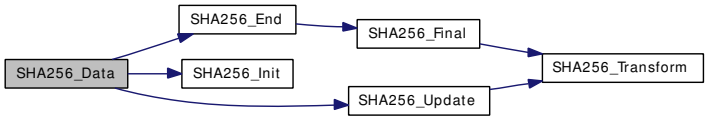
7.28.3 Function Documentation

7.28.3.1 char* SHA256_Data (const u_int8_t *, size_t, char[SHA256_DIGEST_STRING_LENGTH])

Definition at line 641 of file sha2.c.

References SHA256_End(), SHA256_Init(), and SHA256_Update().

Here is the call graph for this function:



7.28.3.2 char* SHA256_End (SHA256_CTX *, char[SHA256_DIGEST_STRING_LENGTH])

7.28.3.3 void SHA256_Final (u_int8_t[SHA256_DIGEST_LENGTH], SHA256_CTX *)

7.28.3.4 void SHA256_Init (SHA256_CTX *)

Definition at line 323 of file sha2.c.

References _SHA256_CTX::bitcount, _SHA256_CTX::buffer, SHA256_BLOCK_LENGTH, SHA256_DIGEST_LENGTH, sha256_initial_hash_value, and _SHA256_CTX::state.

Referenced by SHA256_Data().

7.28.3.5 void SHA256_Update (SHA256_CTX *, const u_int8_t *, size_t)

Definition at line 507 of file sha2.c.

References assert, _SHA256_CTX::bitcount, _SHA256_CTX::buffer, SHA256_BLOCK_LENGTH, and SHA256_Transform().

Referenced by SHA256_Data().

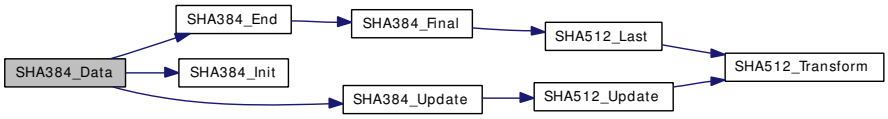
Here is the call graph for this function:

7.28.3.6 char* SHA384_Data (const u_int8_t *, size_t, char[SHA384_DIGEST_STRING_LENGTH])

Definition at line 1045 of file sha2.c.

References SHA384_End(), SHA384_Init(), and SHA384_Update().

Here is the call graph for this function:



7.28.3.7 char* SHA384_End (SHA384_CTX *, char[SHA384_DIGEST_STRING_LENGTH])

7.28.3.8 void SHA384_Final (u_int8_t[SHA384_DIGEST_LENGTH], SHA384_CTX *)

7.28.3.9 void SHA384_Init (SHA384_CTX *)

Definition at line 980 of file sha2.c.

References _SHA512_CTX::bitcount, _SHA512_CTX::buffer, SHA384_BLOCK_LENGTH, sha384_initial_hash_value, SHA512_DIGEST_LENGTH, and _SHA512_CTX::state.

Referenced by SHA384_Data().

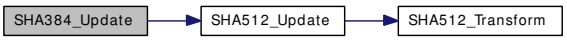
7.28.3.10 void SHA384_Update (SHA384_CTX *, const u_int8_t *, size_t)

Definition at line 989 of file sha2.c.

References SHA512_Update().

Referenced by SHA384_Data().

Here is the call graph for this function:

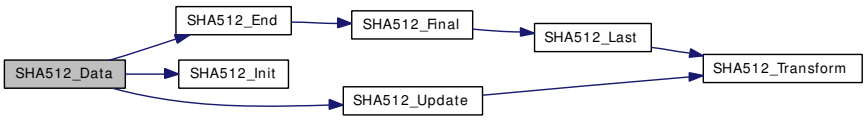


7.28.3.11 char* SHA512_Data (const u_int8_t *, size_t, char[SHA512_DIGEST_STRING_LENGTH])

Definition at line 970 of file sha2.c.

References SHA512_End(), SHA512_Init(), and SHA512_Update().

Here is the call graph for this function:



7.28.3.12 char* SHA512_End (SHA512_CTX *, char[SHA512_DIGEST_STRING_LENGTH])

7.28.3.13 void SHA512_Final (u_int8_t[SHA512_DIGEST_LENGTH], SHA512_CTX *)

7.28.3.14 void SHA512_Init (SHA512_CTX *)

Definition at line 651 of file sha2.c.

References _SHA512_CTX::bitcount, _SHA512_CTX::buffer, SHA512_BLOCK_LENGTH, SHA512_-DIGEST_LENGTH, sha512_initial_hash_value, and _SHA512_CTX::state.

Referenced by SHA512_Data().

7.28.3.15 void SHA512_Update (SHA512_CTX *, const u_int8_t *, size_t)

Definition at line 829 of file sha2.c.

References ADDINC128, assert, _SHA512_CTX::bitcount, _SHA512_CTX::buffer, SHA512_BLOCK_-LENGTH, and SHA512_Transform().

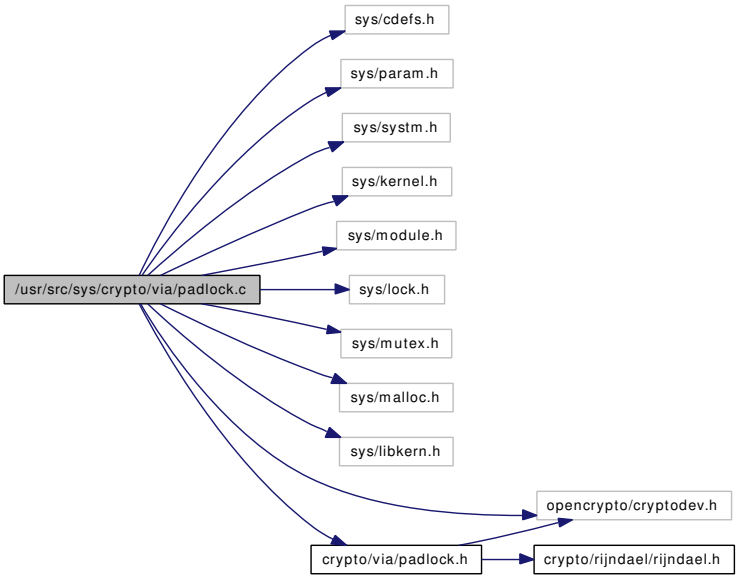
Referenced by SHA384_Update(), and SHA512_Data().

Here is the call graph for this function:

7.29 /usr/src/sys/crypto/via/padlock.c File Reference

```
#include <sys/cdefs.h>
#include <sys/param.h>
#include <sys/systm.h>
#include <sys/kernel.h>
#include <sys/module.h>
#include <sys/lock.h>
#include <sys/mutex.h>
#include <sys/malloc.h>
#include <sys/libkern.h>
#include <opencrypto/cryptodev.h>
#include <crypto/via/padlock.h>
```

Include dependency graph for padlock.c:



Data Structures

- struct padlock_softc

Functions

- __FBSDID ("$FreeBSD: src/sys/crypto/via/padlock.c,v 1.11 2006/07/28 14:48:30 pjd Exp $")
- static int padlock_newsession (void *arg __unused, uint32_t *sidp, struct cryptoini *cri)
- static int padlock_freesession (void *arg __unused, uint64_t tid)
- static int padlock_process (void *arg __unused, struct cryptop *crp, int hint __unused)
- MALLOC_DEFINE (M_PADLOCK,"padlock_data","PadLock Data")

- static int padlock_init (void)
- static int padlock_destroy (void)
- static int padlock_modevent (module_t mod, int type, void *unused __unused)
- DECLARE_MODULE (padlock, padlock_mod, SI_SUB_DRIVERS, SI_ORDER_ANY)
- MODULE_VERSION (padlock, 1)
- MODULE_DEPEND (padlock, crypto, 1, 1, 1)

Variables

- static struct padlock_softc * padlock_sc
- static moduledata_t padlock_mod

7.29.1 Function Documentation

7.29.1.1 __FBSDID ("$FreeBSD: src/sys/crypto/via/padlock. *c*, v 1.11 2006/07/28 14:48:30 pjd Exp $")

7.29.1.2 DECLARE_MODULE (padlock, padlock_mod, SI_SUB_DRIVERS, SI_ORDER_ANY)

7.29.1.3 MALLOC_DEFINE (M_PADLOCK, "padlock_data", "PadLock Data")

7.29.1.4 MODULE_DEPEND (padlock, crypto, 1, 1, 1)

7.29.1.5 MODULE_VERSION (padlock, 1)

7.29.1.6 static int padlock_destroy (void) [static]

Definition at line 137 of file padlock.c.

References padlock_sc, padlock_softc::sc_cid, and padlock_session::ses_used.

Referenced by padlock_modevent().

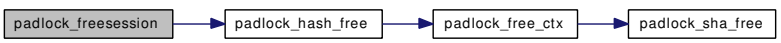
7.29.1.7 static int padlock_freesession (void *arg __*unused*, uint64_t *tid*) [static]

Definition at line 258 of file padlock.c.

References padlock_hash_free(), padlock_sc, and padlock_session::ses_id.

Referenced by padlock_init().

Here is the call graph for this function:



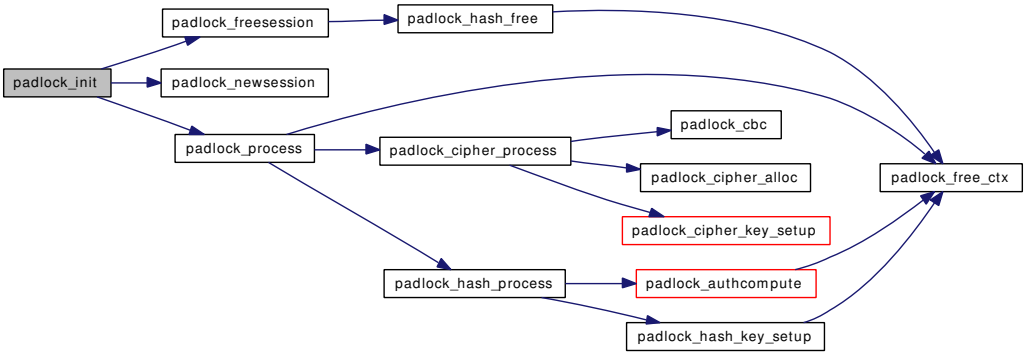
7.29.1.8 static int padlock_init (void) [static]

Definition at line 73 of file padlock.c.

References padlock_freesession(), padlock_newsession(), padlock_process(), padlock_softc::sc_cid, and padlock_softc::sc_sid.

Referenced by padlock_modevent().

Here is the call graph for this function:



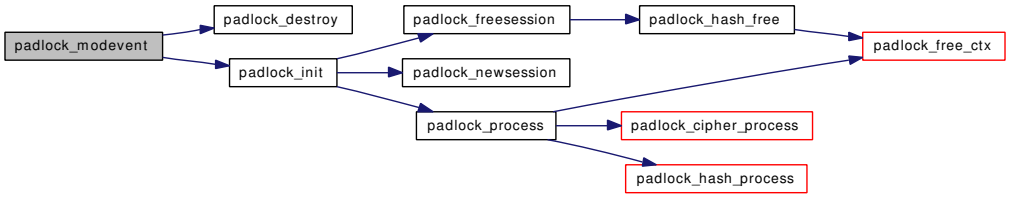
7.29.1.9 static int padlock_modevent (module_t *mod*, int *type*, void *unused *__unused*)

[static]

Definition at line 377 of file padlock.c.

References padlock_destroy(), and padlock_init().

Here is the call graph for this function:



7.29.1.10 static int padlock_newsession (void *arg *__unused*, uint32_t * *sidp*, struct cryptoini * *cri*)

[static]

Definition at line 169 of file padlock.c.

References padlock_sc.

Referenced by padlock_init().

7.29.1.11 static int padlock_process (void *arg *__unused*, struct cryptop * *crp*, int hint *__unused*)

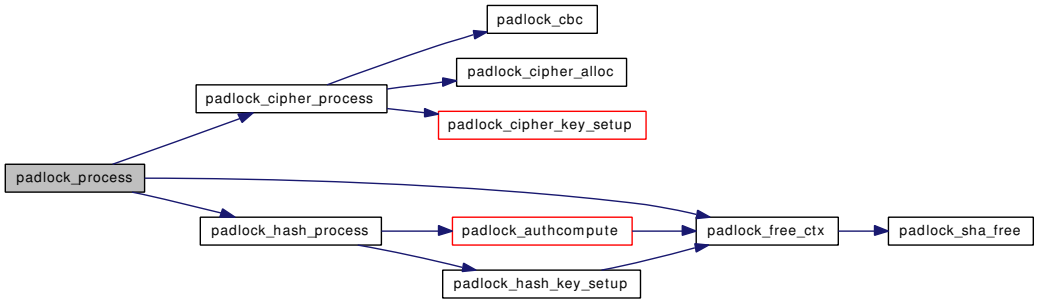
[static]

Definition at line 285 of file padlock.c.

References padlock_cipher_process(), padlock_free_ctx(), padlock_hash_process(), padlock_sc, padlock_session::ses_axf, padlock_session::ses_ictx, padlock_session::ses_id, and padlock_session::ses_octx.

Referenced by padlock_init().

Here is the call graph for this function:



7.29.2 Variable Documentation

7.29.2.1 moduledata_t padlock_mod [static]

Initial value:

```
{
        "padlock",
        padlock_modevent,
        0
}
```

Definition at line 393 of file padlock.c.

7.29.2.2 struct padlock_softc* padlock_sc [static]

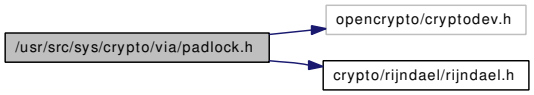
Definition at line 62 of file padlock.c.

Referenced by padlock_destroy(), padlock_freesession(), padlock_newsession(), and padlock_process().

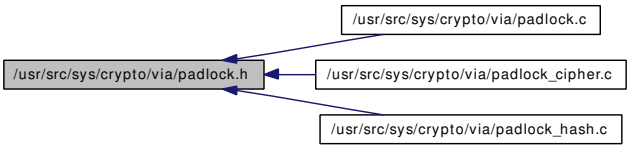
7.30 /usr/src/sys/crypto/via/padlock.h File Reference

```
#include <opencrypto/cryptodev.h>
#include <crypto/rijndael/rijndael.h>
```

Include dependency graph for padlock.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- union padlock_cw
- struct padlock_session

Defines

- #define cw_round_count __field.round_count
- #define cw_algorithm_type __field.algorithm_type
- #define cw_key_generation __field.key_generation
- #define cw_intermediate __field.intermediate
- #define cw_direction __field.direction
- #define cw_key_size __field.key_size
- #define cw_filler0 __field.filler0
- #define cw_filler1 __field.filler1
- #define cw_filler2 __field.filler2
- #define cw_filler3 __field.filler3
- #define PADLOCK_ALIGN(p) (void *)(roundup2((uintptr_t)(p), 16))

Functions

- int padlock_cipher_setup (struct padlock_session *ses, struct cryptoini *encini)
- int padlock_cipher_process (struct padlock_session *ses, struct cryptodesc *enccrd, struct cryptop *crp)
- int padlock_hash_setup (struct padlock_session *ses, struct cryptoini *macini)
- int padlock_hash_process (struct padlock_session *ses, struct cryptodesc *maccrd, struct cryptop *crp)
- void padlock_hash_free (struct padlock_session *ses)

7.30.1 Define Documentation

7.30.1.1 #define cw_algorithm_type __field.algorithm_type

Definition at line 51 of file padlock.h.

7.30.1.2 #define cw_direction __field.direction

Definition at line 54 of file padlock.h.

7.30.1.3 #define cw_filler0 __field.filler0

Definition at line 56 of file padlock.h.

7.30.1.4 #define cw_filler1 __field.filler1

Definition at line 57 of file padlock.h.

7.30.1.5 #define cw_filler2 __field.filler2

Definition at line 58 of file padlock.h.

7.30.1.6 #define cw_filler3 __field.filler3

Definition at line 59 of file padlock.h.

7.30.1.7 #define cw_intermediate __field.intermediate

Definition at line 53 of file padlock.h.

7.30.1.8 #define cw_key_generation __field.key_generation

Definition at line 52 of file padlock.h.

7.30.1.9 #define cw_key_size __field.key_size

Definition at line 55 of file padlock.h.

7.30.1.10 #define cw_round_count __field.round_count

Definition at line 50 of file padlock.h.

7.30.1.11 #define PADLOCK_ALIGN(p) (void *)(roundup2((uintptr_t)(p), 16))

Definition at line 75 of file padlock.h.

Referenced by padlock_cipher_process(), padlock_do_sha1(), and padlock_do_sha256().

7.30.2 Function Documentation

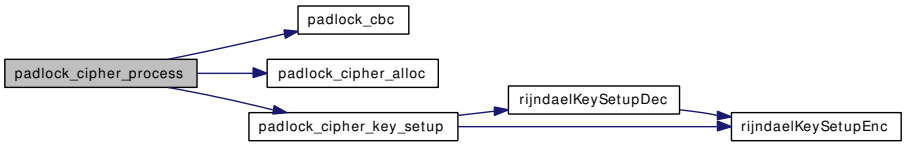
7.30.2.1 int padlock_cipher_process (struct padlock_session * *ses*, struct cryptodesc * *enccrd*, struct cryptop * *crp*)

Definition at line 200 of file padlock_cipher.c.

References PADLOCK_ALIGN, padlock_cbc(), padlock_cipher_alloc(), padlock_cipher_key_setup(), PADLOCK_DIRECTION_DECRYPT, and PADLOCK_DIRECTION_ENCRYPT.

Referenced by padlock_process().

Here is the call graph for this function:

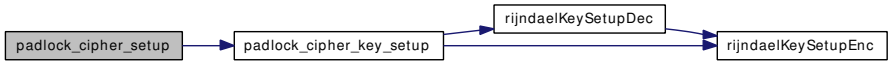


7.30.2.2 int padlock_cipher_setup (struct padlock_session * *ses*, struct cryptoini * *encini*)

Definition at line 121 of file padlock_cipher.c.

References PADLOCK_ALGORITHM_TYPE_AES, padlock_cipher_key_setup(), PADLOCK_KEY_GENERATION_HW, PADLOCK_KEY_GENERATION_SW, PADLOCK_KEY_SIZE_128, PADLOCK_KEY_SIZE_192, PADLOCK_KEY_SIZE_256, PADLOCK_ROUND_COUNT_AES128, PADLOCK_ROUND_COUNT_AES192, and PADLOCK_ROUND_COUNT_AES256.

Here is the call graph for this function:



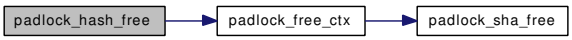
7.30.2.3 void padlock_hash_free (struct padlock_session * *ses*)

Definition at line 374 of file padlock_hash.c.

References padlock_free_ctx(), padlock_session::ses_axf, padlock_session::ses_ictx, and padlock_session::ses_octx.

Referenced by padlock_freesession().

Here is the call graph for this function:



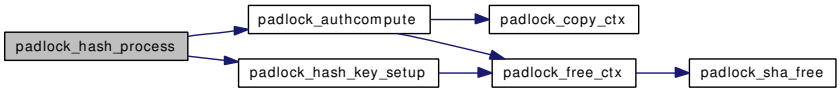
7.30.2.4 int padlock_hash_process (struct padlock_session * *ses*, struct cryptodesc * *maccrd*, struct cryptop * *crp*)

Definition at line 361 of file padlock_hash.c.

References padlock_authcompute(), and padlock_hash_key_setup().

Referenced by padlock_process().

Here is the call graph for this function:

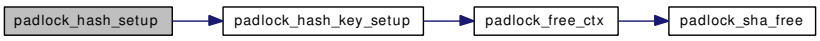


7.30.2.5 int padlock_hash_setup (struct padlock_session * *ses*, struct cryptoini * *macini*)

Definition at line 308 of file padlock_hash.c.

References padlock_hash_key_setup(), padlock_session::ses_axf, padlock_session::ses_ictx, padlock_session::ses_mlen, and padlock_session::ses_octx.

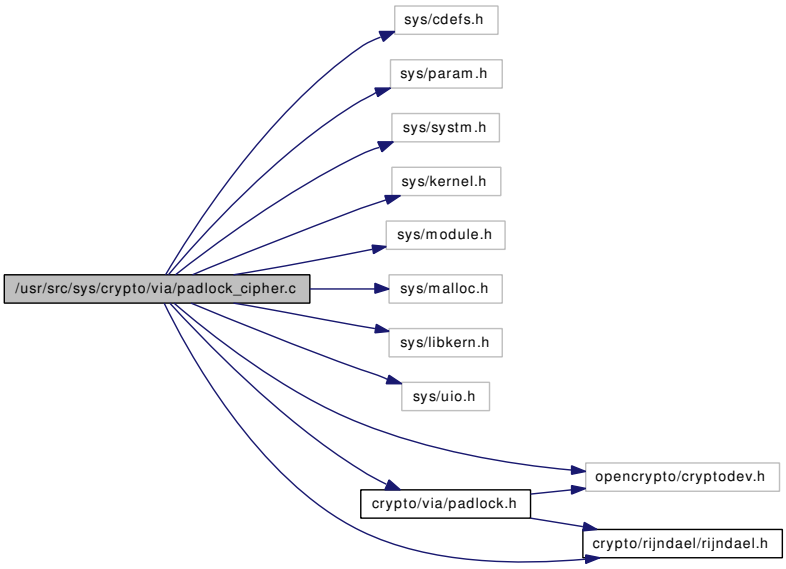
Here is the call graph for this function:



7.31 /usr/src/sys/crypto/via/padlock_cipher.c File Reference

```
#include <sys/cdefs.h>
#include <sys/param.h>
#include <sys/systm.h>
#include <sys/kernel.h>
#include <sys/module.h>
#include <sys/malloc.h>
#include <sys/libkern.h>
#include <sys/uio.h>
#include <opencrypto/cryptodev.h>
#include <crypto/rijndael/rijndael.h>
#include <crypto/via/padlock.h>
```

Include dependency graph for padlock_cipher.c:



Defines

- #define PADLOCK_ROUND_COUNT_AES128 10
- #define PADLOCK_ROUND_COUNT_AES192 12
- #define PADLOCK_ROUND_COUNT_AES256 14
- #define PADLOCK_ALGORITHM_TYPE_AES 0
- #define PADLOCK_KEY_GENERATION_HW 0
- #define PADLOCK_KEY_GENERATION_SW 1
- #define PADLOCK_DIRECTION_ENCRYPT 0
- #define PADLOCK_DIRECTION_DECRYPT 1
- #define PADLOCK_KEY_SIZE_128 0
- #define PADLOCK_KEY_SIZE_192 1

- #define PADLOCK_KEY_SIZE_256 2

Functions

- __FBSDID ("$FreeBSD: src/sys/crypto/via/padlock_cipher.c,v 1.5 2006/09/15 10:44:55 pjd Exp $")
- MALLOC_DECLARE (M_PADLOCK)
- static __inline void padlock_cbc (void *in, void *out, size_t count, void *key, union padlock_cw *cw, void *iv)
- static void padlock_cipher_key_setup (struct padlock_session *ses, caddr_t key, int klen)
- int padlock_cipher_setup (struct padlock_session *ses, struct cryptoini *encini)
- static u_char * padlock_cipher_alloc (struct cryptodesc *enccrd, struct cryptop *crp, int *allocated)
- int padlock_cipher_process (struct padlock_session *ses, struct cryptodesc *enccrd, struct cryptop *crp)

7.31.1 Define Documentation

7.31.1.1 #define PADLOCK_ALGORITHM_TYPE_AES 0

Definition at line 67 of file padlock_cipher.c.

Referenced by padlock_cipher_setup().

7.31.1.2 #define PADLOCK_DIRECTION_DECRYPT 1

Definition at line 73 of file padlock_cipher.c.

Referenced by padlock_cipher_process().

7.31.1.3 #define PADLOCK_DIRECTION_ENCRYPT 0

Definition at line 72 of file padlock_cipher.c.

Referenced by padlock_cipher_process().

7.31.1.4 #define PADLOCK_KEY_GENERATION_HW 0

Definition at line 69 of file padlock_cipher.c.

Referenced by padlock_cipher_setup().

7.31.1.5 #define PADLOCK_KEY_GENERATION_SW 1

Definition at line 70 of file padlock_cipher.c.

Referenced by padlock_cipher_key_setup(), and padlock_cipher_setup().

7.31.1.6 #define PADLOCK_KEY_SIZE_128 0

Definition at line 75 of file padlock_cipher.c.

Referenced by padlock_cipher_setup().

7.31.1.7 #define PADLOCK_KEY_SIZE_192 1

Definition at line 76 of file padlock_cipher.c.

Referenced by padlock_cipher_setup().

7.31.1.8 #define PADLOCK_KEY_SIZE_256 2

Definition at line 77 of file padlock_cipher.c.

Referenced by padlock_cipher_setup().

7.31.1.9 #define PADLOCK_ROUND_COUNT_AES128 10

Definition at line 63 of file padlock_cipher.c.

Referenced by padlock_cipher_setup().

7.31.1.10 #define PADLOCK_ROUND_COUNT_AES192 12

Definition at line 64 of file padlock_cipher.c.

Referenced by padlock_cipher_setup().

7.31.1.11 #define PADLOCK_ROUND_COUNT_AES256 14

Definition at line 65 of file padlock_cipher.c.

Referenced by padlock_cipher_setup().

7.31.2 Function Documentation

7.31.2.1 __FBSDID ("$FreeBSD: src/sys/crypto/via/padlock_cipher. *c*, v 1.5 2006/09/15 10:44:55 pjd Exp $")

7.31.2.2 MALLOC_DECLARE (M_PADLOCK)

7.31.2.3 static __inline void padlock_cbc (void * *in*, void * *out*, size_t *count*, void * *key*, union padlock_cw * *cw*, void * *iv*) [static]

Definition at line 82 of file padlock_cipher.c.

Referenced by padlock_cipher_process().

7.31.2.4 static u_char* padlock_cipher_alloc (struct cryptodesc * *enccrd*, struct cryptop * *crp*, int * *allocated*) [static]

Definition at line 168 of file padlock_cipher.c.

Referenced by padlock_cipher_process().

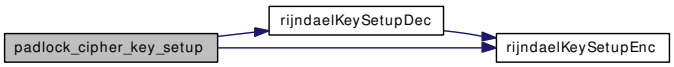
7.31.2.5 static void padlock_cipher_key_setup (struct padlock_session * *ses*, caddr_t *key*, int *klen*) [static]

Definition at line 100 of file padlock_cipher.c.

References PADLOCK_KEY_GENERATION_SW, RIJNDAEL_MAXNR, rijndaelKeySetupDec(), and rijndaelKeySetupEnc().

Referenced by padlock_cipher_process(), and padlock_cipher_setup().

Here is the call graph for this function:



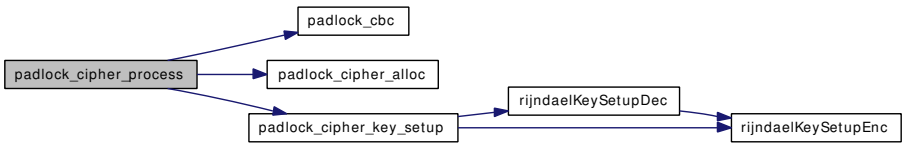
7.31.2.6 int padlock_cipher_process (struct padlock_session * *ses*, struct cryptodesc * *enccrd*, struct cryptop * *crp*)

Definition at line 200 of file padlock_cipher.c.

References PADLOCK_ALIGN, padlock_cbc(), padlock_cipher_alloc(), padlock_cipher_key_setup(), PADLOCK_DIRECTION_DECRYPT, and PADLOCK_DIRECTION_ENCRYPT.

Referenced by padlock_process().

Here is the call graph for this function:

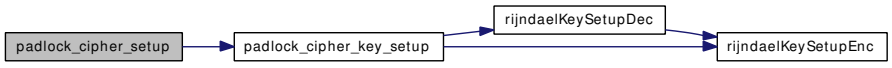


7.31.2.7 int padlock_cipher_setup (struct padlock_session * *ses*, struct cryptoini * *encini*)

Definition at line 121 of file padlock_cipher.c.

References PADLOCK_ALGORITHM_TYPE_AES, padlock_cipher_key_setup(), PADLOCK_KEY_GENERATION_HW, PADLOCK_KEY_GENERATION_SW, PADLOCK_KEY_SIZE_128, PADLOCK_KEY_SIZE_192, PADLOCK_KEY_SIZE_256, PADLOCK_ROUND_COUNT_AES128, PADLOCK_ROUND_COUNT_AES192, and PADLOCK_ROUND_COUNT_AES256.

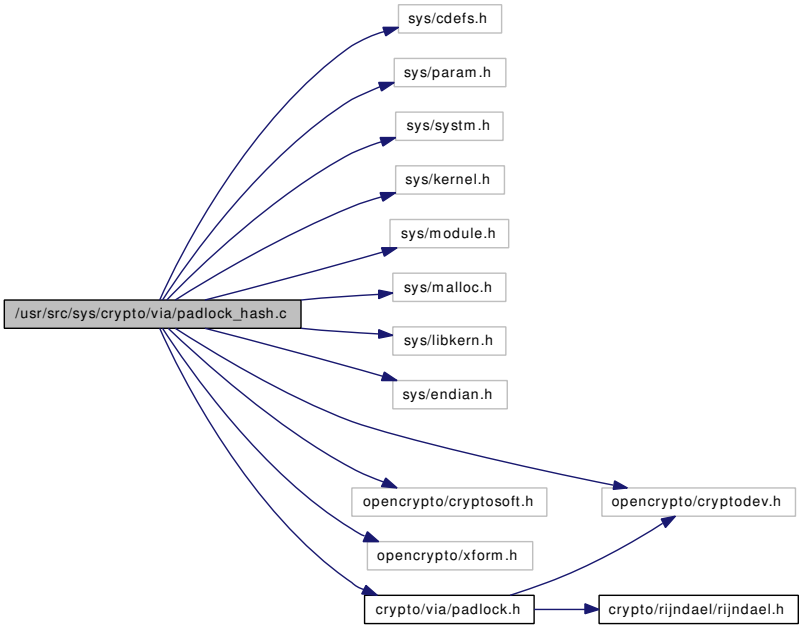
Here is the call graph for this function:



7.32 /usr/src/sys/crypto/via/padlock_hash.c File Reference

```
#include <sys/cdefs.h>
#include <sys/param.h>
#include <sys/systm.h>
#include <sys/kernel.h>
#include <sys/module.h>
#include <sys/malloc.h>
#include <sys/libkern.h>
#include <sys/endian.h>
#include <opencrypto/cryptodev.h>
#include <opencrypto/cryptosoft.h>
#include <opencrypto/xform.h>
#include <crypto/via/padlock.h>
```

Include dependency graph for padlock_hash.c:



Data Structures

- struct padlock_sha_ctx

Functions

- __FBSDID ("$FreeBSD: src/sys/crypto/via/padlock_hash.c,v 1.2 2006/07/25 19:04:26 pjd Exp $")
- CTASSERT (sizeof(struct padlock_sha_ctx)<=sizeof(union authctx))

- static void padlock_sha_init (struct padlock_sha_ctx *ctx)
- static int padlock_sha_update (struct padlock_sha_ctx *ctx, uint8_t *buf, uint16_t bufsize)
- static void padlock_sha1_final (uint8_t *hash, struct padlock_sha_ctx *ctx)
- static void padlock_sha256_final (uint8_t *hash, struct padlock_sha_ctx *ctx)
- MALLOC_DECLARE (M_PADLOCK)
- static __inline void padlock_output_block (uint32_t *src, uint32_t *dst, size_t count)
- static void padlock_do_sha1 (const u_char *in, u_char *out, int count)
- static void padlock_do_sha256 (const char *in, char *out, int count)
- static void padlock_sha_free (struct padlock_sha_ctx *ctx)
- static void padlock_copy_ctx (struct auth_hash *axf, void *sctx, void *dctx)
- static void padlock_free_ctx (struct auth_hash *axf, void *ctx)
- static void padlock_hash_key_setup (struct padlock_session *ses, caddr_t key, int klen)
- static int padlock_authcompute (struct padlock_session *ses, struct cryptodesc *crd, caddr_t buf, int flags)
- int padlock_hash_setup (struct padlock_session *ses, struct cryptoini *macini)
- int padlock_hash_process (struct padlock_session *ses, struct cryptodesc *maccrd, struct cryptop *crp)
- void padlock_hash_free (struct padlock_session *ses)

Variables

- static struct auth_hash padlock_hmac_sha1
- static struct auth_hash padlock_hmac_sha256

7.32.1 Function Documentation

7.32.1.1 __FBSDID ("$FreeBSD: src/sys/crypto/via/padlock_hash. *c*, v 1.2 2006/07/25 19:04:26 pjd Exp $")

7.32.1.2 CTASSERT (sizeof(struct padlock_sha_ctx)<=sizeof(union authctx))

7.32.1.3 MALLOC_DECLARE (M_PADLOCK)

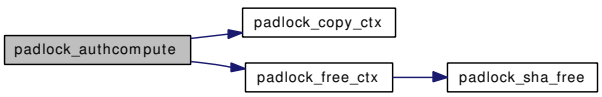
7.32.1.4 static int padlock_authcompute (struct padlock_session * *ses*, struct cryptodesc * *crd*, caddr_t *buf*, int *flags*) [static]

Definition at line 278 of file padlock_hash.c.

References padlock_copy_ctx(), padlock_free_ctx(), padlock_session::ses_axf, padlock_session::ses_ictx, padlock_session::ses_mlen, and padlock_session::ses_octx.

Referenced by padlock_hash_process().

Here is the call graph for this function:



7.32.1.5 static void padlock_copy_ctx (struct auth_hash * *axf*, void * *sctx*, void * *dctx*) [static]

Definition at line 211 of file padlock_hash.c.

References padlock_sha_ctx::psc_buf, padlock_sha_ctx::psc_offset, and padlock_sha_ctx::psc_size.

Referenced by padlock_authcompute().

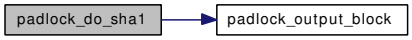
7.32.1.6 static void padlock_do_sha1 (const u_char * *in*, u_char * *out*, int *count*) [static]

Definition at line 108 of file padlock_hash.c.

References PADLOCK_ALIGN, and padlock_output_block().

Referenced by padlock_sha1_final().

Here is the call graph for this function:



7.32.1.7 static void padlock_do_sha256 (const char * *in*, char * *out*, int *count*) [static]

Definition at line 132 of file padlock_hash.c.

References PADLOCK_ALIGN, and padlock_output_block().

Referenced by padlock_sha256_final().

Here is the call graph for this function:

7.32.1.8 static void padlock_free_ctx (struct auth_hash * *axf*, void * *ctx*) [static]

Definition at line 229 of file padlock_hash.c.

References padlock_sha_free().

Referenced by padlock_authcompute(), padlock_hash_free(), padlock_hash_key_setup(), and padlock_process().

Here is the call graph for this function:

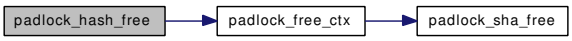
7.32.1.9 void padlock_hash_free (struct padlock_session * *ses*)

Definition at line 374 of file padlock_hash.c.

References padlock_free_ctx(), padlock_session::ses_axf, padlock_session::ses_ictx, and padlock_-session::ses_octx.

Referenced by padlock_freesession().

Here is the call graph for this function:



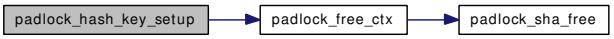
7.32.1.10 static void padlock_hash_key_setup (struct padlock_session * *ses*, caddr_t *key*, int *klen*) [static]

Definition at line 240 of file padlock_hash.c.

References padlock_free_ctx(), padlock_session::ses_axf, padlock_session::ses_ictx, and padlock_-session::ses_octx.

Referenced by padlock_hash_process(), and padlock_hash_setup().

Here is the call graph for this function:



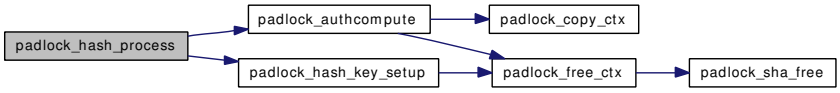
7.32.1.11 int padlock_hash_process (struct padlock_session * *ses*, struct cryptodesc * *maccrd*, struct cryptop * *crp*)

Definition at line 361 of file padlock_hash.c.

References padlock_authcompute(), and padlock_hash_key_setup().

Referenced by padlock_process().

Here is the call graph for this function:

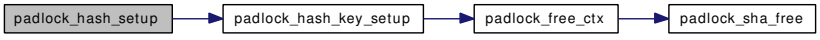


7.32.1.12 int padlock_hash_setup (struct padlock_session * *ses*, struct cryptoini * *macini*)

Definition at line 308 of file padlock_hash.c.

References padlock_hash_key_setup(), padlock_session::ses_axf, padlock_session::ses_ictx, padlock_-session::ses_mlen, and padlock_session::ses_octx.

Here is the call graph for this function:



7.32.1.13 static __inline void padlock_output_block (uint32_t * *src*, uint32_t * *dst*, size_t *count*) [static]

Definition at line 100 of file padlock_hash.c.

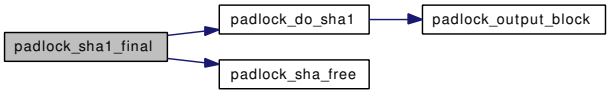
Referenced by padlock_do_sha1(), and padlock_do_sha256().

7.32.1.14 static void padlock_sha1_final (uint8_t * *hash*, struct padlock_sha_ctx * *ctx*) [static]

Definition at line 195 of file padlock_hash.c.

References padlock_do_sha1(), padlock_sha_free(), padlock_sha_ctx::psc_buf, and padlock_sha_ctx::psc_offset.

Here is the call graph for this function:

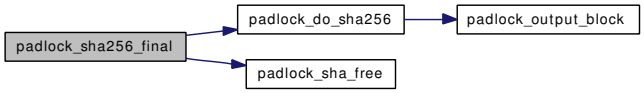


7.32.1.15 static void padlock_sha256_final (uint8_t * *hash*, struct padlock_sha_ctx * *ctx*) [static]

Definition at line 203 of file padlock_hash.c.

References padlock_do_sha256(), padlock_sha_free(), padlock_sha_ctx::psc_buf, and padlock_sha_ctx::psc_offset.

Here is the call graph for this function:



7.32.1.16 static void padlock_sha_free (struct padlock_sha_ctx * *ctx*) [static]

Definition at line 182 of file padlock_hash.c.

References padlock_sha_ctx::psc_buf, padlock_sha_ctx::psc_offset, and padlock_sha_ctx::psc_size.

Referenced by padlock_free_ctx(), padlock_sha1_final(), and padlock_sha256_final().

7.32.1.17 static void padlock_sha_init (struct padlock_sha_ctx * *ctx*) [static]

Definition at line 159 of file padlock_hash.c.

References padlock_sha_ctx::psc_buf, padlock_sha_ctx::psc_offset, and padlock_sha_ctx::psc_size.

7.32.1.18 static int padlock_sha_update (struct padlock_sha_ctx * *ctx*, uint8_t * *buf*, uint16_t *bufsize*) [static]

Definition at line 168 of file padlock_hash.c.

References padlock_sha_ctx::psc_buf, padlock_sha_ctx::psc_offset, and padlock_sha_ctx::psc_size.

7.32.2 Variable Documentation

7.32.2.1 struct auth_hash padlock_hmac_sha1 [static]

Initial value:

```
{
        CRYPTO_SHA1_HMAC, "HMAC-SHA1",
        20, SHA1_HASH_LEN, SHA1_HMAC_BLOCK_LEN, sizeof(struct padlock_sha_ctx),
        (void (*)(void *))padlock_sha_init,
        (int (*)(void *, uint8_t *, uint16_t))padlock_sha_update,
        (void (*)(uint8_t *, void *))padlock_sha1_final
}
```

Definition at line 81 of file padlock_hash.c.

7.32.2.2 struct auth_hash padlock_hmac_sha256 [static]

Initial value:

```
{
        CRYPTO_SHA2_256_HMAC, "HMAC-SHA2-256",
        32, SHA2_256_HASH_LEN, SHA2_256_HMAC_BLOCK_LEN, sizeof(struct padlock_sha_ctx),
        (void (*)(void *))padlock_sha_init,
        (int (*)(void *, uint8_t *, uint16_t))padlock_sha_update,
        (void (*)(uint8_t *, void *))padlock_sha256_final
}
```

Definition at line 89 of file padlock_hash.c.

Index