

# FreeBSD kernel crypto code Reference Manual

Generated by Doxygen 1.4.7

Sat Feb 24 11:04:51 2007



# Contents

<b>1</b>	<b>FreeBSD kernel crypto code Main Page</b>	<b>1</b>
<b>2</b>	<b>FreeBSD kernel crypto code Directory Hierarchy</b>	<b>3</b>
2.1	FreeBSD kernel crypto code Directories . . . . .	3
<b>3</b>	<b>FreeBSD kernel crypto code Data Structure Index</b>	<b>5</b>
3.1	FreeBSD kernel crypto code Data Structures . . . . .	5
<b>4</b>	<b>FreeBSD kernel crypto code File Index</b>	<b>7</b>
4.1	FreeBSD kernel crypto code File List . . . . .	7
<b>5</b>	<b>FreeBSD kernel crypto code Directory Documentation</b>	<b>9</b>
5.1	/usr/src/sys/crypto/blowfish/ Directory Reference . . . . .	9
5.2	/usr/src/sys/crypto/ Directory Reference . . . . .	10
5.3	/usr/src/sys/crypto/des/ Directory Reference . . . . .	11
5.4	/usr/src/sys/crypto/rc4/ Directory Reference . . . . .	12
5.5	/usr/src/sys/crypto/rijndael/ Directory Reference . . . . .	13
5.6	/usr/src/sys/crypto/sha2/ Directory Reference . . . . .	14
5.7	/usr/src/ Directory Reference . . . . .	15
5.8	/usr/src/sys/ Directory Reference . . . . .	16
5.9	/usr/ Directory Reference . . . . .	17
5.10	/usr/src/sys/crypto/via/ Directory Reference . . . . .	18
<b>6</b>	<b>FreeBSD kernel crypto code Data Structure Documentation</b>	<b>19</b>
6.1	_SHA256_CTX Struct Reference . . . . .	19
6.2	_SHA512_CTX Struct Reference . . . . .	21
6.3	bf_key_st Struct Reference . . . . .	22
6.4	cipherInstance Struct Reference . . . . .	23
6.5	des_ks_struct Struct Reference . . . . .	24
6.6	keyInstance Struct Reference . . . . .	25

6.7	<code>padlock_cw</code> Union Reference . . . . .	27
6.8	<code>padlock_session</code> Struct Reference . . . . .	29
6.9	<code>padlock_sha_ctx</code> Struct Reference . . . . .	31
6.10	<code>padlock_softc</code> Struct Reference . . . . .	32
6.11	<code>rc4_state</code> Struct Reference . . . . .	33
6.12	<code>rijndael_ctx</code> Struct Reference . . . . .	34
6.13	<code>sha1_ctxt</code> Struct Reference . . . . .	35
<b>7</b>	<b>FreeBSD kernel crypto code File Documentation</b>	<b>37</b>
7.1	<code>notreviewed.dox</code> File Reference . . . . .	37
7.2	<code>/usr/src/sys/crypto/blowfish/bf_ecb.c</code> File Reference . . . . .	38
7.3	<code>/usr/src/sys/crypto/blowfish/bf_enc.c</code> File Reference . . . . .	39
7.4	<code>/usr/src/sys/crypto/blowfish/bf_locl.h</code> File Reference . . . . .	41
7.5	<code>/usr/src/sys/crypto/blowfish/bf_pi.h</code> File Reference . . . . .	45
7.6	<code>/usr/src/sys/crypto/blowfish/bf_skey.c</code> File Reference . . . . .	46
7.7	<code>/usr/src/sys/crypto/blowfish/blowfish.h</code> File Reference . . . . .	47
7.8	<code>/usr/src/sys/crypto/des/des.h</code> File Reference . . . . .	50
7.9	<code>/usr/src/sys/crypto/des/des_ecb.c</code> File Reference . . . . .	56
7.10	<code>/usr/src/sys/crypto/des/des_enc.c</code> File Reference . . . . .	58
7.11	<code>/usr/src/sys/crypto/des/des_locl.h</code> File Reference . . . . .	60
7.12	<code>/usr/src/sys/crypto/des/des_setkey.c</code> File Reference . . . . .	65
7.13	<code>/usr/src/sys/crypto/des/podd.h</code> File Reference . . . . .	69
7.14	<code>/usr/src/sys/crypto/des/sk.h</code> File Reference . . . . .	70
7.15	<code>/usr/src/sys/crypto/des/spr.h</code> File Reference . . . . .	71
7.16	<code>/usr/src/sys/crypto/rc4/rc4.c</code> File Reference . . . . .	72
7.17	<code>/usr/src/sys/crypto/rc4/rc4.h</code> File Reference . . . . .	75
7.18	<code>/usr/src/sys/crypto/rijndael/rijndael-alg-fst.c</code> File Reference . . . . .	76
7.19	<code>/usr/src/sys/crypto/rijndael/rijndael-api-fst.c</code> File Reference . . . . .	81
7.20	<code>/usr/src/sys/crypto/rijndael/rijndael-api-fst.h</code> File Reference . . . . .	85
7.21	<code>/usr/src/sys/crypto/rijndael/rijndael-api.c</code> File Reference . . . . .	90
7.22	<code>/usr/src/sys/crypto/rijndael/rijndael.h</code> File Reference . . . . .	92
7.23	<code>/usr/src/sys/crypto/rijndael/rijndael_local.h</code> File Reference . . . . .	96
7.24	<code>/usr/src/sys/crypto/rijndael/test00.c</code> File Reference . . . . .	97
7.25	<code>/usr/src/sys/crypto/sha1.c</code> File Reference . . . . .	99
7.26	<code>/usr/src/sys/crypto/sha1.h</code> File Reference . . . . .	103
7.27	<code>/usr/src/sys/crypto/sha2/sha2.c</code> File Reference . . . . .	105
7.28	<code>/usr/src/sys/crypto/sha2/sha2.h</code> File Reference . . . . .	116

---

7.29	<a href="#">/usr/src/sys/crypto/via/padlock.c File Reference</a>	121
7.30	<a href="#">/usr/src/sys/crypto/via/padlock.h File Reference</a>	125
7.31	<a href="#">/usr/src/sys/crypto/via/padlock_cipher.c File Reference</a>	129
7.32	<a href="#">/usr/src/sys/crypto/via/padlock_hash.c File Reference</a>	133



# 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:

usr . . . . .	17
src . . . . .	15
sys . . . . .	16
crypto . . . . .	10
blowfish . . . . .	9
des . . . . .	11
rc4 . . . . .	12
rijndael . . . . .	13
sha2 . . . . .	14
via . . . . .	18



## 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:

<a href="#">_SHA256_CTX</a>	19
<a href="#">_SHA512_CTX</a>	21
<a href="#">bf_key_st</a>	22
<a href="#">cipherInstance</a>	23
<a href="#">des_ks_struct</a>	24
<a href="#">keyInstance</a>	25
<a href="#">padlock_cw</a>	27
<a href="#">padlock_session</a>	29
<a href="#">padlock_sha_ctx</a>	31
<a href="#">padlock_softc</a>	32
<a href="#">rc4_state</a>	33
<a href="#">rijndael_ctx</a>	34
<a href="#">sha1_ctxt</a>	35



## 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:

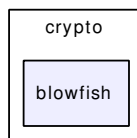
/usr/src/sys/crypto/sha1.c . . . . .	99
/usr/src/sys/crypto/sha1.h . . . . .	103
/usr/src/sys/crypto/blowfish/bf_ecb.c . . . . .	38
/usr/src/sys/crypto/blowfish/bf_enc.c . . . . .	39
/usr/src/sys/crypto/blowfish/bf_locl.h . . . . .	41
/usr/src/sys/crypto/blowfish/bf_pi.h . . . . .	45
/usr/src/sys/crypto/blowfish/bf_skey.c . . . . .	46
/usr/src/sys/crypto/blowfish/blowfish.h . . . . .	47
/usr/src/sys/crypto/des/des.h . . . . .	50
/usr/src/sys/crypto/des/des_ecb.c . . . . .	56
/usr/src/sys/crypto/des/des_enc.c . . . . .	58
/usr/src/sys/crypto/des/des_locl.h . . . . .	60
/usr/src/sys/crypto/des/des_setkey.c . . . . .	65
/usr/src/sys/crypto/des/podd.h . . . . .	69
/usr/src/sys/crypto/des/sk.h . . . . .	70
/usr/src/sys/crypto/des/spr.h . . . . .	71
/usr/src/sys/crypto/rc4/rc4.c . . . . .	72
/usr/src/sys/crypto/rc4/rc4.h . . . . .	75
/usr/src/sys/crypto/rijndael/rijndael-alg-fst.c . . . . .	76
/usr/src/sys/crypto/rijndael/rijndael-api-fst.c . . . . .	81
/usr/src/sys/crypto/rijndael/rijndael-api-fst.h . . . . .	85
/usr/src/sys/crypto/rijndael/rijndael-api.c . . . . .	90
/usr/src/sys/crypto/rijndael/rijndael.h . . . . .	92
/usr/src/sys/crypto/rijndael/rijndael_local.h . . . . .	96
/usr/src/sys/crypto/rijndael/test00.c . . . . .	97
/usr/src/sys/crypto/sha2/sha2.c . . . . .	105
/usr/src/sys/crypto/sha2/sha2.h . . . . .	116
/usr/src/sys/crypto/via/padlock.c . . . . .	121
/usr/src/sys/crypto/via/padlock.h . . . . .	125
/usr/src/sys/crypto/via/padlock_cipher.c . . . . .	129
/usr/src/sys/crypto/via/padlock_hash.c . . . . .	133



## Chapter 5

# FreeBSD kernel crypto code Directory Documentation

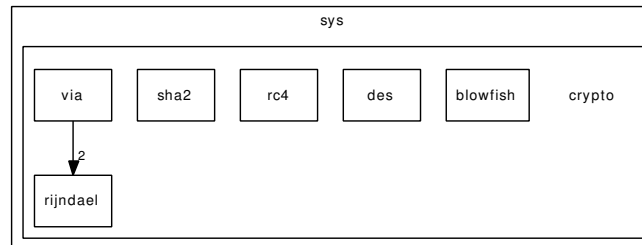
### 5.1 /usr/src/sys/crypto/blowfish/ Directory Reference



#### 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

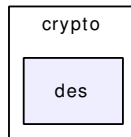
- directory [blowfish](#)
- directory [des](#)
- directory [rc4](#)
- directory [rijndael](#)
- directory [sha2](#)
- directory [via](#)

### Files

- file [sha1.c](#)
- file [sha1.h](#)



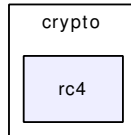
## 5.3 /usr/src/sys/crypto/des/ Directory Reference



### 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](#)

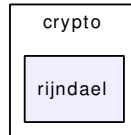
## 5.4 /usr/src/sys/crypto/rc4/ Directory Reference



### Files

- file [rc4.c](#)
- file [rc4.h](#)

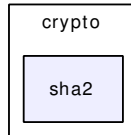
## 5.5 /usr/src/sys/crypto/rijndael/ Directory Reference



### 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](#)

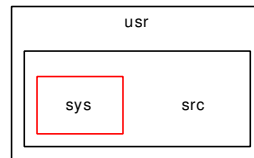
## 5.6 /usr/src/sys/crypto/sha2/ Directory Reference



### Files

- file [sha2.c](#)
- file [sha2.h](#)

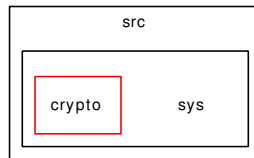
## 5.7 /usr/src/ Directory Reference



### Directories

- directory [sys](#)

## 5.8 /usr/src/sys/ Directory Reference



### Directories

- directory [crypto](#)

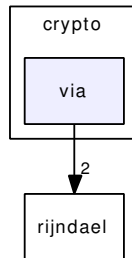
## 5.9 /usr/ Directory Reference



### 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]
- 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**

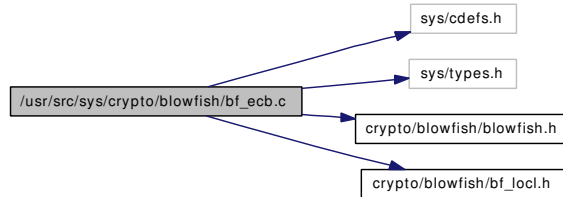
# **FreeBSD kernel crypto code File Documentation**

### **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

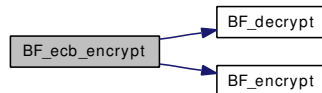
**7.2.1.1** `__FBSDID` ("\$FreeBSD: src/sys/crypto/blowfish/bf\_ecb.c, v 1.1 2003/10/13 19:26:08 ume Exp \$")

**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`.

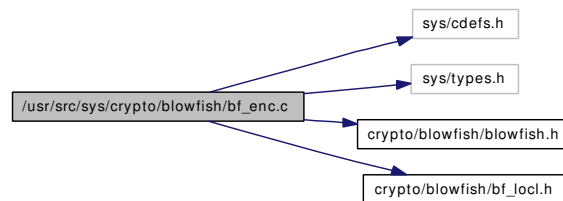
Here is the call graph for this function:



## 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()`.

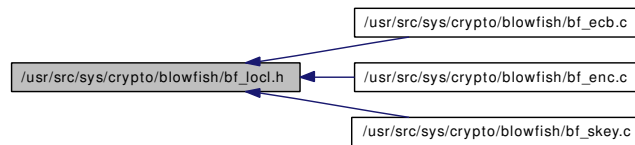
## 7.3.2 Variable Documentation

### 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]

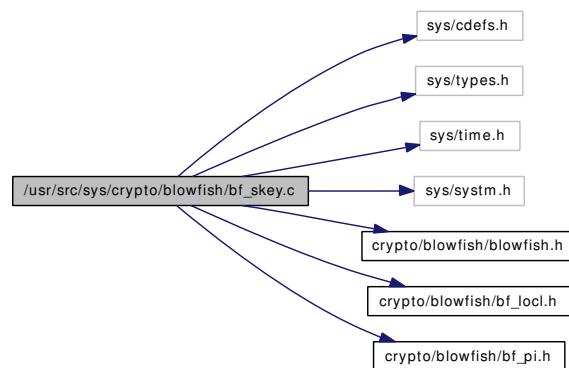
Definition at line 62 of file bf\_pi.h.

Referenced by BF\_set\_key().

## 7.6 /usr/src/sys/crypto/blowfish/bf\_key.c File Reference

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

Include dependency graph for bf\_key.c:



### Functions

- `__FBSDID` ("FreeBSD: src/sys/crypto/blowfish/bf\_key.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\_key.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\_key.c.

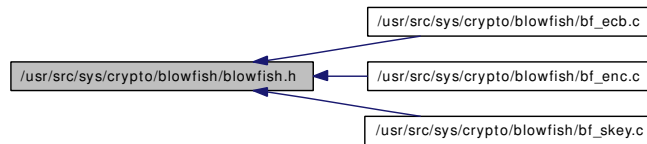
References `BF_encrypt()`, `bf_init`, `BF_LONG`, and `BF_ROUNDS`.

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.

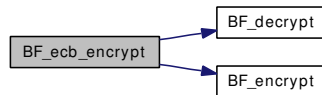
Referenced by BF\_ecb\_encrypt().

#### 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:



## 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\\_edec3\\_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:

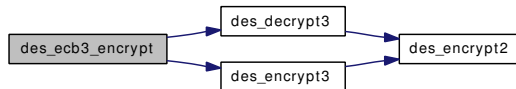


### 7.8.3.3 void [des\\_ecb3\\_encrypt](#) ([des\\_cblock](#) \*, [des\\_cblock](#) \*, [des\\_key\\_schedule](#), [des\\_key\\_schedule](#), [des\\_key\\_schedule](#), int)

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.

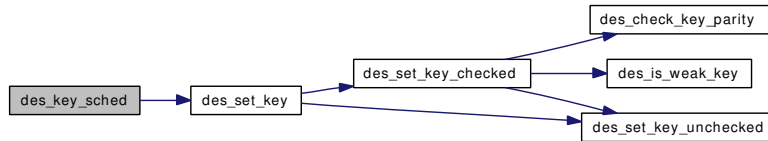
Referenced by des\_set\_key\_checked().

#### 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.

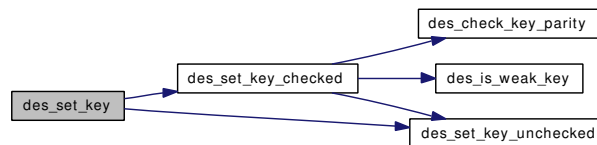
#### 7.8.3.14 int des\_set\_key (des\_cblock \*, des\_key\_schedule)

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:



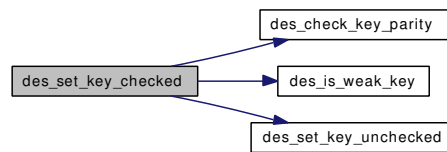
#### 7.8.3.15 int des\_set\_key\_checked (des\_cblock \*, des\_key\_schedule)

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`

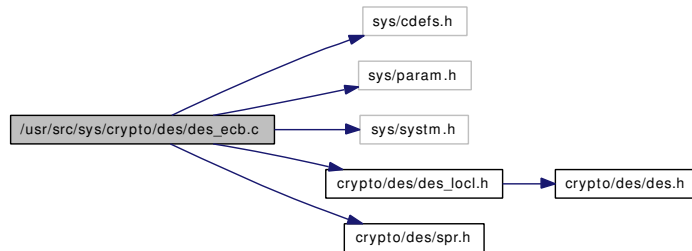
Definition at line 68 of file `des_setkey.c`.

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/system.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

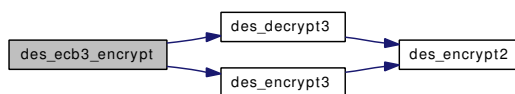
**7.9.1.1** `__FBSDID` ("\$FreeBSD: src/sys/crypto/des/des\_ecb. c, v 1.6 2004/06/14 00:38:54 obrien Exp \$")

**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:

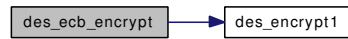


**7.9.1.3 void des\_ecb\_encrypt (des\_cblock \* input, des\_cblock \* output, des\_key\_schedule ks, int enc)**

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)**

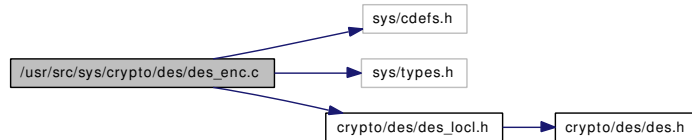
Definition at line 62 of file des\_ecb.c.

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().

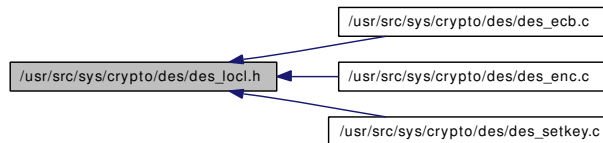
## 7.11 /usr/src/sys/crypto/des/des\_locl.h File Reference

```
#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, 4,0xf0f0f0fL); \
}

```

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, 4,0xf0f0f0fL); \
    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))))

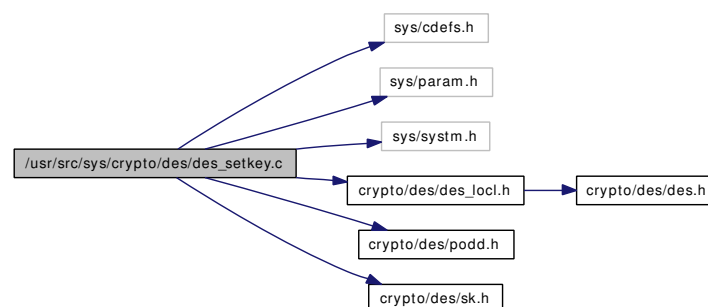
Definition at line 124 of file des\_locl.h.

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/system.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().

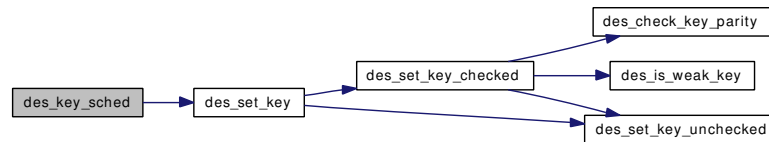


### 7.12.2.5 int des\_key\_sched (des\_cblock \* key, des\_key\_schedule schedule)

Definition at line 228 of file des\_setkey.c.

References des\_set\_key().

Here is the call graph for this function:



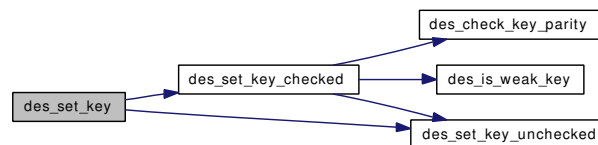
### 7.12.2.6 int des\_set\_key (des\_cblock \* key, des\_key\_schedule schedule)

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:



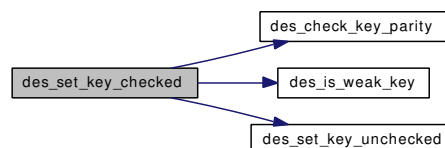
### 7.12.2.7 int des\_set\_key\_checked (des\_cblock \* key, des\_key\_schedule schedule)

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:**

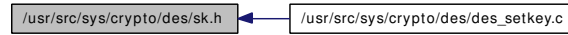
```
{
  1,  1,  2,  2,  4,  4,  7,  7,  8,  8, 11, 11, 13, 13, 14, 14,
 16, 16, 19, 19, 21, 21, 22, 22, 25, 25, 26, 26, 28, 28, 31, 31,
 32, 32, 35, 35, 37, 37, 38, 38, 41, 41, 42, 42, 44, 44, 47, 47,
 49, 49, 50, 50, 52, 52, 55, 55, 56, 56, 59, 59, 61, 61, 62, 62,
 64, 64, 67, 67, 69, 69, 70, 70, 73, 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]

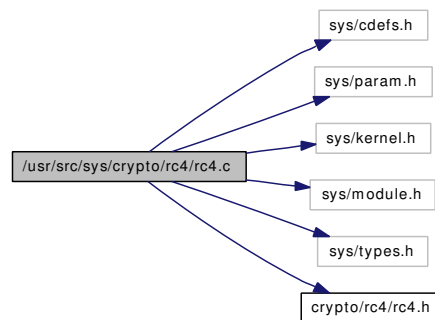
Definition at line 62 of file spr.h.

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\(\)](#).

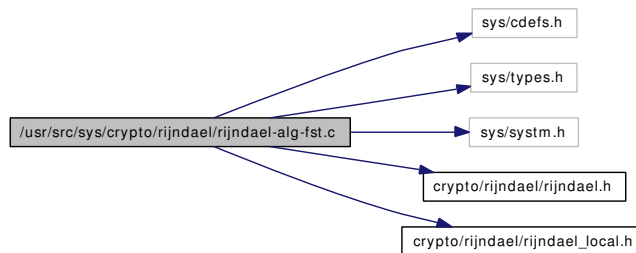
Here is the call graph for this function:



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

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

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



### Defines

- #define **SWAP**(x) ( $\_Irotl(x, 8) \& 0x00ff00ff \mid \_Irotr(x, 8) \& 0xff00ff00$ )
- #define **GETU32**(pt) ( $((u32)(pt)[0] \ll 24) \wedge ((u32)(pt)[1] \ll 16) \wedge ((u32)(pt)[2] \ll 8) \wedge ((u32)(pt)[3])$ )
- #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>

This code is hereby placed in the public domain.

THIS SOFTWARE IS PROVIDED BY THE AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

**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]`

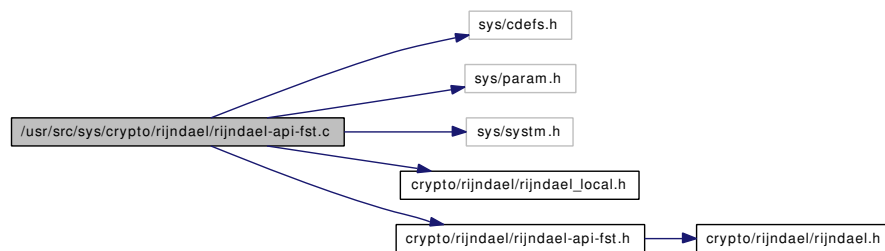
Definition at line 322 of file rijndael-alg-fst.c.

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/system.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 \$")
- `rijndael_makeKey` (`keyInstance` \*key, `BYTE` direction, int keyLen, char \*keyMaterial)
- `rijndael_cipherInit` (`cipherInstance` \*cipher, `BYTE` mode, char \*IV)
- `rijndael_blockEncrypt` (`cipherInstance` \*cipher, `keyInstance` \*key, `BYTE` \*input, int inputLen, `BYTE` \*outBuffer)
- `rijndael_padEncrypt` (`cipherInstance` \*cipher, `keyInstance` \*key, `BYTE` \*input, int inputOctets, `BYTE` \*outBuffer)
- `rijndael_blockDecrypt` (`cipherInstance` \*cipher, `keyInstance` \*key, `BYTE` \*input, int inputLen, `BYTE` \*outBuffer)
- `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 \$")

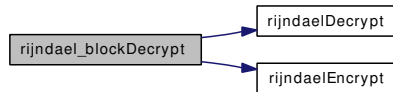
### 7.19.3.2 int rijndael\_blockDecrypt (cipherInstance \* cipher, keyInstance \* key, BYTE \* input, int inputLen, BYTE \* outBuffer)

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.

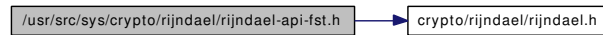
Here is the call graph for this function:



## 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.

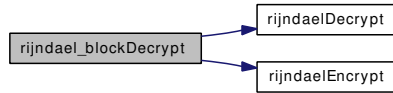
**7.20.2 Function Documentation****7.20.2.1 int rijndael\_blockDecrypt (cipherInstance \*, keyInstance \*, u\_int8\_t \*, int, u\_int8\_t \*)**

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:

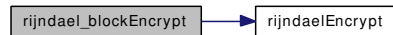


### 7.20.2.2 int rijndael\_blockEncrypt (cipherInstance \*, keyInstance \*, u\_int8\_t \*, int, u\_int8\_t \*)

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.

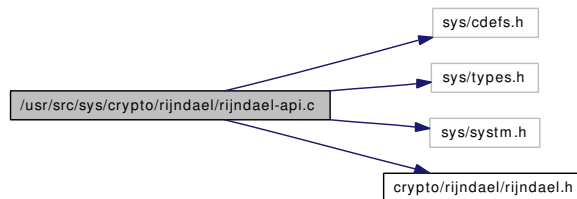
Here is the call graph for this function:



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

```
#include <sys/cdefs.h>
#include <sys/types.h>
#include <sys/system.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:



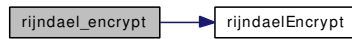
**7.21.1.3** void `rijndael_encrypt` (const `rijndael_ctx` \* ctx, const u\_char \* src, u\_char \* dst)

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.

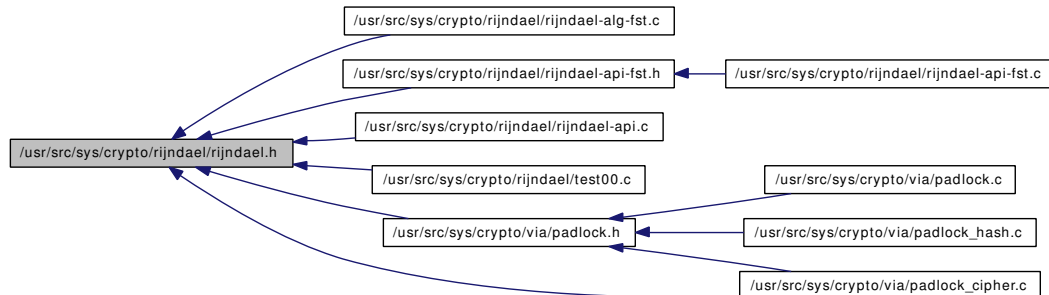
References rijndael\_ctx::dk, rijndael\_ctx::ek, rijndael\_ctx::Nr, rijndaelKeySetupDec(), and rijndaelKeySetupEnc().

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>

This code is hereby placed in the public domain.

THIS SOFTWARE IS PROVIDED BY THE AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

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:

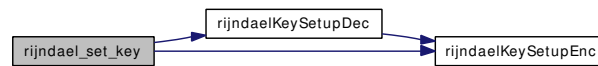


### 7.22.2.3 void rijndael\_set\_key (rijndael\_ctx \*, const u\_char \*, int)

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.

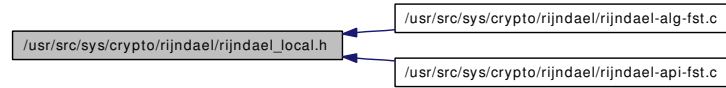
Definition at line 735 of file rijndael-alg-fst.c.

References GETU32, rcon, and Te4.

Referenced by padlock\_cipher\_key\_setup(), rijndael\_makeKey(), rijndael\_set\_key(), and rijndaelKey-SetupDec().

## 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.

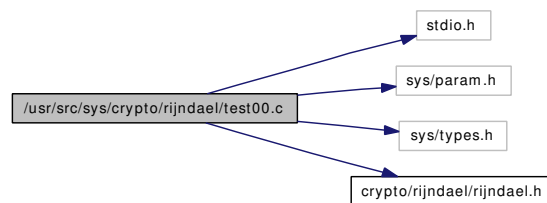
##### 7.23.1.3 typedef u\_int8\_t [u8](#)

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()`.

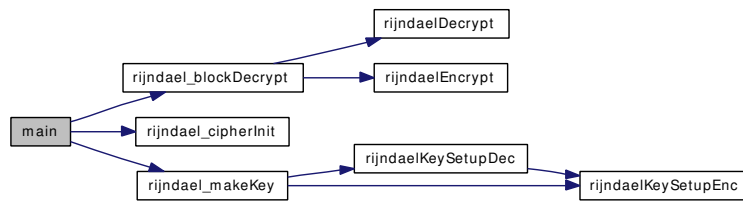
#### 7.24.2 Function Documentation

##### 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:

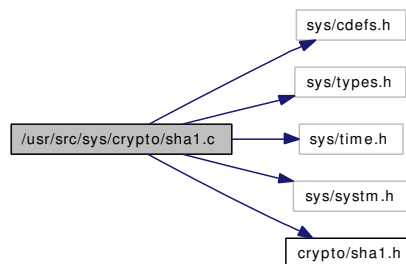




## 7.25 /usr/src/sys/crypto/sha1.c File Reference

```
#include <sys/cdefs.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/system.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:

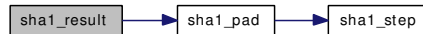


#### 7.25.2.5 void sha1\_result (struct sha1\_ctxt \* ctxt, caddr\_t digest0)

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:

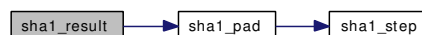


### 7.26.3.4 void sha1\_result (struct [sha1\\_ctxt](#) \*, caddr\_t)

Definition at line 252 of file sha1.c.

References sha1\_pad().

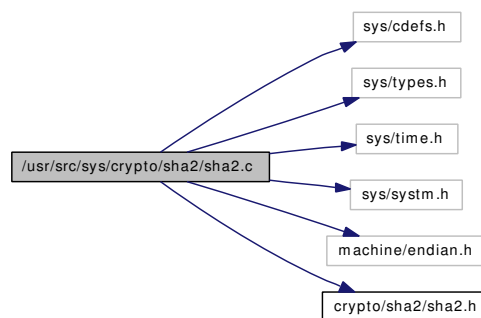
Here is the call graph for this function:



## 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/system.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

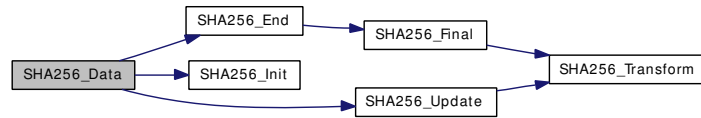
**7.27.3.1** `__FBSDID ("$FreeBSD: src/sys/crypto/sha2/sha2. c, v 1.9 2006/10/22 02:19:33 kevlo Exp $")`

**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:



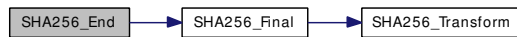
### 7.27.3.3 char\* SHA256\_End (SHA256\_CTX \* context, char buffer[ ])

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().

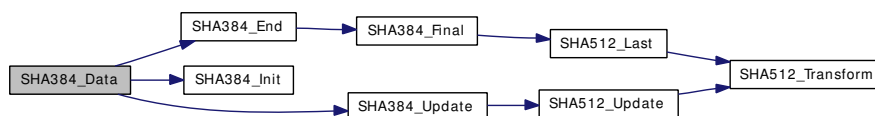
Here is the call graph for this function:

**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:

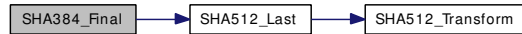
**7.27.3.10 void SHA384\_Final (sha2\_byte digest[ ], SHA384\_CTX \* context)**

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()`.

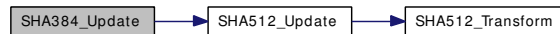
### 7.27.3.12 void SHA384\_Update (SHA384\_CTX \* context, const sha2\_byte \* data, size\_t len)

Definition at line 989 of file sha2.c.

References `SHA512_Update()`.

Referenced by `SHA384_Data()`.

Here is the call graph for this function:

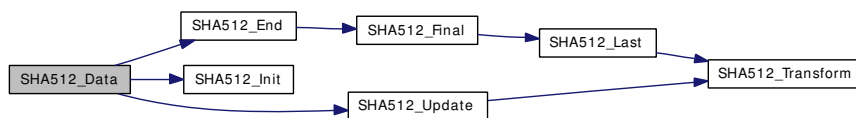


### 7.27.3.13 char\* SHA512\_Data (const sha2\_byte \* data, size\_t len, char digest[SHA512\_DIGEST\_STRING\_LENGTH])

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:



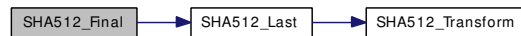
**7.27.3.15 void SHA512\_Final (sha2\_byte digest[], SHA512\_CTX \* context)**

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,
    0x152fec8f70e5939ULL,
    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](#)

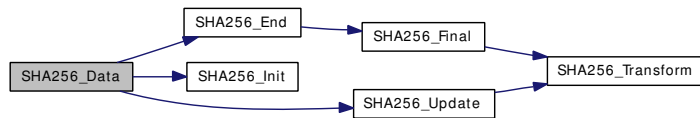
## 7.28.3 Function Documentation

### 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:

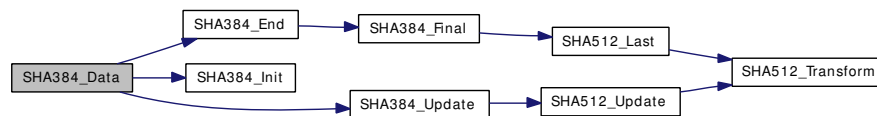


### 7.28.3.6 char\* SHA384\_Data (const u\_int8\_t \*, size\_t, char[SHA384\_DIGEST\_STRING\_LENGTH])

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().

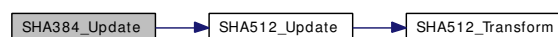
### 7.28.3.10 void SHA384\_Update (SHA384\_CTX \*, const u\_int8\_t \*, size\_t)

Definition at line 989 of file sha2.c.

References SHA512\_Update().

Referenced by SHA384\_Data().

Here is the call graph for this function:

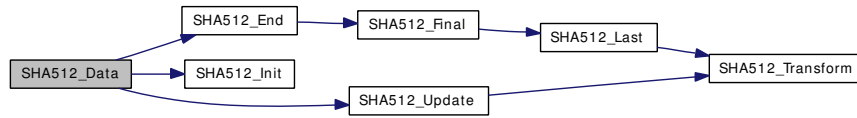


### 7.28.3.11 char\* SHA512\_Data (const u\_int8\_t \*, size\_t, char[SHA512\_DIGEST\_STRING\_LENGTH])

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()`.

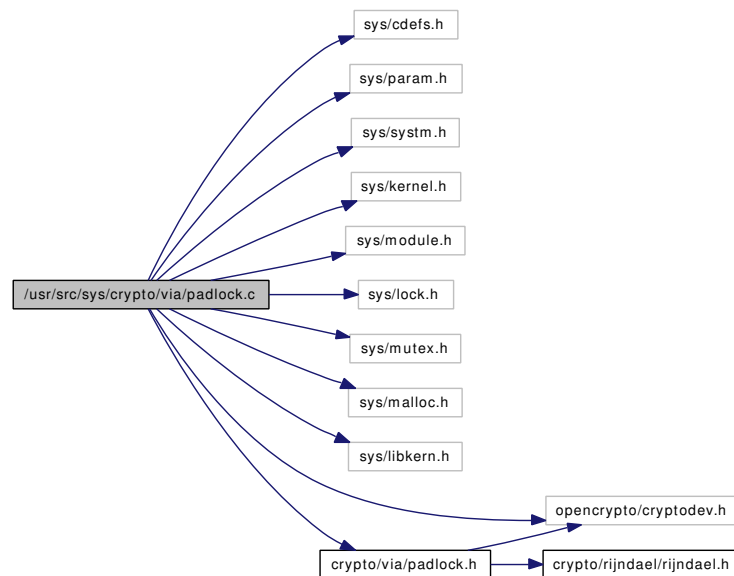
Here is the call graph for this function:



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

```
#include <sys/cdefs.h>
#include <sys/param.h>
#include <sys/system.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]

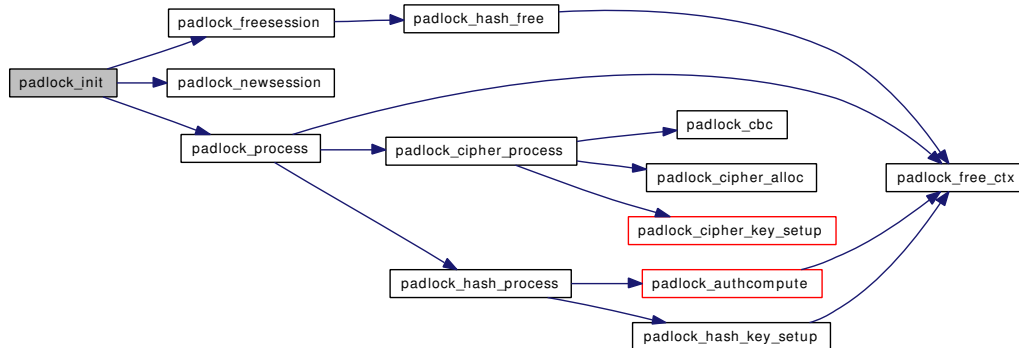
Definition at line 73 of file padlock.c.

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:

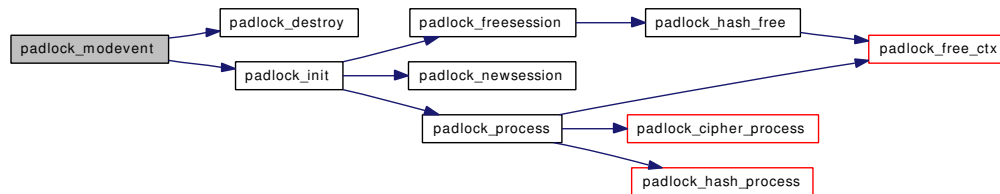


#### 7.29.1.9 static int padlock\_modevent (module\_t mod, int type, void \*unused \_\_unused) [static]

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().

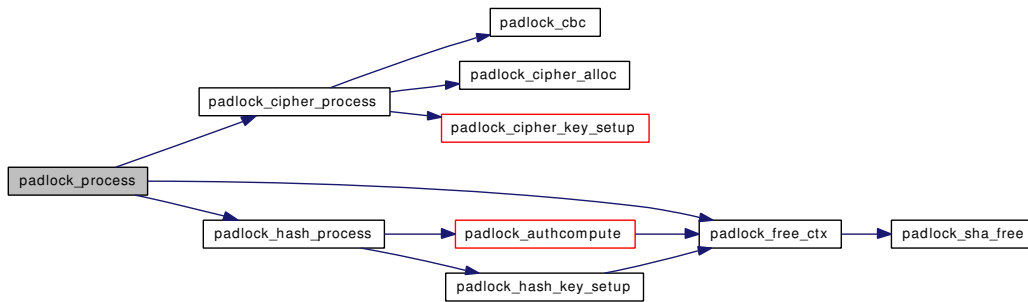
#### 7.29.1.11 static int padlock\_process (void \*arg \_\_unused, struct cryptop \* crp, int hint \_\_unused) [static]

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]

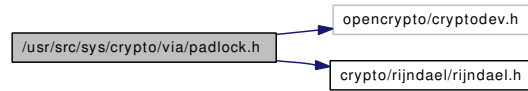
Definition at line 62 of file `padlock.c`.

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

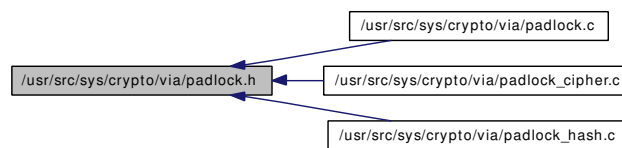
## 7.30 /usr/src/sys/crypto/via/padlock.h File Reference

```
#include <openssl/crypto/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 \*encrd, struct cryptop \*crp)
- int [padlock\\_hash\\_setup](#) (struct [padlock\\_session](#) \*ses, struct cryptoini \*macini)
- int [padlock\\_hash\\_process](#) (struct [padlock\\_session](#) \*ses, struct cryptodesc \*macrd, 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

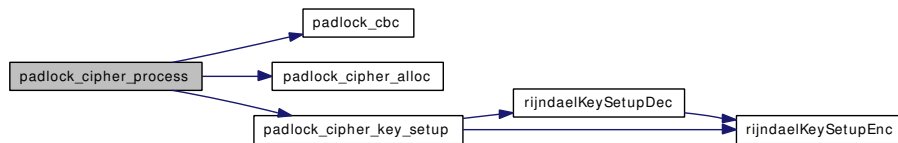
### 7.30.2.1 `int padlock_cipher_process (struct padlock_session * ses, struct cryptodesc * encprd, struct cryptop * crp)`

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:

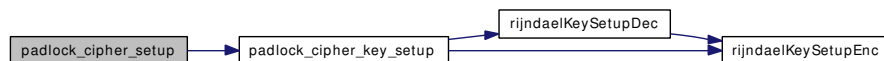


### 7.30.2.2 `int padlock_cipher_setup (struct padlock_session * ses, struct cryptoini * encini)`

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:



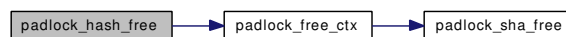
### 7.30.2.3 `void padlock_hash_free (struct padlock_session * ses)`

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:



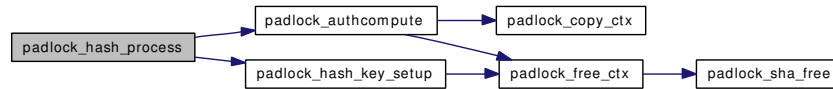
#### 7.30.2.4 `int padlock_hash_process (struct padlock_session * ses, struct cryptodesc * maccrd, struct cryptop * crp)`

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:

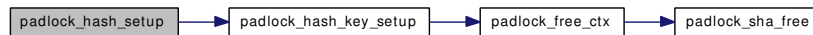


#### 7.30.2.5 `int padlock_hash_setup (struct padlock_session * ses, struct cryptoini * macini)`

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`.

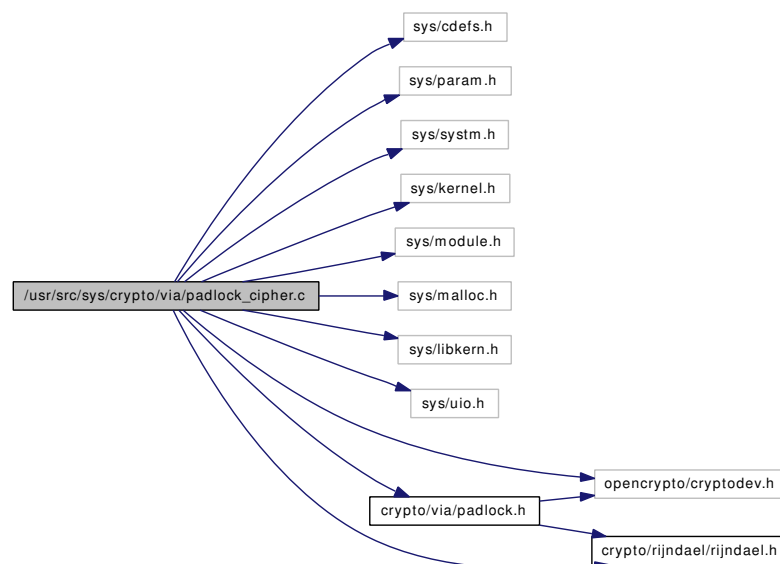
Here is the call graph for this function:



## 7.31 /usr/src/sys/crypto/via/padlock\_cipher.c File Reference

```
#include <sys/cdefs.h>
#include <sys/param.h>
#include <sys/system.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

- [\\_\\_FBSID](#) ("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 \*encrd, struct cryptop \*crp, int \*allocated)
- int [padlock\\_cipher\\_process](#) (struct [padlock\\_session](#) \*ses, struct cryptodesc \*encrd, 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 * encrd, struct cryptop * crp, int  
* allocated) [static]`

Definition at line 168 of file padlock\_cipher.c.

Referenced by padlock\_cipher\_process().

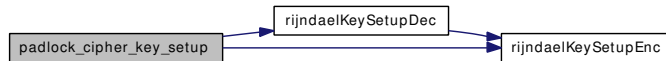
### 7.31.2.5 `static void padlock_cipher_key_setup (struct padlock_session * ses, caddr_t key, int klen)` [static]

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:



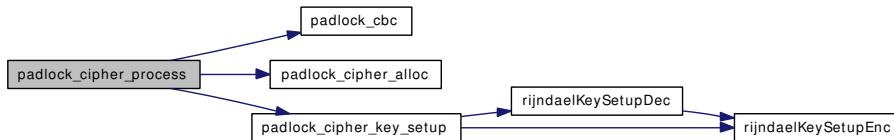
### 7.31.2.6 `int padlock_cipher_process (struct padlock_session * ses, struct cryptodesc * encrdd, struct cryptop * crp)`

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:

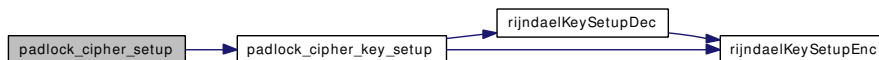


### 7.31.2.7 `int padlock_cipher_setup (struct padlock_session * ses, struct cryptoini * encini)`

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.

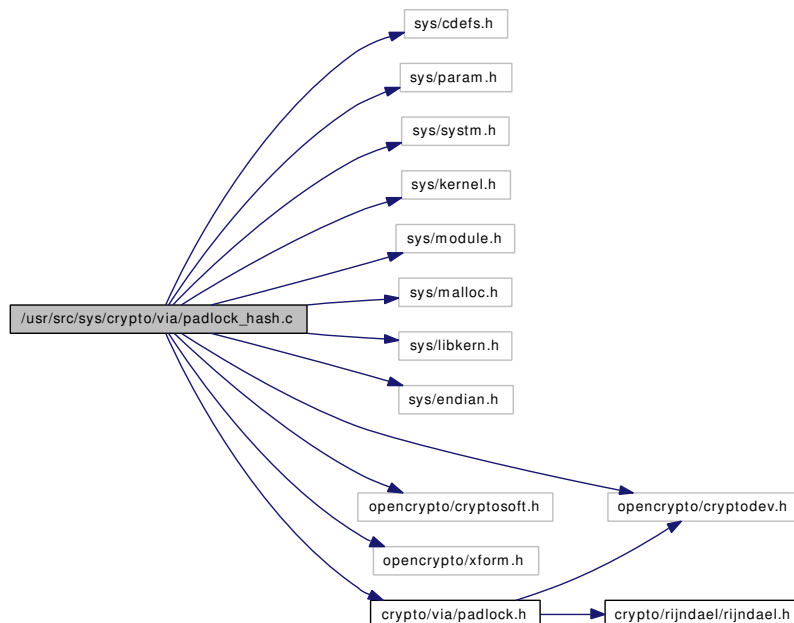
Here is the call graph for this function:



## 7.32 /usr/src/sys/crypto/via/padlock\_hash.c File Reference

```
#include <sys/cdefs.h>
#include <sys/param.h>
#include <sys/system.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` \*macrd, 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)

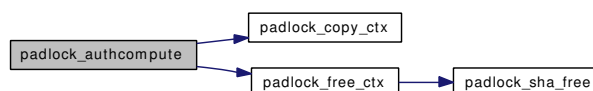
**7.32.1.4** `static int padlock_authcompute` (struct `padlock_session` \*ses, struct `cryptodesc` \*crd, caddr\_t buf, int flags) [static]

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:

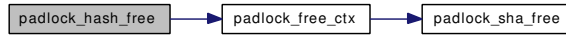
**7.32.1.9 void padlock\_hash\_free (struct padlock\_session \* *ses*)**

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:



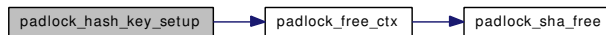
### 7.32.1.10 `static void padlock_hash_key_setup (struct padlock\_session * ses, caddr\_t key, int klen)` [static]

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:



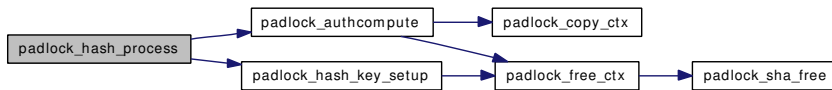
### 7.32.1.11 `int padlock_hash_process (struct padlock\_session * ses, struct cryptodesc * macrd, struct cryptop * crp)`

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:

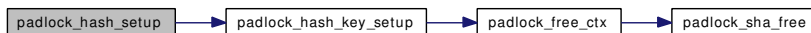


### 7.32.1.12 `int padlock_hash_setup (struct padlock\_session * ses, struct cryptoini * macini)`

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.

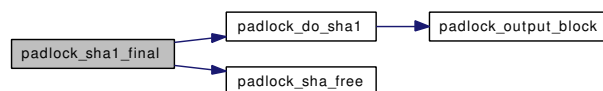
Referenced by padlock\_do\_sha1(), and padlock\_do\_sha256().

**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:

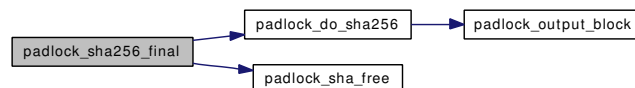


**7.32.1.15** `static void padlock_sha256_final (uint8_t * hash, struct padlock_sha_ctx * ctx)`  
 [static]

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

- /usr/ Directory Reference, 17
- /usr/src/ Directory Reference, 15
- /usr/src/sys/ Directory Reference, 16
- /usr/src/sys/crypto/ Directory Reference, 10
- /usr/src/sys/crypto/blowfish/ Directory Reference, 9
- /usr/src/sys/crypto/blowfish/bf\_ecb.c, 38
- /usr/src/sys/crypto/blowfish/bf\_enc.c, 39
- /usr/src/sys/crypto/blowfish/bf\_locl.h, 41
- /usr/src/sys/crypto/blowfish/bf\_pi.h, 45
- /usr/src/sys/crypto/blowfish/bf\_skey.c, 46
- /usr/src/sys/crypto/blowfish/blowfish.h, 47
- /usr/src/sys/crypto/des/ Directory Reference, 11
- /usr/src/sys/crypto/des/des.h, 50
- /usr/src/sys/crypto/des/des\_ecb.c, 56
- /usr/src/sys/crypto/des/des\_enc.c, 58
- /usr/src/sys/crypto/des/des\_locl.h, 60
- /usr/src/sys/crypto/des/des\_setkey.c, 65
- /usr/src/sys/crypto/des/podd.h, 69
- /usr/src/sys/crypto/des/sk.h, 70
- /usr/src/sys/crypto/des/spr.h, 71
- /usr/src/sys/crypto/rc4/ Directory Reference, 12
- /usr/src/sys/crypto/rc4/rc4.c, 72
- /usr/src/sys/crypto/rc4/rc4.h, 75
- /usr/src/sys/crypto/rijndael/ Directory Reference, 13
- /usr/src/sys/crypto/rijndael/rijndael-alg-fst.c, 76
- /usr/src/sys/crypto/rijndael/rijndael-api-fst.c, 81
- /usr/src/sys/crypto/rijndael/rijndael-api-fst.h, 85
- /usr/src/sys/crypto/rijndael/rijndael-api.c, 90
- /usr/src/sys/crypto/rijndael/rijndael.h, 92
- /usr/src/sys/crypto/rijndael/rijndael\_local.h, 96
- /usr/src/sys/crypto/rijndael/test00.c, 97
- /usr/src/sys/crypto/sha1.c, 99
- /usr/src/sys/crypto/sha1.h, 103
- /usr/src/sys/crypto/sha2/ Directory Reference, 14
- /usr/src/sys/crypto/sha2/sha2.c, 105
- /usr/src/sys/crypto/sha2/sha2.h, 116
- /usr/src/sys/crypto/via/ Directory Reference, 18
- /usr/src/sys/crypto/via/padlock.c, 121
- /usr/src/sys/crypto/via/padlock.h, 125
- /usr/src/sys/crypto/via/padlock\_cipher.c, 129
- /usr/src/sys/crypto/via/padlock\_hash.c, 133
- \_K
  - sha1.c, 102
- \_SHA256\_CTX, 19
  - bitcount, 19
  - buffer, 19
  - state, 19
- \_SHA512\_CTX, 21
  - bitcount, 21
  - buffer, 21
  - state, 21
- \_\_FBSDID
  - bf\_ecb.c, 38
  - bf\_enc.c, 39
  - bf\_skey.c, 46
  - des\_ecb.c, 56
  - des\_enc.c, 58
  - des\_setkey.c, 66
  - padlock.c, 122
  - padlock\_cipher.c, 131
  - padlock\_hash.c, 134
  - rc4.c, 73
  - rijndael-alg-fst.c, 77
  - rijndael-api-fst.c, 82
  - rijndael-api.c, 90
  - sha1.c, 101
  - sha2.c, 109
- \_\_aligned
  - padlock\_session, 29
- \_\_field
  - padlock\_cw, 27
- ADDINC128
  - sha2.c, 106
- algorithm\_type
  - padlock\_cw, 27
- assert
  - sha2.c, 107
- b32
  - sha1\_ctxt, 35
- b64
  - sha1\_ctxt, 35
- b8
  - sha1\_ctxt, 35
- BAD\_BLOCK\_LENGTH
  - rijndael-api-fst.h, 86
- BAD\_CIPHER\_INSTANCE
  - rijndael-api-fst.h, 86
- BAD\_CIPHER\_MODE

- rijndael-api-fst.h, 86
- BAD\_CIPHER\_STATE
  - rijndael-api-fst.h, 86
- BAD\_DATA
  - rijndael-api-fst.h, 86
- BAD\_KEY\_DIR
  - rijndael-api-fst.h, 86
- BAD\_KEY\_INSTANCE
  - rijndael-api-fst.h, 86
- BAD\_KEY\_MAT
  - rijndael-api-fst.h, 86
- BAD\_OTHER
  - rijndael-api-fst.h, 86
- BCOUNT
  - sha1.c, 100
- BF\_0
  - bf\_locl.h, 41
- BF\_1
  - bf\_locl.h, 41
- BF\_2
  - bf\_locl.h, 41
- BF\_3
  - bf\_locl.h, 41
- BF\_BLOCK
  - blowfish.h, 47
- BF\_DECRYPT
  - blowfish.h, 47
- BF\_decrypt
  - bf\_enc.c, 39
  - blowfish.h, 48
- bf\_ecb.c
  - \_\_FBSDDID, 38
  - BF\_ecb\_encrypt, 38
- BF\_ecb\_encrypt
  - bf\_ecb.c, 38
  - blowfish.h, 48
- BF\_ENC
  - bf\_locl.h, 41
- bf\_enc.c
  - \_\_FBSDDID, 39
  - BF\_decrypt, 39
  - BF\_encrypt, 39
  - or, 40
- BF\_ENCRYPT
  - blowfish.h, 47
- BF\_encrypt
  - bf\_enc.c, 39
  - blowfish.h, 48
- bf\_init
  - bf\_pi.h, 45
- BF\_KEY
  - blowfish.h, 48
- bf\_key\_st, 22
  - P, 22
- S, 22
- bf\_locl.h
  - BF\_0, 41
  - BF\_1, 41
  - BF\_2, 41
  - BF\_3, 41
  - BF\_ENC, 41
  - BF\_M, 42
  - c2l, 42
  - c2ln, 42
  - l2c, 42
  - l2cn, 43
  - l2n, 43
  - l2nn, 43
  - n2l, 44
  - n2ln, 44
- BF\_LONG
  - blowfish.h, 47
- BF\_M
  - bf\_locl.h, 42
- bf\_pi.h
  - bf\_init, 45
- BF\_ROUNDS
  - blowfish.h, 48
- BF\_set\_key
  - bf\_skey.c, 46
  - blowfish.h, 48
- bf\_skey.c
  - \_\_FBSDDID, 46
  - BF\_set\_key, 46
- bitcount
  - \_SHA256\_CTX, 19
  - \_SHA512\_CTX, 21
- BITSPERBLOCK
  - rijndael-api-fst.h, 86
- blowfish.h
  - BF\_BLOCK, 47
  - BF\_DECRYPT, 47
  - BF\_decrypt, 48
  - BF\_ecb\_encrypt, 48
  - BF\_ENCRYPT, 47
  - BF\_encrypt, 48
  - BF\_KEY, 48
  - BF\_LONG, 47
  - BF\_ROUNDS, 48
  - BF\_set\_key, 48
- BSIZE
  - des\_locl.h, 60
- buffer
  - \_SHA256\_CTX, 19
  - \_SHA512\_CTX, 21
- BYTE
  - rijndael-api-fst.c, 82

- c
  - sha1\_ctxt, 35
- c21
  - bf\_locl.h, 42
  - des\_locl.h, 60
- c2ln
  - bf\_locl.h, 42
  - des\_locl.h, 61
- cblock
  - des\_ks\_struct, 24
- Ch
  - sha2.c, 107
- cipherInstance, 23
- cipherInstance
  - IV, 23
  - mode, 23
- COUNT
  - sha1.c, 100
- count
  - sha1\_ctxt, 35
- CTASSERT
  - padlock\_hash.c, 134
- cw\_algorithm\_type
  - padlock.h, 126
- cw\_direction
  - padlock.h, 126
- cw\_filler0
  - padlock.h, 126
- cw\_filler1
  - padlock.h, 126
- cw\_filler2
  - padlock.h, 126
- cw\_filler3
  - padlock.h, 126
- cw\_intermediate
  - padlock.h, 126
- cw\_key\_generation
  - padlock.h, 126
- cw\_key\_size
  - padlock.h, 126
- cw\_round\_count
  - padlock.h, 126
- D\_ENCRYPT
  - des\_locl.h, 61
- DECLARE\_MODULE
  - padlock.c, 122
  - rc4.c, 73
- decrypt
  - rijndael\_ctx, 34
- des.h
  - DES\_CBC\_MODE, 51
  - des\_cblock, 52
  - des\_check\_key, 55
  - des\_check\_key\_parity, 52
  - DES\_DECRYPT, 51
  - des\_decrypt3, 52
  - des\_ecb3\_encrypt, 52
  - des\_ecb\_encrypt, 52
  - des\_ed3\_cbc\_encrypt, 53
  - DES\_ENCRYPT, 51
  - des\_encrypt1, 53
  - des\_encrypt2, 53
  - des\_encrypt3, 53
  - des\_fixup\_key\_parity, 53
  - des\_is\_weak\_key, 53
  - des\_key\_sched, 54
  - des\_key\_schedule, 52
  - DES\_KEY\_SZ, 51
  - DES\_LONG, 51
  - des\_ncbc\_encrypt, 54
  - des\_options, 54
  - DES\_PCBC\_MODE, 51
  - DES\_SCHEDULE\_SZ, 51
  - des\_set\_key, 54
  - des\_set\_key\_checked, 54
  - des\_set\_key\_unchecked, 55
  - des\_set\_odd\_parity, 55
  - DES\_CBC\_MODE
    - des.h, 51
  - des\_cblock
    - des.h, 52
  - des\_check\_key
    - des.h, 55
    - des\_setkey.c, 68
  - des\_check\_key\_parity
    - des.h, 52
    - des\_setkey.c, 66
  - DES\_DECRYPT
    - des.h, 51
  - des\_decrypt3
    - des.h, 52
    - des\_enc.c, 58
  - des\_ecb.c
    - \_\_FBSID, 56
    - des\_ecb3\_encrypt, 56
    - des\_ecb\_encrypt, 56
    - des\_options, 57
  - des\_ecb3\_encrypt
    - des.h, 52
    - des\_ecb.c, 56
  - des\_ecb\_encrypt
    - des.h, 52
    - des\_ecb.c, 56
  - des\_ed3\_cbc\_encrypt
    - des.h, 53
  - des\_enc.c
    - \_\_FBSID, 58

- des\_decrypt3, 58
- des\_encrypt1, 58
- des\_encrypt2, 59
- des\_encrypt3, 59
- des\_SPtrans, 59
- DES\_ENCRYPT
  - des.h, 51
- des\_encrypt1
  - des.h, 53
  - des\_enc.c, 58
- des\_encrypt2
  - des.h, 53
  - des\_enc.c, 59
- des\_encrypt3
  - des.h, 53
  - des\_enc.c, 59
- des\_fixup\_key\_parity
  - des.h, 53
  - des\_setkey.c, 66
- des\_is\_weak\_key
  - des.h, 53
  - des\_setkey.c, 66
- des\_key\_sched
  - des.h, 54
  - des\_setkey.c, 66
- des\_key\_schedule
  - des.h, 52
- DES\_KEY\_SZ
  - des.h, 51
- des\_ks\_struct, 24
  - cblock, 24
  - deslong, 24
  - ks, 24
  - weak\_key, 24
- des\_locl.h
  - BSIZE, 60
  - c2l, 60
  - c2ln, 61
  - D\_ENCRYPT, 61
  - FP, 61
  - HALF\_ITERATIONS, 62
  - HDRSIZE, 62
  - IP, 62
  - ITERATIONS, 62
  - l2c, 62
  - l2cn, 62
  - l2n, 63
  - LOAD\_DATA, 63
  - LOAD\_DATA\_tmp, 63
  - MAXWRITE, 63
  - n2l, 63
  - PERM\_OP, 64
  - ROTATE, 64
- DES\_LONG
  - des.h, 51
- des\_ncbc\_encrypt
  - des.h, 54
- des\_options
  - des.h, 54
  - des\_ecb.c, 57
- DES\_PCBC\_MODE
  - des.h, 51
- DES\_SCHEDULE\_SZ
  - des.h, 51
- des\_set\_key
  - des.h, 54
  - des\_setkey.c, 67
- des\_set\_key\_checked
  - des.h, 54
  - des\_setkey.c, 67
- des\_set\_key\_unchecked
  - des.h, 55
  - des\_setkey.c, 67
- des\_set\_odd\_parity
  - des.h, 55
  - des\_setkey.c, 68
- des\_setkey.c
  - \_\_FBSDID, 66
  - des\_check\_key, 68
  - des\_check\_key\_parity, 66
  - des\_fixup\_key\_parity, 66
  - des\_is\_weak\_key, 66
  - des\_key\_sched, 66
  - des\_set\_key, 67
  - des\_set\_key\_checked, 67
  - des\_set\_key\_unchecked, 67
  - des\_set\_odd\_parity, 68
  - HPERM\_OP, 66
  - NUM\_WEAK\_KEY, 66
  - weak\_keys, 68
- des\_skb
  - sk.h, 70
- des\_SPtrans
  - des\_enc.c, 59
  - spr.h, 71
- deslong
  - des\_ks\_struct, 24
- DIR\_DECRYPT
  - rijndael-api-fst.h, 87
- DIR\_ENCRYPT
  - rijndael-api-fst.h, 87
- direction
  - keyInstance, 25
  - padlock\_cw, 27
- dk
  - rijndael\_ctx, 34
- ek

- keyInstance, [25](#)
- rijndael\_ctx, [34](#)
- F0
  - sha1.c, [100](#)
- F1
  - sha1.c, [100](#)
- F2
  - sha1.c, [100](#)
- F3
  - sha1.c, [100](#)
- filler0
  - padlock\_cw, [27](#)
- filler1
  - padlock\_cw, [27](#)
- filler2
  - padlock\_cw, [27](#)
- filler3
  - padlock\_cw, [27](#)
- FP
  - des\_locl.h, [61](#)
- GETU32
  - rijndael-alg-fst.c, [77](#)
- H
  - sha1.c, [100](#)
- h
  - sha1\_ctxt, [35](#)
- HALF\_ITERATIONS
  - des\_locl.h, [62](#)
- HDRSIZE
  - des\_locl.h, [62](#)
- HPERM\_OP
  - des\_setkey.c, [66](#)
- index1
  - rc4\_state, [33](#)
- index2
  - rc4\_state, [33](#)
- intermediate
  - padlock\_cw, [28](#)
- IP
  - des\_locl.h, [62](#)
- ITERATIONS
  - des\_locl.h, [62](#)
- IV
  - cipherInstance, [23](#)
- K
  - sha1.c, [100](#)
- K256
  - sha2.c, [114](#)
- K512
  - sha2.c, [114](#)
- key\_generation
  - padlock\_cw, [28](#)
- key\_size
  - padlock\_cw, [28](#)
- keyInstance, [25](#)
- keyInstance
  - direction, [25](#)
  - ek, [25](#)
  - keyLen, [25](#)
  - keyMaterial, [25](#)
  - Nr, [25](#)
  - rk, [25](#)
- keyLen
  - keyInstance, [25](#)
- keyMaterial
  - keyInstance, [25](#)
- ks
  - des\_ks\_struct, [24](#)
- l2c
  - bf\_locl.h, [42](#)
  - des\_locl.h, [62](#)
- l2cn
  - bf\_locl.h, [43](#)
  - des\_locl.h, [62](#)
- l2n
  - bf\_locl.h, [43](#)
  - des\_locl.h, [63](#)
- l2nn
  - bf\_locl.h, [43](#)
- LL
  - test00.c, [97](#)
- LOAD\_DATA
  - des\_locl.h, [63](#)
- LOAD\_DATA\_tmp
  - des\_locl.h, [63](#)
- m
  - sha1\_ctxt, [36](#)
- main
  - test00.c, [97](#)
- Maj
  - sha2.c, [107](#)
- MALLOC\_DECLARE
  - padlock\_cipher.c, [131](#)
  - padlock\_hash.c, [134](#)
- MALLOC\_DEFINE
  - padlock.c, [122](#)
- MAXWRITE
  - des\_locl.h, [63](#)
- mode
  - cipherInstance, [23](#)
- MODE\_CBC
  - rijndael-api-fst.h, [87](#)

- MODE\_CFB1
  - rijndael-api-fst.h, 87
- MODE\_ECB
  - rijndael-api-fst.h, 87
- MODULE\_DEPEND
  - padlock.c, 122
- MODULE\_VERSION
  - padlock.c, 122
  - rc4.c, 73
- n2l
  - bf\_locl.h, 44
  - des\_locl.h, 63
- n2ln
  - bf\_locl.h, 44
- notreviewed.dox, 37
- Nr
  - keyInstance, 25
  - rijndael\_ctx, 34
- NUM\_WEAK\_KEY
  - des\_setkey.c, 66
- odd\_parity
  - podd.h, 69
- or
  - bf\_enc.c, 40
- P
  - bf\_key\_st, 22
- padlock.c
  - \_\_FBSDID, 122
  - DECLARE\_MODULE, 122
  - MALLOC\_DEFINE, 122
  - MODULE\_DEPEND, 122
  - MODULE\_VERSION, 122
  - padlock\_destroy, 122
  - padlock\_freesession, 122
  - padlock\_init, 122
  - padlock\_mod, 124
  - padlock\_modevent, 123
  - padlock\_newsession, 123
  - padlock\_process, 123
  - padlock\_sc, 124
- padlock.h
  - cw\_algorithm\_type, 126
  - cw\_direction, 126
  - cw\_filler0, 126
  - cw\_filler1, 126
  - cw\_filler2, 126
  - cw\_filler3, 126
  - cw\_intermediate, 126
  - cw\_key\_generation, 126
  - cw\_key\_size, 126
  - cw\_round\_count, 126
  - PADLOCK\_ALIGN, 126
  - padlock\_cipher\_process, 127
  - padlock\_cipher\_setup, 127
  - padlock\_hash\_free, 127
  - padlock\_hash\_process, 127
  - padlock\_hash\_setup, 128
  - PADLOCK\_ALGORITHM\_TYPE\_AES
    - padlock\_cipher.c, 130
  - PADLOCK\_ALIGN
    - padlock.h, 126
  - padlock\_authcompute
    - padlock\_hash.c, 134
  - padlock\_cbc
    - padlock\_cipher.c, 131
  - padlock\_cipher.c
    - \_\_FBSDID, 131
    - MALLOC\_DECLARE, 131
    - PADLOCK\_ALGORITHM\_TYPE\_AES, 130
    - padlock\_cbc, 131
    - padlock\_cipher\_alloc, 131
    - padlock\_cipher\_key\_setup, 131
    - padlock\_cipher\_process, 132
    - padlock\_cipher\_setup, 132
    - PADLOCK\_DIRECTION\_DECRYPT, 130
    - PADLOCK\_DIRECTION\_ENCRYPT, 130
    - PADLOCK\_KEY\_GENERATION\_HW, 130
    - PADLOCK\_KEY\_GENERATION\_SW, 130
    - PADLOCK\_KEY\_SIZE\_128, 130
    - PADLOCK\_KEY\_SIZE\_192, 130
    - PADLOCK\_KEY\_SIZE\_256, 131
    - PADLOCK\_ROUND\_COUNT\_AES128, 131
    - PADLOCK\_ROUND\_COUNT\_AES192, 131
    - PADLOCK\_ROUND\_COUNT\_AES256, 131
  - padlock\_cipher\_alloc
    - padlock\_cipher.c, 131
  - padlock\_cipher\_key\_setup
    - padlock\_cipher.c, 131
  - padlock\_cipher\_process
    - padlock.h, 127
    - padlock\_cipher.c, 132
  - padlock\_cipher\_setup
    - padlock.h, 127
    - padlock\_cipher.c, 132
  - padlock\_copy\_ctx
    - padlock\_hash.c, 134
  - padlock\_cw, 27
    - \_\_field, 27
    - algorithm\_type, 27
    - direction, 27
    - filler0, 27
    - filler1, 27
    - filler2, 27
    - filler3, 27
    - intermediate, 28

- key\_generation, 28
- key\_size, 28
- raw, 28
- round\_count, 28
- padlock\_destroy
  - padlock.c, 122
- PADLOCK\_DIRECTION\_DECRYPT
  - padlock\_cipher.c, 130
- PADLOCK\_DIRECTION\_ENCRYPT
  - padlock\_cipher.c, 130
- padlock\_do\_sha1
  - padlock\_hash.c, 135
- padlock\_do\_sha256
  - padlock\_hash.c, 135
- padlock\_free\_ctx
  - padlock\_hash.c, 135
- padlock\_freesession
  - padlock.c, 122
- padlock\_hash.c
  - \_\_FBSDID, 134
  - CTASSERT, 134
  - MALLOC\_DECLARE, 134
  - padlock\_authcompute, 134
  - padlock\_copy\_ctx, 134
  - padlock\_do\_sha1, 135
  - padlock\_do\_sha256, 135
  - padlock\_free\_ctx, 135
  - padlock\_hash\_free, 135
  - padlock\_hash\_key\_setup, 136
  - padlock\_hash\_process, 136
  - padlock\_hash\_setup, 136
  - padlock\_hmac\_sha1, 138
  - padlock\_hmac\_sha256, 138
  - padlock\_output\_block, 136
  - padlock\_sha1\_final, 137
  - padlock\_sha256\_final, 137
  - padlock\_sha\_free, 137
  - padlock\_sha\_init, 137
  - padlock\_sha\_update, 137
- padlock\_hash\_free
  - padlock.h, 127
  - padlock\_hash.c, 135
- padlock\_hash\_key\_setup
  - padlock\_hash.c, 136
- padlock\_hash\_process
  - padlock.h, 127
  - padlock\_hash.c, 136
- padlock\_hash\_setup
  - padlock.h, 128
  - padlock\_hash.c, 136
- padlock\_hmac\_sha1
  - padlock\_hash.c, 138
- padlock\_hmac\_sha256
  - padlock\_hash.c, 138
- padlock\_init
  - padlock.c, 122
- PADLOCK\_KEY\_GENERATION\_HW
  - padlock\_cipher.c, 130
- PADLOCK\_KEY\_GENERATION\_SW
  - padlock\_cipher.c, 130
- PADLOCK\_KEY\_SIZE\_128
  - padlock\_cipher.c, 130
- PADLOCK\_KEY\_SIZE\_192
  - padlock\_cipher.c, 130
- PADLOCK\_KEY\_SIZE\_256
  - padlock\_cipher.c, 131
- padlock\_mod
  - padlock.c, 124
- padlock\_modevent
  - padlock.c, 123
- padlock\_newsession
  - padlock.c, 123
- padlock\_output\_block
  - padlock\_hash.c, 136
- padlock\_process
  - padlock.c, 123
- PADLOCK\_ROUND\_COUNT\_AES128
  - padlock\_cipher.c, 131
- PADLOCK\_ROUND\_COUNT\_AES192
  - padlock\_cipher.c, 131
- PADLOCK\_ROUND\_COUNT\_AES256
  - padlock\_cipher.c, 131
- padlock\_sc
  - padlock.c, 124
- padlock\_session, 29
  - \_\_aligned, 29
  - ses\_axf, 29
  - ses\_ictx, 29
  - ses\_id, 29
  - ses\_mlen, 30
  - ses\_octx, 30
  - ses\_used, 30
  - TAILQ\_ENTRY, 29
- padlock\_sha1\_final
  - padlock\_hash.c, 137
- padlock\_sha256\_final
  - padlock\_hash.c, 137
- padlock\_sha\_ctx, 31
  - psc\_buf, 31
  - psc\_offset, 31
  - psc\_size, 31
- padlock\_sha\_free
  - padlock\_hash.c, 137
- padlock\_sha\_init
  - padlock\_hash.c, 137
- padlock\_sha\_update
  - padlock\_hash.c, 137
- padlock\_softc, 32

- sc\_cid, 32
- sc\_sid, 32
- perm
  - rc4\_state, 33
- PERM\_OP
  - des\_locl.h, 64
- podd.h
  - odd\_parity, 69
- psc\_buf
  - padlock\_sha\_ctx, 31
- psc\_offset
  - padlock\_sha\_ctx, 31
- psc\_size
  - padlock\_sha\_ctx, 31
- PUTBYTE
  - sha1.c, 100
- PUTPAD
  - sha1.c, 101
- PUTU32
  - rijndael-alg-fst.c, 77
- R
  - sha2.c, 107
- raw
  - padlock\_cw, 28
- rc4.c
  - \_\_FBSDDID, 73
  - DECLARE\_MODULE, 73
  - MODULE\_VERSION, 73
  - rc4\_crypt, 73
  - rc4\_init, 73
  - rc4\_mod, 73
  - rc4\_modevent, 73
  - swap\_bytes, 73
- rc4.h
  - rc4\_crypt, 75
  - rc4\_init, 75
- rc4\_crypt
  - rc4.c, 73
  - rc4.h, 75
- rc4\_init
  - rc4.c, 73
  - rc4.h, 75
- rc4\_mod
  - rc4.c, 73
- rc4\_modevent
  - rc4.c, 73
- rc4\_state, 33
  - index1, 33
  - index2, 33
  - perm, 33
- rcon
  - rijndael-alg-fst.c, 78
- REVERSE32
  - sha2.c, 107
- REVERSE64
  - sha2.c, 107
- rijndael-alg-fst.c
  - \_\_FBSDDID, 77
  - GETU32, 77
  - PUTU32, 77
  - rcon, 78
  - rijndaelDecrypt, 77
  - rijndaelEncrypt, 78
  - rijndaelKeySetupDec, 78
  - rijndaelKeySetupEnc, 78
  - SWAP, 77
  - Td0, 79
  - Td1, 79
  - Td2, 79
  - Td3, 79
  - Td4, 79
  - Te0, 79
  - Te1, 79
  - Te2, 79
  - Te3, 80
  - Te4, 80
- rijndael-api-fst.c
  - \_\_FBSDDID, 82
  - BYTE, 82
  - rijndael\_blockDecrypt, 82
  - rijndael\_blockEncrypt, 82
  - rijndael\_cipherInit, 82
  - rijndael\_makeKey, 82
  - rijndael\_padDecrypt, 83
  - rijndael\_padEncrypt, 83
  - TRUE, 81
- rijndael-api-fst.h
  - BAD\_BLOCK\_LENGTH, 86
  - BAD\_CIPHER\_INSTANCE, 86
  - BAD\_CIPHER\_MODE, 86
  - BAD\_CIPHER\_STATE, 86
  - BAD\_DATA, 86
  - BAD\_KEY\_DIR, 86
  - BAD\_KEY\_INSTANCE, 86
  - BAD\_KEY\_MAT, 86
  - BAD\_OTHER, 86
  - BITSPERBLOCK, 86
  - DIR\_DECRYPT, 87
  - DIR\_ENCRYPT, 87
  - MODE\_CBC, 87
  - MODE\_CFB1, 87
  - MODE\_ECB, 87
  - rijndael\_blockDecrypt, 87
  - rijndael\_blockEncrypt, 88
  - rijndael\_cipherInit, 88
  - rijndael\_makeKey, 88
  - RIJNDAEL\_MAX\_IV\_SIZE, 87



- RIJNDAEL\_MAX\_KEY\_SIZE, 87
- rijndael\_padDecrypt, 88
- rijndael\_padEncrypt, 89
- rijndael-api.c
  - \_\_FBSDDID, 90
  - rijndael\_decrypt, 90
  - rijndael\_encrypt, 90
  - rijndael\_set\_key, 91
- rijndael.h
  - rijndael\_decrypt, 93
  - rijndael\_encrypt, 93
  - RIJNDAEL\_MAXKB, 92
  - RIJNDAEL\_MAXKC, 92
  - RIJNDAEL\_MAXNR, 93
  - rijndael\_set\_key, 93
  - rijndaelDecrypt, 94
  - rijndaelEncrypt, 94
  - rijndaelKeySetupDec, 94
  - rijndaelKeySetupEnc, 94
- rijndael\_blockDecrypt
  - rijndael-api-fst.c, 82
  - rijndael-api-fst.h, 87
- rijndael\_blockEncrypt
  - rijndael-api-fst.c, 82
  - rijndael-api-fst.h, 88
- rijndael\_cipherInit
  - rijndael-api-fst.c, 82
  - rijndael-api-fst.h, 88
- rijndael\_ctx, 34
  - decrypt, 34
  - dk, 34
  - ek, 34
  - Nr, 34
- rijndael\_decrypt
  - rijndael-api.c, 90
  - rijndael.h, 93
- rijndael\_encrypt
  - rijndael-api.c, 90
  - rijndael.h, 93
- rijndael\_local.h
  - u16, 96
  - u32, 96
  - u8, 96
- rijndael\_makeKey
  - rijndael-api-fst.c, 82
  - rijndael-api-fst.h, 88
- RIJNDAEL\_MAX\_IV\_SIZE
  - rijndael-api-fst.h, 87
- RIJNDAEL\_MAX\_KEY\_SIZE
  - rijndael-api-fst.h, 87
- RIJNDAEL\_MAXKB
  - rijndael.h, 92
- RIJNDAEL\_MAXKC
  - rijndael.h, 92
- RIJNDAEL\_MAXNR
  - rijndael.h, 93
- rijndael\_padDecrypt
  - rijndael-api-fst.c, 83
  - rijndael-api-fst.h, 88
- rijndael\_padEncrypt
  - rijndael-api-fst.c, 83
  - rijndael-api-fst.h, 89
- rijndael\_set\_key
  - rijndael-api.c, 91
  - rijndael.h, 93
- rijndaelDecrypt
  - rijndael-alg-fst.c, 77
  - rijndael.h, 94
- rijndaelEncrypt
  - rijndael-alg-fst.c, 78
  - rijndael.h, 94
- rijndaelKeySetupDec
  - rijndael-alg-fst.c, 78
  - rijndael.h, 94
- rijndaelKeySetupEnc
  - rijndael-alg-fst.c, 78
  - rijndael.h, 94
- rk
  - keyInstance, 25
- ROTATE
  - des\_locl.h, 64
- round\_count
  - padlock\_cw, 28
- S
  - bf\_key\_st, 22
  - sha1.c, 101
- S32
  - sha2.c, 108
- S64
  - sha2.c, 108
- sc\_cid
  - padlock\_softc, 32
- sc\_sid
  - padlock\_softc, 32
- ses\_axf
  - padlock\_session, 29
- ses\_ictx
  - padlock\_session, 29
- ses\_id
  - padlock\_session, 29
- ses\_mlen
  - padlock\_session, 30
- ses\_octx
  - padlock\_session, 30
- ses\_used
  - padlock\_session, 30
- sha1.c

- [\\_K](#), 102
- [\\_\\_FBSDID](#), 101
- [BCOUNT](#), 100
- [COUNT](#), 100
- [F0](#), 100
- [F1](#), 100
- [F2](#), 100
- [F3](#), 100
- [H](#), 100
- [K](#), 100
- [PUTBYTE](#), 100
- [PUTPAD](#), 101
- [S](#), 101
- [sha1\\_init](#), 101
- [sha1\\_loop](#), 101
- [sha1\\_pad](#), 102
- [sha1\\_result](#), 102
- [sha1\\_step](#), 102
- [W](#), 101
- [sha1.h](#)
  - [SHA1\\_CTX](#), 104
  - [sha1\\_init](#), 104
  - [sha1\\_loop](#), 104
  - [sha1\\_pad](#), 104
  - [sha1\\_result](#), 104
  - [SHA1\\_RESULTLEN](#), 103
  - [SHA1Final](#), 103
  - [SHA1Init](#), 103
  - [SHA1Update](#), 103
- [SHA1\\_CTX](#)
  - [sha1.h](#), 104
- [sha1\\_ctxt](#), 35
  - [b32](#), 35
  - [b64](#), 35
  - [b8](#), 35
  - [c](#), 35
  - [count](#), 35
  - [h](#), 35
  - [m](#), 36
- [sha1\\_init](#)
  - [sha1.c](#), 101
  - [sha1.h](#), 104
- [sha1\\_loop](#)
  - [sha1.c](#), 101
  - [sha1.h](#), 104
- [sha1\\_pad](#)
  - [sha1.c](#), 102
  - [sha1.h](#), 104
- [sha1\\_result](#)
  - [sha1.c](#), 102
  - [sha1.h](#), 104
- [SHA1\\_RESULTLEN](#)
  - [sha1.h](#), 103
- [sha1\\_step](#)
  - [sha1.c](#), 102
- [SHA1Final](#)
  - [sha1.h](#), 103
- [SHA1Init](#)
  - [sha1.h](#), 103
- [SHA1Update](#)
  - [sha1.h](#), 103
- [sha2.c](#)
  - [\\_\\_FBSDID](#), 109
  - [ADDINC128](#), 106
  - [assert](#), 107
  - [Ch](#), 107
  - [K256](#), 114
  - [K512](#), 114
  - [Maj](#), 107
  - [R](#), 107
  - [REVERSE32](#), 107
  - [REVERSE64](#), 107
  - [S32](#), 108
  - [S64](#), 108
  - [SHA256\\_Data](#), 109
  - [SHA256\\_End](#), 110
  - [SHA256\\_Final](#), 110
  - [SHA256\\_Init](#), 110
  - [sha256\\_initial\\_hash\\_value](#), 114
  - [SHA256\\_SHORT\\_BLOCK\\_LENGTH](#), 108
  - [SHA256\\_Transform](#), 110
  - [SHA256\\_Update](#), 110
  - [sha2\\_byte](#), 109
  - [sha2\\_hex\\_digits](#), 114
  - [sha2\\_word32](#), 109
  - [sha2\\_word64](#), 109
  - [SHA384\\_Data](#), 111
  - [SHA384\\_End](#), 111
  - [SHA384\\_Final](#), 111
  - [SHA384\\_Init](#), 112
  - [sha384\\_initial\\_hash\\_value](#), 115
  - [SHA384\\_SHORT\\_BLOCK\\_LENGTH](#), 108
  - [SHA384\\_Update](#), 112
  - [SHA512\\_Data](#), 112
  - [SHA512\\_End](#), 112
  - [SHA512\\_Final](#), 112
  - [SHA512\\_Init](#), 113
  - [sha512\\_initial\\_hash\\_value](#), 115
  - [SHA512\\_Last](#), 113
  - [SHA512\\_SHORT\\_BLOCK\\_LENGTH](#), 108
  - [SHA512\\_Transform](#), 113
  - [SHA512\\_Update](#), 113
  - [Sigma0\\_256](#), 108
  - [sigma0\\_256](#), 108
  - [Sigma0\\_512](#), 108
  - [sigma0\\_512](#), 108
  - [Sigma1\\_256](#), 109
  - [sigma1\\_256](#), 108

- Sigma1\_512, 109
- sigma1\_512, 109
- sha2.h
  - SHA256\_BLOCK\_LENGTH, 117
  - SHA256\_CTX, 118
  - SHA256\_Data, 118
  - SHA256\_DIGEST\_LENGTH, 117
  - SHA256\_DIGEST\_STRING\_LENGTH, 117
  - SHA256\_End, 118
  - SHA256\_Final, 118
  - SHA256\_Init, 118
  - SHA256\_Update, 118
  - SHA384\_BLOCK\_LENGTH, 117
  - SHA384\_CTX, 118
  - SHA384\_Data, 119
  - SHA384\_DIGEST\_LENGTH, 117
  - SHA384\_DIGEST\_STRING\_LENGTH, 117
  - SHA384\_End, 119
  - SHA384\_Final, 119
  - SHA384\_Init, 119
  - SHA384\_Update, 119
  - SHA512\_BLOCK\_LENGTH, 117
  - SHA512\_CTX, 118
  - SHA512\_Data, 119
  - SHA512\_DIGEST\_LENGTH, 117
  - SHA512\_DIGEST\_STRING\_LENGTH, 117
  - SHA512\_End, 120
  - SHA512\_Final, 120
  - SHA512\_Init, 120
  - SHA512\_Update, 120
- SHA256\_BLOCK\_LENGTH
  - sha2.h, 117
- SHA256\_CTX
  - sha2.h, 118
- SHA256\_Data
  - sha2.c, 109
  - sha2.h, 118
- SHA256\_DIGEST\_LENGTH
  - sha2.h, 117
- SHA256\_DIGEST\_STRING\_LENGTH
  - sha2.h, 117
- SHA256\_End
  - sha2.c, 110
  - sha2.h, 118
- SHA256\_Final
  - sha2.c, 110
  - sha2.h, 118
- SHA256\_Init
  - sha2.c, 110
  - sha2.h, 118
- sha256\_initial\_hash\_value
  - sha2.c, 114
- SHA256\_SHORT\_BLOCK\_LENGTH
  - sha2.c, 108
- SHA256\_Transform
  - sha2.c, 110
- SHA256\_Update
  - sha2.c, 110
  - sha2.h, 118
- sha2\_byte
  - sha2.c, 109
- sha2\_hex\_digits
  - sha2.c, 114
- sha2\_word32
  - sha2.c, 109
- sha2\_word64
  - sha2.c, 109
- SHA384\_BLOCK\_LENGTH
  - sha2.h, 117
- SHA384\_CTX
  - sha2.h, 118
- SHA384\_Data
  - sha2.c, 111
  - sha2.h, 119
- SHA384\_DIGEST\_LENGTH
  - sha2.h, 117
- SHA384\_DIGEST\_STRING\_LENGTH
  - sha2.h, 117
- SHA384\_End
  - sha2.c, 111
  - sha2.h, 119
- SHA384\_Final
  - sha2.c, 111
  - sha2.h, 119
- SHA384\_Init
  - sha2.c, 112
  - sha2.h, 119
- sha384\_initial\_hash\_value
  - sha2.c, 115
- SHA384\_SHORT\_BLOCK\_LENGTH
  - sha2.c, 108
- SHA384\_Update
  - sha2.c, 112
  - sha2.h, 119
- SHA512\_BLOCK\_LENGTH
  - sha2.h, 117
- SHA512\_CTX
  - sha2.h, 118
- SHA512\_Data
  - sha2.c, 112
  - sha2.h, 119
- SHA512\_DIGEST\_LENGTH
  - sha2.h, 117
- SHA512\_DIGEST\_STRING\_LENGTH
  - sha2.h, 117
- SHA512\_End
  - sha2.c, 112
  - sha2.h, 120

- SHA512\_Final
  - sha2.c, [112](#)
  - sha2.h, [120](#)
- SHA512\_Init
  - sha2.c, [113](#)
  - sha2.h, [120](#)
- sha512\_initial\_hash\_value
  - sha2.c, [115](#)
- SHA512\_Last
  - sha2.c, [113](#)
- SHA512\_SHORT\_BLOCK\_LENGTH
  - sha2.c, [108](#)
- SHA512\_Transform
  - sha2.c, [113](#)
- SHA512\_Update
  - sha2.c, [113](#)
  - sha2.h, [120](#)
- Sigma0\_256
  - sha2.c, [108](#)
- sigma0\_256
  - sha2.c, [108](#)
- Sigma0\_512
  - sha2.c, [108](#)
- sigma0\_512
  - sha2.c, [108](#)
- Sigma1\_256
  - sha2.c, [109](#)
- sigma1\_256
  - sha2.c, [108](#)
- Sigma1\_512
  - sha2.c, [109](#)
- sigma1\_512
  - sha2.c, [109](#)
- sk.h
  - des\_skb, [70](#)
- spr.h
  - des\_SPtrans, [71](#)
- state
  - \_SHA256\_CTX, [19](#)
  - \_SHA512\_CTX, [21](#)
- SWAP
  - rijndael-alg-fst.c, [77](#)
- swap\_bytes
  - rc4.c, [73](#)
- TAILQ\_ENTRY
  - padlock\_session, [29](#)
- Td0
  - rijndael-alg-fst.c, [79](#)
- Td1
  - rijndael-alg-fst.c, [79](#)
- Td2
  - rijndael-alg-fst.c, [79](#)
- Td3
  - rijndael-alg-fst.c, [79](#)
- Td4
  - rijndael-alg-fst.c, [79](#)
- Te0
  - rijndael-alg-fst.c, [79](#)
- Te1
  - rijndael-alg-fst.c, [79](#)
- Te2
  - rijndael-alg-fst.c, [79](#)
- Te3
  - rijndael-alg-fst.c, [80](#)
- Te4
  - rijndael-alg-fst.c, [80](#)
- test00.c
  - LL, [97](#)
  - main, [97](#)
- TRUE
  - rijndael-api-fst.c, [81](#)
- u16
  - rijndael\_local.h, [96](#)
- u32
  - rijndael\_local.h, [96](#)
- u8
  - rijndael\_local.h, [96](#)
- W
  - sha1.c, [101](#)
- weak\_key
  - des\_ks\_struct, [24](#)
- weak\_keys
  - des\_setkey.c, [68](#)