# SEcube

Generated by Doxygen 1.8.11

# Contents

**1 Module Index** 1

1.1 Modules . . . . . . . . . . 1

**2 Data Structure Index** 3

2.1 Data Structures . . . . . . . . . . 3

**3 File Index** 5

3.1 File List . . . . . . . . . . 5

**4 Module Documentation** 7

4.1 AES return values . . . . . . . . . . 7

4.1.1 Detailed Description . . . . . . . . . . 7

4.2 AES Key, IV, Block Sizes . . . . . . . . . . 8

4.2.1 Detailed Description . . . . . . . . . . 8

4.2.2 Macro Definition Documentation . . . . . . . . . . 8

4.2.2.1 B5_AES_128 . . . . . . . . . . 8

4.2.2.2 B5_AES_192 . . . . . . . . . . 8

4.2.2.3 B5_AES_256 . . . . . . . . . . 8

4.2.2.4 B5_AES_BLK_SIZE . . . . . . . . . . 8

4.2.2.5 B5_AES_IV_SIZE . . . . . . . . . . 8

4.3 AES modes . . . . . . . . . . 9

4.3.1 Detailed Description . . . . . . . . . . 9

4.3.2 Macro Definition Documentation . . . . . . . . . . 9

4.3.2.1 B5_AES256_CBC_DEC . . . . . . . . . . 9

4.3.2.2 B5_AES256_CBC_ENC . . . . . . . . . . 9

4.3.2.3 B5_AES256_CFB_DEC . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
4.3.2.4 B5_AES256_CFB_ENC . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
4.3.2.5 B5_AES256_CTR . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
4.3.2.6 B5_AES256_ECB_DEC . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
4.3.2.7 B5_AES256_ECB_ENC . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
4.3.2.8 B5_AES256_OFB . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
4.4 AES data structures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
4.4.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
4.5 AES functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
4.5.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
4.5.2 Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 11
4.5.2.1 B5_Aes256_Finit(B5_tAesCtx ∗ctx) . . . . . . . . . . . . . . . . . . . . . 11
4.5.2.2 B5_Aes256_Init(B5_tAesCtx ∗ctx, const uint8_t ∗Key, int16_t keySize, uint8_t aesMode) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
4.5.2.3 B5_Aes256_SetIV(B5_tAesCtx ∗ctx, const uint8_t ∗IV) . . . . . . . . . . . . . 12
4.5.2.4 B5_Aes256_Update(B5_tAesCtx ∗ctx, uint8_t ∗encData, uint8_t ∗clrData, int16_t nBlk) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
4.6 CMAC-AES Key, Blk Sizes . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
4.6.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
4.6.2 Macro Definition Documentation . . . . . . . . . . . . . . . . . . . . . . 14
4.6.2.1 B5_CMAC_AES_128 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
4.6.2.2 B5_CMAC_AES_192 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
4.6.2.3 B5_CMAC_AES_256 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
4.6.2.4 B5_CMAC_AES_BLK_SIZE . . . . . . . . . . . . . . . . . . . . . . . . . . 14
4.7 CMAC-AES return values . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
4.7.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
4.8 CMAC-AES data structures . . . . . . . . . . . . . . . . . . . . . . . . . . 16
4.8.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
4.9 CMAC-AES functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
4.9.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
4.9.2 Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 17

4.9.2.1 B5_CmacAes256_Finit(B5_tCmacAesCtx ∗ctx, uint8_t ∗rSignature) . . . . . . . 17
4.9.2.2 B5_CmacAes256_Init(B5_tCmacAesCtx ∗ctx, const uint8_t ∗Key, int16_t keySize) 17
4.9.2.3 B5_CmacAes256_Reset(B5_tCmacAesCtx ∗ctx) . . . . . . . . . . . . . . . . 18
4.9.2.4 B5_CmacAes256_Sign(const uint8_t ∗data, int32_t dataLen, const uint8_t ∗Key, int16_t keySize, uint8_t ∗rSignature) . . . . . . . . . . . . . . . . . . . . . . . 18
4.9.2.5 B5_CmacAes256_Update(B5_tCmacAesCtx ∗ctx, const uint8_t ∗data, int32_t dataLen) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
4.10 AccessLogin . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
4.10.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
4.11 KeyOpEdit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
4.11.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
4.11.2 Enumeration Type Documentation . . . . . . . . . . . . . . . . . . . . . . . 20
4.11.2.1 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
4.12 AlgorithmAvail . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
4.12.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
4.12.2 Enumeration Type Documentation . . . . . . . . . . . . . . . . . . . . . . . 21
4.12.2.1 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
4.13 SHA256 return values . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
4.13.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
4.14 SHA256 digest and block sizes . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
4.14.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
4.15 SHA256 data structures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
4.15.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
4.16 SHA256 functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
4.16.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
4.16.2 Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
4.16.2.1 B5_Sha256_Finit(B5_tSha256Ctx ∗ctx, uint8_t ∗rDigest) . . . . . . . . . . . . 25
4.16.2.2 B5_Sha256_Init(B5_tSha256Ctx ∗ctx) . . . . . . . . . . . . . . . . . . . . 25
4.16.2.3 B5_Sha256_Update(B5_tSha256Ctx ∗ctx, const uint8_t ∗data, int32_t dataLen) 25
4.17 HMAC-SHA256 return values . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
4.17.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
4.18 HMAC-SHA256 data structures . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
4.18.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
4.19 HMAC-SHA256 functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
4.19.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
4.19.2 Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
4.19.2.1 B5_HmacSha256_Finit(B5_tHmacSha256Ctx ∗ctx, uint8_t ∗rDigest) . . . . . . 29
4.19.2.2 B5_HmacSha256_Init(B5_tHmacSha256Ctx ∗ctx, const uint8_t ∗Key, int16_t keySize) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
4.19.2.3 B5_HmacSha256_Update(B5_tHmacSha256Ctx ∗ctx, const uint8_t ∗data, int32_t dataLen) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29

5 Data Structure Documentation 31

5.1 B5_tAesCtx Struct Reference . . . 31

5.1.1 Field Documentation . . . 31

5.1.1.1 InitVector . . . 31

5.1.1.2 mode . . . 31

5.1.1.3 Nr . . . 31

5.1.1.4 rk . . . 32

5.2 B5_tCmacAesCtx Struct Reference . . . 32

5.3 B5_tHmacSha256Ctx Struct Reference . . . 32

5.4 B5_tSha256Ctx Struct Reference . . . 32

5.5 se3_algo_ Struct Reference . . . 33

5.5.1 Detailed Description . . . 33

5.6 se3_device_ Struct Reference . . . 33

5.6.1 Detailed Description . . . 33

5.7 se3_device_info_ Struct Reference . . . 34

5.7.1 Detailed Description . . . 34

5.8 se3_disco_it_ Struct Reference . . . 34

5.8.1 Detailed Description . . . 34

5.9 se3_discover_info_ Struct Reference . . . 35

5.10 se3_drive_it_ Struct Reference . . . 35

5.11 se3_file Struct Reference . . . 35

5.12 se3_key_ Struct Reference . . . 35

5.12.1 Detailed Description . . . 36

5.13 se3_payload_cryptoctx_ Struct Reference . . . 36

5.14 se3_session_ Struct Reference . . . 36

5.14.1 Detailed Description . . . 36

## 6 File Documentation 37

6.1 src/Common/crc16.h File Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
6.1.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
6.1.2 Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
6.1.2.1 se3_crc16_update(size_t length, const uint8_t ∗data, uint16_t crc) . . . . . . . . 37
6.2 src/Common/se3_common.h File Reference . . . . . . . . . . . . . . . . . . . . . . . . . . 38
6.2.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
6.2.2 Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
6.2.2.1 se3_nblocks(uint16_t len) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
6.2.2.2 se3_req_len_data(uint16_t len_data_and_headers) . . . . . . . . . . . . . . . . 39
6.2.2.3 se3_req_len_data_and_headers(uint16_t len_data) . . . . . . . . . . . . . . . . 39
6.2.2.4 se3_resp_len_data(uint16_t len_data_and_headers) . . . . . . . . . . . . . . . 39
6.2.2.5 se3_resp_len_data_and_headers(uint16_t len_data) . . . . . . . . . . . . . . . 40
6.3 src/Common/se3c1def.h File Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
6.3.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
6.3.2 Enumeration Type Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
6.3.2.1 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
6.3.2.2 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
6.3.2.3 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
6.3.2.4 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
6.3.2.5 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
6.3.2.6 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
6.3.2.7 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
6.3.2.8 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
6.3.2.9 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
6.3.2.10 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
6.3.2.11 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
6.3.2.12 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
6.3.2.13 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
6.3.2.14 anonymous enum . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43

6.3.2.15 anonymous enum . . . . . . . . . . 43
6.3.2.16 anonymous enum . . . . . . . . . . 44
6.3.2.17 anonymous enum . . . . . . . . . . 44
6.3.2.18 anonymous enum . . . . . . . . . . 44
6.3.2.19 anonymous enum . . . . . . . . . . 44
6.3.2.20 anonymous enum . . . . . . . . . . 44
6.3.2.21 anonymous enum . . . . . . . . . . 44
6.3.2.22 anonymous enum . . . . . . . . . . 44
6.3.2.23 anonymous enum . . . . . . . . . . 44
6.4 src/Host/L0.h File Reference . . . . . . . . . . 45
6.4.1 Detailed Description . . . . . . . . . . 46
6.4.2 Function Documentation . . . . . . . . . . 46
6.4.2.1 L0_close(se3_device ∗dev) . . . . . . . . . . 46
6.4.2.2 L0_discover_init(se3_disco_it ∗it) . . . . . . . . . . 46
6.4.2.3 L0_discover_next(se3_disco_it ∗it) . . . . . . . . . . 46
6.4.2.4 L0_discover_serialno(uint8_t ∗serialno, se3_device_info ∗device) . . . . . . . . 47
6.4.2.5 L0_echo(se3_device ∗device, const uint8_t ∗data_in, uint16_t data_in_len, uint8_t ∗data_out) . . . . . . . . . . 47
6.4.2.6 L0_factoryinit(se3_device ∗device, const uint8_t ∗serialno) . . . . . . . . . . 47
6.4.2.7 L0_open(se3_device ∗dev, se3_device_info ∗dev_info, uint32_t timeout) . . . . . 47
6.4.2.8 L0_TXRX(se3_device ∗device, uint16_t req_cmd, uint16_t req_cmdflags, uint16_t req_len, const uint8_t ∗req_data, uint16_t ∗resp_status, uint16_t ∗resp_len, uint8_t ∗resp_data) . . . . . . . . . . 48
6.5 src/Host/L1.h File Reference . . . . . . . . . . 48
6.5.1 Detailed Description . . . . . . . . . . 50
6.5.2 Function Documentation . . . . . . . . . . 50
6.5.2.1 L1_crypto_init(se3_session ∗s, uint16_t algorithm, uint16_t mode, uint32_t key_id, uint32_t ∗sess_id) . . . . . . . . . . 50
6.5.2.2 L1_crypto_set_time(se3_session ∗s, uint32_t devtime) . . . . . . . . . . 50
6.5.2.3 L1_crypto_update(se3_session ∗s, uint32_t sess_id, uint16_t flags, uint16_t data1_len, uint8_t ∗data1, uint16_t data2_len, uint8_t ∗data2, uint16_t ∗dataout_len, uint8_t ∗data_out) . . . . . . . . . . 51

6.5.2.4 L1_decrypt(se3_session *s, uint16_t algorithm, uint16_t mode, uint32_t key_id, size_t datain_len, int8_t *data_in, size_t *dataout_len, uint8_t *data_out) . . . . 51

6.5.2.5 L1_digest(se3_session *s, uint16_t algorithm, size_t datain_len, int8_t *data_in, size_t *dataout_len, uint8_t *data_out) . . . . . . . . . . . . . . . . . . . . . . 52

6.5.2.6 L1_encrypt(se3_session *s, uint16_t algorithm, uint16_t mode, uint32_t key_id, size_t datain_len, int8_t *data_in, size_t *dataout_len, uint8_t *data_out) . . . . 52

6.5.2.7 L1_find_key(se3_session *s, uint32_t key_id) . . . . . . . . . . . . . . . . . . . 52

6.5.2.8 L1_get_algorithms(se3_session *s, uint16_t skip, uint16_t max_algorithms, se3_algo *algorithms_array, uint16_t *count) . . . . . . . . . . . . . . . . . . . 53

6.5.2.9 L1_key_edit(se3_session *s, uint16_t op, se3_key *k) . . . . . . . . . . . . . . 53

6.5.2.10 L1_key_list(se3_session *s, uint16_t skip, uint16_t max_keys, se3_key *key_array, uint16_t *count) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53

6.5.2.11 L1_login(se3_session *s, se3_device *dev, const uint8_t *pin, uint16_t access) . 54

6.5.2.12 L1_logout(se3_session *s) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54

6.5.2.13 L1_set_admin_PIN(se3_session *s, uint8_t *pin) . . . . . . . . . . . . . . . . . 54

6.5.2.14 L1_set_user_PIN(se3_session *s, uint8_t *pin) . . . . . . . . . . . . . . . . . . 55

**Index** **57**

# Chapter 1

# Module Index

## 1.1 Modules

Here is a list of all modules:

AES return values . . . . . . . . . 7
AES Key, IV, Block Sizes . . . . . . . . . 8
AES modes . . . . . . . . . 9
AES data structures . . . . . . . . . 10
AES functions . . . . . . . . . 11
CMAC-AES Key, Blk Sizes . . . . . . . . . 14
CMAC-AES return values . . . . . . . . . 15
CMAC-AES data structures . . . . . . . . . 16
CMAC-AES functions . . . . . . . . . 17
AccessLogin . . . . . . . . . 19
KeyOpEdit . . . . . . . . . 20
AlgorithmAvail . . . . . . . . . 21
SHA256 return values . . . . . . . . . 22
SHA256 digest and block sizes . . . . . . . . . 23
SHA256 data structures . . . . . . . . . 24
SHA256 functions . . . . . . . . . 25
HMAC-SHA256 return values . . . . . . . . . 27
HMAC-SHA256 data structures . . . . . . . . . 28
HMAC-SHA256 functions . . . . . . . . . 29

# Chapter 2

# Data Structure Index

## 2.1 Data Structures

Here are the data structures with brief descriptions:

B5_tAesCtx . . . . . 31
B5_tCmacAesCtx . . . . . 32
B5_tHmacSha256Ctx . . . . . 32
B5_tSha256Ctx . . . . . 32
se3_algo_
    SEcube Algorithm structure . . . . . 33
se3_device_
    SEcube Device structure . . . . . 33
se3_device_info_
    SEcube Device Information structure . . . . . 34
se3_disco_it_
    Discovery iterator . . . . . 34
se3_discover_info_ . . . . . 35
se3_drive_it_ . . . . . 35
se3_file . . . . . 35
se3_key_
    SEcube Key structure . . . . . 35
se3_payload_cryptoctx_ . . . . . 36
se3_session_
    SEcube Communication session structure . . . . . 36

# Chapter 3

# File Index

## 3.1 File List

Here is a list of all documented files with brief descriptions:

src/Common/**aes256.h** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **??**
src/Common/crc16.h
This file contains defines and functions for computing CRC . . . . . . . . . . . . . . . . . . . . 37
src/Common/**pbkdf2.h** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **??**
src/Common/se3_common.h
This file contains defines and functions common for L0 and L1 . . . . . . . . . . . . . . . . . . 38
src/Common/**se3c0def.h** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **??**
src/Common/se3c1def.h
This file contains defines to be used both for L1 and L0 functions . . . . . . . . . . . . . . . . 40
src/Common/**sha256.h** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **??**
src/Host/L0.h
This file contains L0 functions and structures . . . . . . . . . . . . . . . . . . . . . . . . . . 45
src/Host/L1.h
This file contains L1 functions and structures . . . . . . . . . . . . . . . . . . . . . . . . . . 48
src/Host/**se3comm.h** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **??**

# Chapter 4

# Module Documentation

## 4.1 AES return values

### AES return values

- #define **B5_AES256_RES_OK** ( 0)
- #define **B5_AES256_RES_INVALID_CONTEXT** (-1)
- #define **B5_AES256_RES_CANNOT_ALLOCATE_CONTEXT** (-2)
- #define **B5_AES256_RES_INVALID_KEY_SIZE** (-3)
- #define **B5_AES256_RES_INVALID_ARGUMENT** (-4)
- #define **B5_AES256_RES_INVALID_MODE** (-5)

### 4.1.1 Detailed Description

## 4.2 AES Key, IV, Block Sizes

### AES Key, IV, Block Sizes

- #define B5_AES_256 32
- #define B5_AES_192 24
- #define B5_AES_128 16
- #define B5_AES_IV_SIZE 16
- #define B5_AES_BLK_SIZE 16

### 4.2.1 Detailed Description

### 4.2.2 Macro Definition Documentation

#### 4.2.2.1 #define B5_AES_128 16

Key Size in Bytes.

#### 4.2.2.2 #define B5_AES_192 24

Key Size in Bytes.

#### 4.2.2.3 #define B5_AES_256 32

Key Size in Bytes.

#### 4.2.2.4 #define B5_AES_BLK_SIZE 16

Block Size in Bytes.

#### 4.2.2.5 #define B5_AES_IV_SIZE 16

IV Size in Bytes.

## 4.3 AES modes

### AES modes

- #define B5_AES256_OFB 1
- #define B5_AES256_ECB_ENC 2
- #define B5_AES256_ECB_DEC 3
- #define B5_AES256_CBC_ENC 4
- #define B5_AES256_CBC_DEC 5
- #define B5_AES256_CFB_ENC 6
- #define B5_AES256_CFB_DEC 7
- #define B5_AES256_CTR 8

### 4.3.1 Detailed Description

### 4.3.2 Macro Definition Documentation

#### 4.3.2.1 #define B5_AES256_CBC_DEC 5

CBC decryption

#### 4.3.2.2 #define B5_AES256_CBC_ENC 4

CBC encryption

#### 4.3.2.3 #define B5_AES256_CFB_DEC 7

CFB decryption

#### 4.3.2.4 #define B5_AES256_CFB_ENC 6

CFB decryption

#### 4.3.2.5 #define B5_AES256_CTR 8

CTR counter mode encryption-decryption

#### 4.3.2.6 #define B5_AES256_ECB_DEC 3

ECB decryption

#### 4.3.2.7 #define B5_AES256_ECB_ENC 2

ECB encryption

#### 4.3.2.8 #define B5_AES256_OFB 1

OFB full feedback encryption-decryption

## 4.4 AES data structures

### Data Structures

- struct B5_tAesCtx

### 4.4.1 Detailed Description

## 4.5 AES functions

### AES functions

- int32_t B5_Aes256_Init (B5_tAesCtx ∗ctx, const uint8_t ∗Key, int16_t keySize, uint8_t aesMode)

  *Initialize the AES context.*
- int32_t B5_Aes256_SetIV (B5_tAesCtx ∗ctx, const uint8_t ∗IV)

  *Set the IV for the current AES context.*
- int32_t B5_Aes256_Update (B5_tAesCtx ∗ctx, uint8_t ∗encData, uint8_t ∗clrData, int16_t nBlk)

  *Encrypt/Decrypt data based on the status of current AES context.*
- int32_t B5_Aes256_Finit (B5_tAesCtx ∗ctx)

  *De-initialize the current AES context.*

### 4.5.1 Detailed Description

### 4.5.2 Function Documentation

#### 4.5.2.1 int32_t B5_Aes256_Finit ( **B5_tAesCtx** ∗ *ctx* )

De-initialize the current AES context.

**Parameters**

| | |
|---|---|
| *ctx* | Pointer to the AES context to de-initialize. |

**Returns**

See AES return values .

#### 4.5.2.2 int32_t B5_Aes256_Init ( **B5_tAesCtx** ∗ *ctx,* const uint8_t ∗ *Key,* int16_t *keySize,* uint8_t *aesMode* )

Initialize the AES context.

**Parameters**

| | |
|---|---|
| *ctx* | Pointer to the AES data structure to be initialized. |
| *Key* | Pointer to the Key that must be used for encryption/decryption. |
| *keySize* | Key size. See AES Key, IV, Block Sizes for supported sizes. |
| *aesMode* | AES mode. See AES modes for supported modes. |

**Returns**

See AES return values .

#### 4.5.2.3 int32_t B5_Aes256_SetIV ( B5_tAesCtx ∗ *ctx*, const uint8_t ∗ *IV* )

Set the IV for the current AES context.

**Parameters**

| | |
|---|---|
| *ctx* | Pointer to the AES data structure to be initialized. |
| *IV* | Pointer to the IV. |

**Returns**

See AES return values .

#### 4.5.2.4 int32_t B5_Aes256_Update ( B5_tAesCtx ∗ *ctx*, uint8_t ∗ *encData*, uint8_t ∗ *clrData*, int16_t *nBlk* )

Encrypt/Decrypt data based on the status of current AES context.

**Parameters**

| | |
|---|---|
| *ctx* | Pointer to the current AES context. |
| *encData* | Encrypted data. |
| *clrData* | Clear data. |
| *nBlk* | Number of AES blocks to process. |

**Returns**

See AES return values .

# 4.6 CMAC-AES Key, Blk Sizes

## CMAC-AES Key, Block Sizes

- #define B5_CMAC_AES_256 32
- #define B5_CMAC_AES_192 24
- #define B5_CMAC_AES_128 16
- #define B5_CMAC_AES_BLK_SIZE 16

## 4.6.1 Detailed Description

## 4.6.2 Macro Definition Documentation

### 4.6.2.1 #define B5_CMAC_AES_128 16

Key Size in Bytes

### 4.6.2.2 #define B5_CMAC_AES_192 24

Key Size in Bytes

### 4.6.2.3 #define B5_CMAC_AES_256 32

Key Size in Bytes

### 4.6.2.4 #define B5_CMAC_AES_BLK_SIZE 16

Block Size in Bytes

## 4.7 CMAC-AES return values

### CMAC-AES return values

- #define **B5_CMAC_AES256_RES_OK** ( 0)
- #define **B5_CMAC_AES256_RES_INVALID_CONTEXT** (-1)
- #define **B5_CMAC_AES256_RES_CANNOT_ALLOCATE_CONTEXT** (-2)
- #define **B5_CMAC_AES256_RES_INVALID_KEY_SIZE** (-3)
- #define **B5_CMAC_AES256_RES_INVALID_ARGUMENT** (-4)

### 4.7.1 Detailed Description

## 4.8 CMAC-AES data structures

### Data Structures

- struct B5_tCmacAesCtx

### 4.8.1 Detailed Description

## 4.9 CMAC-AES functions

### CMAC-AES functions

- int32_t B5_CmacAes256_Init (B5_tCmacAesCtx ∗ctx, const uint8_t ∗Key, int16_t keySize)
  *Initialize the CMAC-AES context.*
- int32_t B5_CmacAes256_Update (B5_tCmacAesCtx ∗ctx, const uint8_t ∗data, int32_t dataLen)
  *Compute the CMAC-AES algorithm on input data depending on the current status of the CMAC-AES context.*
- int32_t B5_CmacAes256_Finit (B5_tCmacAesCtx ∗ctx, uint8_t ∗rSignature)
  *De-initialize the current CMAC-AES context.*
- int32_t B5_CmacAes256_Reset (B5_tCmacAesCtx ∗ctx)
  *Reset the current CMAC-AES context.*
- int32_t B5_CmacAes256_Sign (const uint8_t ∗data, int32_t dataLen, const uint8_t ∗Key, int16_t keySize, uint8_t ∗rSignature)
  *Compute the signature through the CMAC-AES algorithm.*

### 4.9.1 Detailed Description

### 4.9.2 Function Documentation

#### 4.9.2.1 int32_t B5_CmacAes256_Finit ( **B5_tCmacAesCtx** ∗ *ctx*, uint8_t ∗ *rSignature* )

De-initialize the current CMAC-AES context.

**Parameters**

| | |
|---|---|
| *ctx* | Pointer to the CMAC-AES context to de-initialize. |
| *rSignature* | Pointer to a blank memory area that can store the computed output signature. |

**Returns**

See CMAC-AES return values .

#### 4.9.2.2 int32_t B5_CmacAes256_Init ( **B5_tCmacAesCtx** ∗ *ctx*, const uint8_t ∗ *Key*, int16_t *keySize* )

Initialize the CMAC-AES context.

**Parameters**

| | |
|---|---|
| *ctx* | Pointer to the CMAC-AES data structure to be initialized. |
| *Key* | Pointer to the Key that must be used. |
| *keySize* | Key size. See CMAC-AES Key, Blk Sizes for supported sizes. |

**Returns**

See CMAC-AES return values .

#### 4.9.2.3 int32_t B5_CmacAes256_Reset ( **B5_tCmacAesCtx** ∗ *ctx* )

Reset the current CMAC-AES context.

**Parameters**

| | |
|---|---|
| *ctx* | Pointer to the CMAC-AES context to reset. |

**Returns**

See CMAC-AES return values .

#### 4.9.2.4 int32_t B5_CmacAes256_Sign ( const uint8_t ∗ *data*, int32_t *dataLen*, const uint8_t ∗ *Key*, int16_t *keySize*, uint8_t ∗ *rSignature* )

Compute the signature through the CMAC-AES algorithm.

**Parameters**

| | |
|---|---|
| *data* | Pointer to the input data. |
| *dataLen* | Input data length (in Bytes). |
| *Key* | Pointer to the Key that must be used. |
| *keySize* | Key size. See CMAC-AES Key, Blk Sizes for supported sizes. |
| *rSignature* | Pointer to a blank memory area that can store the computed output signature. |

**Returns**

See CMAC-AES return values .

#### 4.9.2.5 int32_t B5_CmacAes256_Update ( **B5_tCmacAesCtx** ∗ *ctx*, const uint8_t ∗ *data*, int32_t *dataLen* )

Compute the CMAC-AES algorithm on input data depending on the current status of the CMAC-AES context.

**Parameters**

| | |
|---|---|
| *ctx* | Pointer to the current CMAC-AES context. |
| *data* | Pointer to the input data. |
| *dataLen* | Bytes to be processed. |

**Returns**

See CMAC-AES return values .

## 4.10 AccessLogin

Use this values as access parameter when using L1_login.

### Enumerations

- enum { **SE3_ACCESS_USER** = 100, **SE3_ACCESS_ADMIN** = 1000, **SE3_ACCESS_MAX** = 0xFFFF }

### 4.10.1 Detailed Description

Use this values as access parameter when using L1_login.

## 4.11 KeyOpEdit

Use these values when using L1_key_edit.

### Enumerations

- enum { SE3_KEY_OP_INSERT = 1, SE3_KEY_OP_DELETE = 2, SE3_KEY_OP_UPSERT = 3 }

### 4.11.1 Detailed Description

Use these values when using L1_key_edit.

### 4.11.2 Enumeration Type Documentation

#### 4.11.2.1 anonymous enum

**Enumerator**

| | |
|---|---|
| ***SE3_KEY_OP_INSERT*** | Use this value to insert a new key |
| ***SE3_KEY_OP_DELETE*** | Use this value to delete a new key |
| ***SE3_KEY_OP_UPSERT*** | Use this value to update/insert a key |

## 4.12 AlgorithmAvail

### Enumerations

- enum {
  SE3_ALGO_AES = 0, SE3_ALGO_SHA256 = 1, SE3_ALGO_HMACSHA256 = 2, SE3_ALGO_AES_HMACSHA256 = 3,
  SE3_ALGO_AES_HMAC = 4, **SE3_ALGO_MAX** = 8 }

### 4.12.1 Detailed Description

### 4.12.2 Enumeration Type Documentation

#### 4.12.2.1 anonymous enum

**Enumerator**

***SE3_ALGO_AES*** AES.

***SE3_ALGO_SHA256*** SHA256.

***SE3_ALGO_HMACSHA256*** HMAC-SHA256.

***SE3_ALGO_AES_HMACSHA256*** AES + HMAC-SHA256.

***SE3_ALGO_AES_HMAC*** AES 256 + HMAC Auth TODO remove.

## 4.13 SHA256 return values

### SHA256 return values

- #define **B5_SHA256_RES_OK** ( 0)
- #define **B5_SHA256_RES_INVALID_CONTEXT** (-1)
- #define **B5_SHA256_RES_CANNOT_ALLOCATE_CONTEXT** (-2)
- #define **B5_SHA256_RES_INVALID_ARGUMENT** (-3)
- #define **B5_HMAC_SHA256_RES_OK** ( 0)
- #define **B5_HMAC_SHA256_RES_INVALID_CONTEXT** (-1)
- #define **B5_HMAC_SHA256_RES_CANNOT_ALLOCATE_CONTEXT** (-2)
- #define **B5_HMAC_SHA256_RES_INVALID_ARGUMENT** (-3)

### 4.13.1 Detailed Description

## 4.14 SHA256 digest and block sizes

### SHA256 digest and block sizes

- #define **B5_SHA256_DIGEST_SIZE** 32
- #define **B5_SHA256_BLOCK_SIZE** 64

### 4.14.1 Detailed Description

## 4.15 SHA256 data structures

### Data Structures

- struct B5_tSha256Ctx

### 4.15.1 Detailed Description

## 4.16 SHA256 functions

### SHA256 functions

- int32_t B5_Sha256_Init (B5_tSha256Ctx ∗ctx)

  *Initialize the SHA256 context.*
- int32_t B5_Sha256_Update (B5_tSha256Ctx ∗ctx, const uint8_t ∗data, int32_t dataLen)

  *Compute the SHA256 algorithm on input data depending on the current status of the SHA256 context.*
- int32_t B5_Sha256_Finit (B5_tSha256Ctx ∗ctx, uint8_t ∗rDigest)

  *De-initialize the current SHA256 context.*

### 4.16.1 Detailed Description

### 4.16.2 Function Documentation

#### 4.16.2.1 int32_t B5_Sha256_Finit ( B5_tSha256Ctx ∗ *ctx,* uint8_t ∗ *rDigest* )

De-initialize the current SHA256 context.

**Parameters**

| | |
|---|---|
| *ctx* | Pointer to the SHA context to de-initialize. |
| *rDigest* | Pointer to a blank memory area that can store the computed output digest. |

**Returns**

See SHA256 return values .

#### 4.16.2.2 int32_t B5_Sha256_Init ( B5_tSha256Ctx ∗ *ctx* )

Initialize the SHA256 context.

**Parameters**

| | |
|---|---|
| *ctx* | Pointer to the SHA256 data structure to be initialized. |

**Returns**

See SHA256 return values .

#### 4.16.2.3 int32_t B5_Sha256_Update ( B5_tSha256Ctx ∗ *ctx,* const uint8_t ∗ *data,* int32_t *dataLen* )

Compute the SHA256 algorithm on input data depending on the current status of the SHA256 context.

**Parameters**

| | |
|---|---|
| *ctx* | Pointer to the current SHA context. |
| *data* | Pointer to the input data. |
| *dataLen* | Bytes to be processed. |

**Returns**

See SHA256 return values .

## 4.17 HMAC-SHA256 return values

### SHA256 return values

- #define **B5_HMAC_SHA256_RES_OK** ( 0)
- #define **B5_HMAC_SHA256_RES_INVALID_CONTEXT** (-1)
- #define **B5_HMAC_SHA256_RES_CANNOT_ALLOCATE_CONTEXT** (-2)
- #define **B5_HMAC_SHA256_RES_INVALID_ARGUMENT** (-3)

### 4.17.1 Detailed Description

## 4.18 HMAC-SHA256 data structures

### Data Structures

- struct B5_tHmacSha256Ctx

### 4.18.1 Detailed Description

## 4.19 HMAC-SHA256 functions

### HMAC-SHA256 functions

- int32_t B5_HmacSha256_Init (B5_tHmacSha256Ctx ∗ctx, const uint8_t ∗Key, int16_t keySize)

  *Initialize the HMAC-SHA256 context.*
- int32_t B5_HmacSha256_Update (B5_tHmacSha256Ctx ∗ctx, const uint8_t ∗data, int32_t dataLen)

  *Compute the HMAC-SHA256 algorithm on input data depending on the current status of the HMAC-SHA256 context.*
- int32_t B5_HmacSha256_Finit (B5_tHmacSha256Ctx ∗ctx, uint8_t ∗rDigest)

  *De-initialize the current HMAC-SHA256 context.*

### 4.19.1 Detailed Description

### 4.19.2 Function Documentation

#### 4.19.2.1 int32_t B5_HmacSha256_Finit ( **B5_tHmacSha256Ctx** ∗ *ctx,* uint8_t ∗ *rDigest* )

De-initialize the current HMAC-SHA256 context.

**Parameters**

| | |
|---|---|
| *ctx* | Pointer to the HMAC-SHA256 context to de-initialize. |
| *rDigest* | Pointer to a blank memory area that can store the computed output digest. |

**Returns**

See HMAC-SHA256 return values .

#### 4.19.2.2 int32_t B5_HmacSha256_Init ( **B5_tHmacSha256Ctx** ∗ *ctx,* const uint8_t ∗ *Key,* int16_t *keySize* )

Initialize the HMAC-SHA256 context.

**Parameters**

| | |
|---|---|
| *ctx* | Pointer to the HMAC-SHA256 data structure to be initialized. |
| *Key* | Pointer to the Key that must be used. |
| *keySize* | Key size. |

**Returns**

See HMAC-SHA256 return values .

#### 4.19.2.3 int32_t B5_HmacSha256_Update ( **B5_tHmacSha256Ctx** ∗ *ctx,* const uint8_t ∗ *data,* int32_t *dataLen* )

Compute the HMAC-SHA256 algorithm on input data depending on the current status of the HMAC-SHA256 context.

**Parameters**

| | |
|---|---|
| *ctx* | Pointer to the current HMAC-SHA256 context. |
| *data* | Pointer to the input data. |
| *dataLen* | Bytes to be processed. |

**Returns**

See HMAC-SHA256 return values .

# Chapter 5

# Data Structure Documentation

## 5.1 B5_tAesCtx Struct Reference

### Data Fields

- uint32_t rk [4 ∗(14+1)]
- uint8_t Nr
- uint8_t InitVector [16]
- uint8_t mode
- uint32_t const ∗ **Te0**
- uint32_t const ∗ **Te1**
- uint32_t const ∗ **Te2**
- uint32_t const ∗ **Te3**
- uint32_t const ∗ **Te4**
- uint32_t const ∗ **Td0**
- uint32_t const ∗ **Td1**
- uint32_t const ∗ **Td2**
- uint32_t const ∗ **Td3**
- uint32_t const ∗ **Td4**

### 5.1.1 Field Documentation

#### 5.1.1.1 uint8_t B5_tAesCtx::InitVector[16]

IV for OFB, CBC, CTR

#### 5.1.1.2 uint8_t B5_tAesCtx::mode

Active mode

#### 5.1.1.3 uint8_t B5_tAesCtx::Nr

Number of rounds

#### 5.1.1.4 uint32_t B5_tAesCtx::rk[4 *(14+1)]

Precomputed round keys

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

- src/Common/aes256.h

## 5.2 B5_tCmacAesCtx Struct Reference

### Data Fields

- B5_tAesCtx **aesCtx**
- uint8_t **K1** [32]
- uint8_t **K2** [32]
- uint8_t **tmpBlk** [B5_AES_BLK_SIZE]
- uint8_t **tmpBlkLen**
- uint8_t **C** [B5_AES_BLK_SIZE]

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

- src/Common/aes256.h

## 5.3 B5_tHmacSha256Ctx Struct Reference

### Data Fields

- B5_tSha256Ctx **shaCtx**
- uint8_t **iPad** [64]
- uint8_t **oPad** [64]

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

- src/Common/sha256.h

## 5.4 B5_tSha256Ctx Struct Reference

### Data Fields

- uint32_t **total** [2]
- uint32_t **state** [8]
- uint8_t **buffer** [64]
- uint32_t **W** [64]

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

- src/Common/sha256.h

## 5.5 se3_algo_ Struct Reference

SEcube Algorithm structure.

`#include <L1.h>`

### Data Fields

- uint8_t **name** [SE3_CMD1_CRYPTO_ALGOINFO_NAME_SIZE]
- uint16_t **type**
- uint16_t **block_size**
- uint16_t **key_size**

### 5.5.1 Detailed Description

SEcube Algorithm structure.

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

- src/Host/L1.h

## 5.6 se3_device_ Struct Reference

SEcube Device structure.

`#include <L0.h>`

### Data Fields

- se3_device_info **info**
- uint8_t ∗ **request**
- uint8_t ∗ **response**
- se3_file **f**
- bool **opened**

### 5.6.1 Detailed Description

SEcube Device structure.

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

- src/Host/L0.h

## 5.7 se3_device_info_ Struct Reference

SEcube Device Information structure.

`#include <L0.h>`

### Data Fields

- se3_char **path** [SE3_MAX_PATH]
- uint8_t **serialno** [SE3_SN_SIZE]
- uint8_t **hello_msg** [SE3_HELLO_SIZE]
- uint16_t **status**

### 5.7.1 Detailed Description

SEcube Device Information structure.

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

- src/Host/L0.h

## 5.8 se3_disco_it_ Struct Reference

Discovery iterator.

`#include <L0.h>`

### Data Fields

- se3_device_info **device_info**
- se3_drive_it **_drive_it**

### 5.8.1 Detailed Description

Discovery iterator.

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

- src/Host/L0.h

## 5.9 se3_discover_info_ Struct Reference

### Data Fields

- uint8_t **serialno** [SE3_SERIAL_SIZE]
- uint8_t **hello_msg** [SE3_HELLO_SIZE]
- uint16_t **status**

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

- src/Host/se3comm.h

## 5.10 se3_drive_it_ Struct Reference

### Data Fields

- se3_char ∗ **path**
- se3_char **buf_** [SE3_DRIVE_BUF_MAX+1]
- size_t **buf_len_**
- size_t **pos_**

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

- src/Host/se3comm.h

## 5.11 se3_file Struct Reference

### Data Fields

- OVERLAPPED **ol**
- HANDLE **h**

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

- src/Host/se3comm.h

## 5.12 se3_key_ Struct Reference

SEcube Key structure.

`#include <L1.h>`

### Data Fields

- uint32_t **id**
- uint32_t **validity**
- uint16_t **data_size**
- uint16_t **name_size**
- uint8_t ∗ **data**
- uint8_t **name** [SE3_KEY_NAME_MAX]

### 5.12.1 Detailed Description

SEcube Key structure.

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

- src/Host/L1.h

## 5.13 se3_payload_cryptoctx_ Struct Reference

### Data Fields

- B5_tAesCtx **aesenc**
- B5_tAesCtx **aesdec**
- B5_tHmacSha256Ctx **hmac**
- uint8_t **hmac_key** [B5_AES_256]
- uint8_t **auth** [B5_SHA256_DIGEST_SIZE]

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

- src/Common/se3_common.h

## 5.14 se3_session_ Struct Reference

SEcube Communication session structure.

`#include <L1.h>`

### Data Fields

- se3_device **device**
- uint8_t **token** [SE3_L1_TOKEN_SIZE]
- uint8_t **key** [SE3_L1_KEY_SIZE]
- uint8_t **buf** [SE3_COMM_N ∗SE3_COMM_BLOCK]
- bool **locked**
- bool **logged_in**
- uint32_t **timeout**
- se3_file **hfile**
- se3_payload_cryptoctx **cryptoctx**
- bool **cryptoctx_initialized**

### 5.14.1 Detailed Description

SEcube Communication session structure.

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

- src/Host/L1.h

# Chapter 6

# File Documentation

## 6.1 src/Common/crc16.h File Reference

This file contains defines and functions for computing CRC.

```
#include <stddef.h>
#include <stdint.h>
```

### Functions

- uint16_t se3_crc16_update (size_t length, const uint8_t ∗data, uint16_t crc)

  *Compute CRC.*

### Variables

- const uint16_t **se3_crc16_table** [0x100]

### 6.1.1 Detailed Description

This file contains defines and functions for computing CRC.

### 6.1.2 Function Documentation

#### 6.1.2.1 uint16_t se3_crc16_update ( size_t *length,* const uint8_t ∗ *data,* uint16_t *crc* )

Compute CRC.

**Parameters**

| | | |
|---|---|---|
| `in` | *length* | Data length |
| `in` | *data* | Data on which CRC is computed |
| `in` | *crc* | CRC |

**Returns**

CRC computed

## 6.2 src/Common/se3_common.h File Reference

This file contains defines and functions common for L0 and L1.

```
#include "se3c0def.h"
#include "aes256.h"
#include "sha256.h"
#include "pbkdf2.h"
```

### Data Structures

- struct se3_payload_cryptoctx_

### Typedefs

- typedef struct se3_payload_cryptoctx_ **se3_payload_cryptoctx**

### Functions

- uint16_t se3_req_len_data (uint16_t len_data_and_headers)

  *Compute length of data in a request in terms of SE3_COMM_BLOCK blocks.*
- uint16_t se3_req_len_data_and_headers (uint16_t len_data)

  *Compute length of data in a request accounting for headers.*
- uint16_t se3_resp_len_data (uint16_t len_data_and_headers)

  *Compute length of data in a request in terms of SE3_COMM_BLOCK blocks.*
- uint16_t se3_resp_len_data_and_headers (uint16_t len_data)

  *Compute length of data in a response accounting for headers.*
- uint16_t se3_nblocks (uint16_t len)

  *Compute number of SE3_COMM_BLOCK blocks, given length in Bytes.*
- void **se3_payload_cryptoinit** (se3_payload_cryptoctx ∗ctx, const uint8_t ∗key)
- void **se3_payload_encrypt** (se3_payload_cryptoctx ∗ctx, uint8_t ∗auth, uint8_t ∗iv, uint8_t ∗data, uint16_t nblocks, uint16_t flags)
- bool **se3_payload_decrypt** (se3_payload_cryptoctx ∗ctx, const uint8_t ∗auth, const uint8_t ∗iv, uint8_t ∗data, uint16_t nblocks, uint16_t flags)

### Variables

- const uint8_t **se3_magic** [SE3_MAGIC_SIZE]

### 6.2.1 Detailed Description

This file contains defines and functions common for L0 and L1.

### 6.2.2 Function Documentation

#### 6.2.2.1 uint16_t se3_nblocks ( uint16_t *len* )

Compute number of SE3_COMM_BLOCK blocks, given length in Bytes.

**Parameters**

| | | |
|---|---|---|
| `in` | *len* | Length |

**Returns**

Number of Blocks

#### 6.2.2.2 uint16_t se3_req_len_data ( uint16_t *len_data_and_headers* )

Compute length of data in a request in terms of SE3_COMM_BLOCK blocks.

**Parameters**

| | | |
|---|---|---|
| `in` | *len_data_and_headers* | Data length |

**Returns**

Number of SE3_COMM_BLOCK blocks

#### 6.2.2.3 uint16_t se3_req_len_data_and_headers ( uint16_t *len_data* )

Compute length of data in a request accounting for headers.

**Parameters**

| | | |
|---|---|---|
| `in` | *len_data* | Data length |

**Returns**

Number of Bytes

#### 6.2.2.4 uint16_t se3_resp_len_data ( uint16_t *len_data_and_headers* )

Compute length of data in a request in terms of SE3_COMM_BLOCK blocks.

**Parameters**

| | | |
|---|---|---|
| `in` | *len_data_and_headers* | Data length |

**Returns**

Number of SE3_COMM_BLOCK blocks

#### 6.2.2.5 uint16_t se3_resp_len_data_and_headers ( uint16_t *len_data* )

Compute length of data in a response accounting for headers.

**Parameters**

| | | |
|---|---|---|
| `in` | *len_data* | Data Length |

**Returns**

Number of Bytes

## 6.3 src/Common/se3c1def.h File Reference

This file contains defines to be used both for L1 and L0 functions.

`#include "se3c0def.h"`

### Macros

- #define **SE3_DIR_SHIFT** (8)

### Enumerations

- enum {
  SE3_ERR_ACCESS = 100, SE3_ERR_PIN = 101, SE3_ERR_RESOURCE = 200, SE3_ERR_EXPIRED = 201,
  SE3_ERR_MEMORY = 400, SE3_ERR_AUTH = 401 }
- enum { **SE3_ACCESS_USER** = 100, **SE3_ACCESS_ADMIN** = 1000, **SE3_ACCESS_MAX** = 0xFFFF }
- enum { **SE3_RECORD_SIZE** = 32, **SE3_RECORD_MAX** = 2 }
- enum { **SE3_RECORD_TYPE_ADMINPIN** = 0, **SE3_RECORD_TYPE_USERPIN** = 1 }
- enum {
  **SE3_L1_PIN_SIZE** = 32, **SE3_L1_KEY_SIZE** = 32, **SE3_L1_AUTH_SIZE** = 16, **SE3_L1_CRYPTOBLOCK_SIZE** = 16,
  **SE3_L1_CHALLENGE_SIZE** = 32, **SE3_L1_CHALLENGE_ITERATIONS** = 32, **SE3_L1_IV_SIZE** = 16, **SE3_L1_TOKEN_SIZE** = 16 }
- enum {
  **SE3_REQ1_OFFSET_AUTH** = 0, **SE3_REQ1_OFFSET_IV** = 16, **SE3_REQ1_OFFSET_TOKEN** = 32, **SE3_REQ1_OFFSET_LEN** = 48,
  **SE3_REQ1_OFFSET_CMD** = 50, **SE3_REQ1_OFFSET_DATA** = 64, **SE3_REQ1_MAX_DATA** = (SE3_REQ_MAX_DATA - SE3_REQ1_OFFSET_DATA) }
- enum {
  **SE3_RESP1_OFFSET_AUTH** = 0, **SE3_RESP1_OFFSET_IV** = 16, **SE3_RESP1_OFFSET_TOKEN** = 32, **SE3_RESP1_OFFSET_LEN** = 48,
  **SE3_RESP1_OFFSET_STATUS** = 50, **SE3_RESP1_OFFSET_DATA** = 64, **SE3_RESP1_MAX_DATA** = (SE3_RESP_MAX_DATA - SE3_RESP1_OFFSET_DATA) }

- enum {
  **SE3_CMD1_CHALLENGE** = 1, **SE3_CMD1_LOGIN** = 2, **SE3_CMD1_LOGOUT** = 3, **SE3_CMD1_CONFIG** = 4,
  **SE3_CMD1_KEY_EDIT** = 5, **SE3_CMD1_KEY_LIST** = 6, **SE3_CMD1_CRYPTO_INIT** = 7, **SE3_CMD1_CRYPTO_UPDATE** = 8,
  **SE3_CMD1_CRYPTO_LIST** = 9, **SE3_CMD1_CRYPTO_SET_TIME** = 10 }
- enum { **SE3_CONFIG_OP_GET** = 1, **SE3_CONFIG_OP_SET** = 2 }
- enum { **SE3_CMD1_CONFIG_REQ_OFF_ID** = 0, **SE3_CMD1_CONFIG_REQ_OFF_OP** = 2, **SE3_CMD1_CONFIG_REQ_OFF_VALUE** = 4, **SE3_CMD1_CONFIG_RESP_OFF_VALUE** = 0 }
- enum {
  **SE3_CMD1_CHALLENGE_REQ_OFF_CC1** = 0, **SE3_CMD1_CHALLENGE_REQ_OFF_CC2** = 32, **SE3_CMD1_CHALLENGE_REQ_OFF_ACCESS** = 64, **SE3_CMD1_CHALLENGE_REQ_SIZE** = 66,
  **SE3_CMD1_CHALLENGE_RESP_OFF_SC** = 0, **SE3_CMD1_CHALLENGE_RESP_OFF_SRESP** = 32,
  **SE3_CMD1_CHALLENGE_RESP_SIZE** = 64 }
- enum { **SE3_CMD1_LOGIN_REQ_OFF_CRESP** = 0, **SE3_CMD1_LOGIN_REQ_SIZE** = 32, **SE3_CMD1_LOGIN_RESP_OFF_TOKEN** = 0, **SE3_CMD1_LOGIN_RESP_SIZE** = 16 }
- enum { **SE3_KEY_DATA_MAX** = 2048, **SE3_KEY_NAME_MAX** = 32 }
- enum { SE3_KEY_OP_INSERT = 1, SE3_KEY_OP_DELETE = 2, SE3_KEY_OP_UPSERT = 3 }
- enum {
  **SE3_CMD1_KEY_EDIT_REQ_OFF_OP** = 0, **SE3_CMD1_KEY_EDIT_REQ_OFF_ID** = 2, **SE3_CMD1_KEY_EDIT_REQ_OFF_VALIDITY** = 6, **SE3_CMD1_KEY_EDIT_REQ_OFF_DATA_LEN** = 10,
  **SE3_CMD1_KEY_EDIT_REQ_OFF_NAME_LEN** = 12, **SE3_CMD1_KEY_EDIT_REQ_OFF_DATA_AND_NAME** = 14 }
- enum {
  **SE3_CMD1_KEY_LIST_REQ_SIZE** = 4, **SE3_CMD1_KEY_LIST_REQ_OFF_SKIP** = 0, **SE3_CMD1_KEY_LIST_REQ_OFF_NMAX** = 2, **SE3_CMD1_KEY_LIST_RESP_OFF_COUNT** = 0,
  **SE3_CMD1_KEY_LIST_RESP_OFF_KEYINFO** = 2, **SE3_CMD1_KEY_LIST_KEYINFO_OFF_ID** = 0, **SE3_CMD1_KEY_LIST_KEYINFO_OFF_VALIDITY** = 4, **SE3_CMD1_KEY_LIST_KEYINFO_OFF_DATA_LEN** = 8,
  **SE3_CMD1_KEY_LIST_KEYINFO_OFF_NAME_LEN** = 10, **SE3_CMD1_KEY_LIST_KEYINFO_OFF_NAME** = 12 }
- enum { **SE3_ALGO_INVALID** = 0xFFFF, **SE3_SESSION_INVALID** = 0xFFFFFFFF, **SE3_KEY_INVALID** = 0xFFFFFFFF }
- enum {
  SE3_ALGO_AES = 0, SE3_ALGO_SHA256 = 1, SE3_ALGO_HMACSHA256 = 2, SE3_ALGO_AES_HMACSHA256 = 3,
  SE3_ALGO_AES_HMAC = 4, **SE3_ALGO_MAX** = 8 }
- enum {
  **SE3_CMD1_CRYPTO_INIT_REQ_SIZE** = 8, **SE3_CMD1_CRYPTO_INIT_REQ_OFF_ALGO** = 0, **SE3_CMD1_CRYPTO_INIT_REQ_OFF_MODE** = 2, **SE3_CMD1_CRYPTO_INIT_REQ_OFF_KEY_ID** = 4,
  **SE3_CMD1_CRYPTO_INIT_RESP_SIZE** = 4, **SE3_CMD1_CRYPTO_INIT_RESP_OFF_SID** = 0 }
- enum {
  **SE3_CMD1_CRYPTO_UPDATE_REQ_OFF_SID** = 0, **SE3_CMD1_CRYPTO_UPDATE_REQ_OFF_FLAGS** = 4, **SE3_CMD1_CRYPTO_UPDATE_REQ_OFF_DATAIN1_LEN** = 6, **SE3_CMD1_CRYPTO_UPDATE_REQ_OFF_DATAIN2_LEN** = 8,
  **SE3_CMD1_CRYPTO_UPDATE_REQ_OFF_DATA** = 16, **SE3_CMD1_CRYPTO_UPDATE_RESP_OFF_DATAOUT_LEN** = 0, **SE3_CMD1_CRYPTO_UPDATE_RESP_OFF_DATA** = 16 }
- enum {
  **SE3_CRYPTO_FLAG_FINIT** = (1 << 15), **SE3_CRYPTO_FLAG_RESET** = (1 << 14), **SE3_CRYPTO_FLAG_SETIV** = SE3_CRYPTO_FLAG_RESET, **SE3_CRYPTO_FLAG_SETNONCE** = (1 << 13),
  **SE3_CRYPTO_FLAG_AUTH** = (1 << 12) }
- enum { **SE3_CRYPTO_MAX_DATAIN** = (SE3_REQ1_MAX_DATA - SE3_CMD1_CRYPTO_UPDATE_REQ_OFF_DATA), **SE3_CRYPTO_MAX_DATAOUT** = (SE3_RESP1_MAX_DATA - SE3_CMD1_CRYPTO_UPDATE_RESP_OFF_DATA) }
- enum { **SE3_CMD1_CRYPTO_SET_TIME_REQ_SIZE** = 4, **SE3_CMD1_CRYPTO_SET_TIME_REQ_OFF_DEVTIME** = 0 }

- enum {
  **SE3_CMD1_CRYPTO_LIST_REQ_SIZE** = 0, **SE3_CMD1_CRYPTO_LIST_RESP_OFF_COUNT** = 0, **S↩ E3_CMD1_CRYPTO_LIST_RESP_OFF_ALGOINFO** = 2, **SE3_CMD1_CRYPTO_ALGOINFO_SIZE** = 22,
  **SE3_CMD1_CRYPTO_ALGOINFO_OFF_NAME** = 0, **SE3_CMD1_CRYPTO_ALGOINFO_OFF_TYPE** = 16, **SE3_CMD1_CRYPTO_ALGOINFO_OFF_BLOCK_SIZE** = 18, **SE3_CMD1_CRYPTO_ALGOINFO_↩ OFF_KEY_SIZE** = 20,
  **SE3_CMD1_CRYPTO_ALGOINFO_NAME_SIZE** = 16 }
- enum {
  **SE3_CRYPTO_TYPE_BLOCKCIPHER** = 0, **SE3_CRYPTO_TYPE_STREAMCIPHER** = 1, **SE3_CRYPT↩ O_TYPE_DIGEST** = 2, **SE3_CRYPTO_TYPE_BLOCKCIPHER_AUTH** = 3,
  **SE3_CRYPTO_TYPE_OTHER** = 0xFFFF }
- enum {
  **SE3_FEEDBACK_ECB** = 1, **SE3_FEEDBACK_CBC** = 2, **SE3_FEEDBACK_OFB** = 3, **SE3_FEEDBACK↩ _CTR** = 4,
  **SE3_FEEDBACK_CFB** = 5, **SE3_DIR_ENCRYPT** = (1 << SE3_DIR_SHIFT), **SE3_DIR_DECRYPT** = (2 << SE3_DIR_SHIFT) }

  *L1_crypto_init default modes.*

### 6.3.1 Detailed Description

This file contains defines to be used both for L1 and L0 functions.

### 6.3.2 Enumeration Type Documentation

#### 6.3.2.1 anonymous enum

Configuration records definitions

#### 6.3.2.2 anonymous enum

Default configuration record types

#### 6.3.2.3 anonymous enum

L1 field size definitions

#### 6.3.2.4 anonymous enum

L1 request fields definitions

#### 6.3.2.5 anonymous enum

L1 response fields definitions

#### 6.3.2.6 anonymous enum

L1 command codes

#### 6.3.2.7 anonymous enum

L1_config operations

#### 6.3.2.8 anonymous enum

L1_config fields

#### 6.3.2.9 anonymous enum

L1_challenge fields

#### 6.3.2.10 anonymous enum

L1_login fields

#### 6.3.2.11 anonymous enum

Keys: maximum sizes for variable fields

#### 6.3.2.12 anonymous enum

L1_key_edit fields

#### 6.3.2.13 anonymous enum

L1_key_list fields

#### 6.3.2.14 anonymous enum

Invalid handle values

#### 6.3.2.15 anonymous enum

L1_crypto_init fields

#### 6.3.2.16 anonymous enum

L1_crypto_update fields

#### 6.3.2.17 anonymous enum

L1_crypto_update default flags

#### 6.3.2.18 anonymous enum

L1_crypto_update maximum buffer sizes

#### 6.3.2.19 anonymous enum

L1_crypto_set_time fields

#### 6.3.2.20 anonymous enum

L1_crypto_list fields

#### 6.3.2.21 anonymous enum

L1_crypto_list default cipher types

#### 6.3.2.22 anonymous enum

L1_crypto_init default modes.

One FEEDBACK and one DIR may be combined to specify the desired mode Example: Encrypt in CBC mode (SE3_FEEDBACK_CBC | SE3_DIR_ENCRYPT)

#### 6.3.2.23 anonymous enum

L1 errors

**Enumerator**

| | |
|---|---|
| ***SE3_ERR_ACCESS*** | insufficient privileges |
| ***SE3_ERR_PIN*** | pin rejected |
| ***SE3_ERR_RESOURCE*** | resource not found |
| ***SE3_ERR_EXPIRED*** | resource expired |
| ***SE3_ERR_MEMORY*** | no more space to allocate resource |
| ***SE3_ERR_AUTH*** | SHA256HMAC Authentication failed. |

## 6.4 src/Host/L0.h File Reference

This file contains L0 functions and structures.

```
#include "se3_common.h"
#include "se3comm.h"
#include "crc16.h"
```

### Data Structures

- struct se3_device_info_

  *SEcube Device Information structure.*
- struct se3_device_

  *SEcube Device structure.*
- struct se3_disco_it_

  *Discovery iterator.*

### Macros

- #define **SE3_NBLOCKS** (SE3_COMM_N-1)
- #define **SE3_TIMEOUT** (10000)
- #define **SE3_RES_SIZE_HEADER** (32)
- #define **SE3_SIZE_PAYLOAD_MAX** ((SE3_COMM_BLOCK ∗ SE3_NBLOCKS) - SE3_REQ_SIZE_HEA↵ DER - (SE3_COMM_BLOCK ∗ SE3_REQDATA_SIZE_HEADER))

### Typedefs

- typedef struct se3_device_info_ se3_device_info

  *SEcube Device Information structure.*
- typedef struct se3_device_ se3_device

  *SEcube Device structure.*
- typedef struct se3_disco_it_ se3_disco_it

  *Discovery iterator.*

### Functions

- uint16_t L0_TXRX (se3_device ∗device, uint16_t req_cmd, uint16_t req_cmdflags, uint16_t req_len, const uint8_t ∗req_data, uint16_t ∗resp_status, uint16_t ∗resp_len, uint8_t ∗resp_data)

  *Main function for communicating with SEcube device.*
- uint16_t L0_echo (se3_device ∗device, const uint8_t ∗data_in, uint16_t data_in_len, uint8_t ∗data_out)

  *Echo service.*
- uint16_t L0_factoryinit (se3_device ∗device, const uint8_t ∗serialno)

  *Initialise SEcube device.*
- uint16_t L0_open (se3_device ∗dev, se3_device_info ∗dev_info, uint32_t timeout)

  *Open SEcube device.*
- void L0_close (se3_device ∗dev)

  *Close SEcube device.*
- bool L0_discover_serialno (uint8_t ∗serialno, se3_device_info ∗device)

  *Discover Serial Number information.*
- void L0_discover_init (se3_disco_it ∗it)

  *Initialise discovery iterator.*
- bool L0_discover_next (se3_disco_it ∗it)

  *Increment discovery iterator.*

### 6.4.1 Detailed Description

This file contains L0 functions and structures.

### 6.4.2 Function Documentation

#### 6.4.2.1 void L0_close ( se3_device * *dev* )

Close SEcube device.

**Parameters**

| | | |
|---|---|---|
| `in` | *dev* | pointer to SEcube device structure |

**Returns**

Error code or SE3_OK

#### 6.4.2.2 void L0_discover_init ( se3_disco_it * *it* )

Initialise discovery iterator.

**Parameters**

| | | |
|---|---|---|
| `in` | *it* | iterator |

**Returns**

Error code or SE3_OK

#### 6.4.2.3 bool L0_discover_next ( se3_disco_it * *it* )

Increment discovery iterator.

**Parameters**

| | | |
|---|---|---|
| `in` | *it* | iterator |

**Returns**

Error code or SE3_OK

Details

#### 6.4.2.4 bool L0_discover_serialno ( uint8_t ∗ *serialno,* se3_device_info ∗ *device* )

Discover Serial Number information.

**Parameters**

| | | |
|---|---|---|
| `in` | *serialno* | Serial Number of SEcube device |
| `in` | *device* | pointer to SEcube device structure |

**Returns**

Error code or SE3_OK

#### 6.4.2.5 uint16_t L0_echo ( se3_device ∗ *device,* const uint8_t ∗ *data_in,* uint16_t *data_in_len,* uint8_t ∗ *data_out* )

Echo service.

**Parameters**

| | | |
|---|---|---|
| `in` | *device* | pointer to SEcube device structure |
| `in` | *data_in* | Data to be sent |
| `in` | *data_in_len* | Length of input data |
| `in` | *data_out* | Data to be sent |

**Returns**

Error code or SE3_OK

Details

#### 6.4.2.6 uint16_t L0_factoryinit ( se3_device ∗ *device,* const uint8_t ∗ *serialno* )

Initialise SEcube device.

**Parameters**

| | | |
|---|---|---|
| `in` | *device* | pointer to SEcube device structure |
| `in` | *serialno* | Serial Number to be set on SEcube device |

**Returns**

Error code or SE3_OK

Before using the SEcube device, this function must be called. It can be used just once-

#### 6.4.2.7 uint16_t L0_open ( se3_device ∗ *dev,* se3_device_info ∗ *dev_info,* uint32_t *timeout* )

Open SEcube device.

**Parameters**

| | | |
|---|---|---|
| `in` | *dev* | pointer to SEcube device structure |
| `in` | *dev_info* | Device Information structure |
| `in` | *timeout* | timeout in ms |

**Returns**

Error code or SE3_OK

#### 6.4.2.8 uint16_t L0_TXRX ( se3_device ∗ *device,* uint16_t *req_cmd,* uint16_t *req_cmdflags,* uint16_t *req_len,* const uint8_t ∗ *req_data,* uint16_t ∗ *resp_status,* uint16_t ∗ *resp_len,* uint8_t ∗ *resp_data* )

Main function for communicating with SEcube device.

**Parameters**

| | | |
|---|---|---|
| `in` | *device* | pointer to SEcube device structure |
| `in` | *req_cmd* | Command to be executed |
| `in` | *req_cmdflags* | Flag options for the command |
| `in` | *req_len* | Length of the request |
| `in` | *req_data* | array containing the request |
| `in` | *resp_status* | Response status (received response or not) |
| `in` | *resp_len* | Length of the response |
| `in` | *resp_data* | array containing the response |

**Returns**

Error code or SE3_OK

The function receive payload data from upper levels; segment the data and write it to the device.

**Parameters**

| | |
|---|---|
| *resp_len* | in: maximum size of resp_data, out: effective size of resp_data |

## 6.5 src/Host/L1.h File Reference

This file contains L1 functions and structures.

```
#include "L0.h"
#include "se3c1def.h"
```

## Data Structures

- struct se3_session_

  *SEcube Communication session structure.*
- struct se3_key_

  *SEcube Key structure.*
- struct se3_algo_

  *SEcube Algorithm structure.*

## Macros

- #define **SE3_REQ_CHALLENGE_SIZE** (96+16)
- #define **SE3_REQ_CHALLENGE_IV_OFFSET** (0)
- #define **SE3_REQ_CHALLENGE_TOKEN_OFFSET** (16)
- #define **SE3_REQ_CHALLENGE_CC_OFFSET** (32)
- #define **SE3_REQ_CHALLENGE_CC2_OFFSET** (64)
- #define **SE3_REQ_CHALLENGE_ACCESS_OFFSET** (96)
- #define **SE3_RESP_CHALLENGE_SC_OFFSET** (32)
- #define **SE3_RESP_LOGIN_TOKEN_OFFSET** (32)

## Typedefs

- typedef struct se3_session_ se3_session

  *SEcube Communication session structure.*
- typedef struct se3_key_ se3_key

  *SEcube Key structure.*
- typedef struct se3_algo_ se3_algo

  *SEcube Algorithm structure.*

## Functions

- uint16_t L1_login (se3_session ∗s, se3_device ∗dev, const uint8_t ∗pin, uint16_t access)

  *This function is used to let a user/admin login on the device.*
- uint16_t L1_set_admin_PIN (se3_session ∗s, uint8_t ∗pin)

  *This function is used to change the current admin pin.*
- uint16_t L1_set_user_PIN (se3_session ∗s, uint8_t ∗pin)

  *This function is used to change the current user pin.*
- uint16_t L1_logout (se3_session ∗s)

  *This function is used to logout from the device.*
- uint16_t L1_key_list (se3_session ∗s, uint16_t skip, uint16_t max_keys, se3_key ∗key_array, uint16_t ∗count)

  *This function is used get the list of the already of the already available keys on the device.*
- uint16_t L1_key_edit (se3_session ∗s, uint16_t op, se3_key ∗k)

  *This function is used to edit the keys data on the device.*
- bool L1_find_key (se3_session ∗s, uint32_t key_id)

  *Check if a Key is present or not.*
- uint16_t L1_crypto_init (se3_session ∗s, uint16_t algorithm, uint16_t mode, uint32_t key_id, uint32_t ∗sess↩_id)

  *Initialise a crypto session.*
- uint16_t L1_crypto_update (se3_session ∗s, uint32_t sess_id, uint16_t flags, uint16_t data1_len, uint8_↩t ∗data1, uint16_t data2_len, uint8_t ∗data2, uint16_t ∗dataout_len, uint8_t ∗data_out)

*Update a crypto session.*

- uint16_t L1_crypto_set_time (se3_session ∗s, uint32_t devtime)

  *Set time for a crypto session.*
- uint16_t L1_encrypt (se3_session ∗s, uint16_t algorithm, uint16_t mode, uint32_t key_id, size_t datain_len, int8_t ∗data_in, size_t ∗dataout_len, uint8_t ∗data_out)

  *This function is used to encrypt a buffer of data given the algorithm, the encryption mode, the buffer size, and where to store the encrypted data.*
- uint16_t L1_decrypt (se3_session ∗s, uint16_t algorithm, uint16_t mode, uint32_t key_id, size_t datain_len, int8_t ∗data_in, size_t ∗dataout_len, uint8_t ∗data_out)

  *This function is used to decrypt a buffer of data given the algorithm, the decryption mode, the buffer size, and where to store the decrypted data.*
- uint16_t L1_digest (se3_session ∗s, uint16_t algorithm, size_t datain_len, int8_t ∗data_in, size_t ∗dataout_len, uint8_t ∗data_out)

  *This function is used to sign a buffer of data given the algorithm, the amount of data to sign and where to store them.*
- uint16_t L1_get_algorithms (se3_session ∗s, uint16_t skip, uint16_t max_algorithms, se3_algo ∗algorithms_array, uint16_t ∗count)

  *This function is used to retrieve a list from the device of available algorithms.*

## 6.5.1 Detailed Description

This file contains L1 functions and structures.

## 6.5.2 Function Documentation

### 6.5.2.1 uint16_t L1_crypto_init ( se3_session ∗ *s*, uint16_t *algorithm*, uint16_t *mode*, uint32_t *key_id*, uint32_t ∗ *sess_id* )

Initialise a crypto session.

**Parameters**

| | | |
|---|---|---|
| `in` | *s* | Pointer to current se3_session, you must be logged in |
| `in` | *algorithm* | Which algorithm to use, see AlgorithmAvail |
| `in` | *mode* | This parameter strictly depends on the which algorithm is chosen |
| `in` | *key_id* | Which key ID to use for encryption |
| `in` | *sess_id* | Session ID |

**Returns**

Error code or SE3_OK

### 6.5.2.2 uint16_t L1_crypto_set_time ( se3_session ∗ *s*, uint32_t *devtime* )

Set time for a crypto session.

**Parameters**

| | | |
|---|---|---|
| `in` | *s* | Pointer to current se3_session, you must be logged in |
| `in` | *devtime* | Time to be set |

**Returns**

Error code or SE3_OK

#### 6.5.2.3 uint16_t L1_crypto_update ( se3_session ∗ *s,* uint32_t *sess_id,* uint16_t *flags,* uint16_t *data1_len,* uint8_t ∗ *data1,* uint16_t *data2_len,* uint8_t ∗ *data2,* uint16_t ∗ *dataout_len,* uint8_t ∗ *data_out* )

Update a crypto session.

**Parameters**

| | | |
|---|---|---|
| `in` | *s* | Pointer to current se3_session, you must be logged in |
| `in` | *sess_id* | Session ID |
| `in` | *flags* | Parameter_Description |
| `in` | *data1_len* | How long is the buffer you want to encrypt |
| `in` | *data1* | Pointer to input buffer 1 |
| `in` | *data2_len* | Length of input buffer 1 |
| `in` | *data2* | Pointer to input buffer 2 |
| `out` | *dataout_len* | Length of input buffer 1 |
| `out` | *data_out* | Pointer to the output buffer |

**Returns**

Error code or SE3_OK

#### 6.5.2.4 uint16_t L1_decrypt ( se3_session ∗ *s,* uint16_t *algorithm,* uint16_t *mode,* uint32_t *key_id,* size_t *datain_len,* int8_t ∗ *data_in,* size_t ∗ *dataout_len,* uint8_t ∗ *data_out* )

This function is used to decrypt a buffer of data given the algorithm, the decryption mode, the buffer size, and where to store the decrypted data.

**Parameters**

| | | |
|---|---|---|
| `in` | *s* | Pointer to current se3_session, you must be logged in |
| `in` | *algorithm* | Which algorithm to use, see AlgorithmAvail |
| `in` | *mode* | This parameter strictly depends on the which algorithm is chosen |
| `in` | *key_id* | Which key ID to use for decryption |
| `in` | *datain_len* | How long is the buffer you want to decrypt |
| `in` | *data_in* | Pointer to the buffer |
| `out` | *dataout_len* | How many data were actually decrypted |
| `out` | *data_out* | Pointer to a pre-allocated buffer where to store the clear text |

**Returns**

It returns SE3_OK on success, otherwise see se3c1def.h

#### 6.5.2.5 uint16_t L1_digest ( **se3_session** ∗ *s,* uint16_t *algorithm,* size_t *datain_len,* int8_t ∗ *data_in,* size_t ∗ *dataout_len,* uint8_t ∗ *data_out* )

This function is used to sign a buffer of data given the algorithm, the amount of data to sign and where to store them.

**Parameters**

| | | |
|---|---|---|
| `in` | *s* | Pointer to current se3_session, you must be logged in |
| `in` | *algorithm* | Which algorithm to use, see AlgorithmAvail |
| `in` | *datain_len* | How long is the buffer you want to sign |
| `in` | *data_in* | Pointer to the buffer |
| `out` | *dataout_len* | How many data were actually signed (can be NULL) |
| `out` | *data_out* | Pointer to a pre-allocated buffer where to store the digest |

**Returns**

It returns SE3_OK on success, otherwise see se3c1def.h

#### 6.5.2.6 uint16_t L1_encrypt ( **se3_session** ∗ *s,* uint16_t *algorithm,* uint16_t *mode,* uint32_t *key_id,* size_t *datain_len,* int8_t ∗ *data_in,* size_t ∗ *dataout_len,* uint8_t ∗ *data_out* )

This function is used to encrypt a buffer of data given the algorithm, the encryption mode, the buffer size, and where to store the encrypted data.

**Parameters**

| | | |
|---|---|---|
| `in` | *s* | Pointer to current se3_session, you must be logged in |
| `in` | *algorithm* | Which algorithm to use, see AlgorithmAvail |
| `in` | *mode* | This parameter strictly depends on the which algorithm is chosen |
| `in` | *key_id* | Which key ID to use for encryption |
| `in` | *datain_len* | How long is the buffer you want to encrypt |
| `in` | *data_in* | Pointer to the buffer |
| `out` | *dataout_len* | How many data were actually encrypted |
| `out` | *data_out* | Pointer to a pre-allocated buffer where to store the cipher text |

**Returns**

It returns SE3_OK on success, otherwise see se3c1def.h

#### 6.5.2.7 bool L1_find_key ( **se3_session** ∗ *s,* uint32_t *key_id* )

Check if a Key is present or not.

**Parameters**

| | | |
|---|---|---|
| `in` | *s* | Pointer to current se3_session, you must be logged in |
| `in` | *key_id* | ID of key to be found |

**Returns**

true if key is found, false otherwise

#### 6.5.2.8 uint16_t L1_get_algorithms ( **se3_session** ∗ *s,* uint16_t *skip,* uint16_t *max_algorithms,* **se3_algo** ∗ *algorithms_array,* uint16_t ∗ *count* )

This function is used to retrieve a list from the device of available algorithms.

**Parameters**

| | | |
|---|---|---|
| `in` | *s* | Pointer to current se3_session, you must be logged in |
| `in` | *skip* | How many algorithms you want to skip from the beginning of the device list |
| `in` | *max_algorithms* | How many algorithms you want to retrieve from the device |
| `out` | *algorithms_array* | Pointer to the already allocated array where to store the algorithms |
| `in` | *count* | Effective number of retrieved keys |

**Returns**

It returns SE3_OK on success, otherwise see se3c1def.h

#### 6.5.2.9 uint16_t L1_key_edit ( **se3_session** ∗ *s,* uint16_t *op,* **se3_key** ∗ *k* )

This function is used to edit the keys data on the device.

**Parameters**

| | | |
|---|---|---|
| `in` | *s* | Pointer to current se3_session, you must be logged in |
| `in` | *op* | see KeyOpEdit |
| `in` | *k* | Key value you want to add/update/delete |

**Returns**

It returns SE3_OK on success, otherwise see se3c1def.h

#### 6.5.2.10 uint16_t L1_key_list ( **se3_session** ∗ *s,* uint16_t *skip,* uint16_t *max_keys,* **se3_key** ∗ *key_array,* uint16_t ∗ *count* )

This function is used get the list of the already of the already available keys on the device.

**Parameters**

| | | |
|---|---|---|
| `in` | *s* | Pointer to current se3_session, you must be logged in |
| `in` | *skip* | How many keys you want to skip from the beginning of the list |
| `in` | *max_keys* | How many keys you want to retrieve from the device |
| `out` | *key_array* | Pointer to the already allocated array where to store the keys |
| `out` | *count* | Effective number of retrieved keys |

**Returns**

It returns SE3_OK on success, otherwise see se3c1def.h

#### 6.5.2.11 uint16_t L1_login ( **se3_session** ∗ *s,* **se3_device** ∗ *dev,* const uint8_t ∗ *pin,* uint16_t *access* )

This function is used to let a user/admin login on the device.

**Parameters**

| | | |
|---|---|---|
| `out` | *s* | Pointer to an already allocated se3_session object where to store current logged in session |
| `in` | *dev* | Device you want to login to |
| `in` | *pin* | Password to login |
| `in` | *access* | see AccessLogin |

**Returns**

It returns SE3_OK on success, otherwise see se3c1def.h

Before issueing any command to the device, you need to login. Some operations are allowed only to the admin user. After a flash erase, the admin pin and the user pin are both a sequence of 32 0s, please use L1_set_admin_PIN or L1_set_user_PIN to change them.

#### 6.5.2.12 uint16_t L1_logout ( **se3_session** ∗ *s* )

This function is used to logout from the device.

**Parameters**

| | | |
|---|---|---|
| `in` | *s* | Current session you want to end |

**Returns**

It returns SE3_OK on success, otherwise see se3c1def.h

After issueing this function, you will be forbidden to perform any command on the device. This can also be used to free the allocated resources, such as cryptographic sessions, with just one call.

#### 6.5.2.13 uint16_t L1_set_admin_PIN ( **se3_session** ∗ *s,* uint8_t ∗ *pin* )

This function is used to change the current admin pin.

**Parameters**

| | | |
|---|---|---|
| `in` | *s* | Pointer to current se3_session, you must be logged in as admin to issue this command |
| `in` | *pin* | New pin to be set |

**Returns**

It returns SE3_OK on success, otherwise see se3c1def.h

#### 6.5.2.14 uint16_t L1_set_user_PIN ( se3_session ∗ *s*, uint8_t ∗ *pin* )

This function is used to change the current user pin.

**Parameters**

| | | |
|---|---|---|
| `in` | *s* | Pointer to current se3_session, you must be logged in as admin to issue this command |
| `in` | *pin* | New pin to be set |

**Returns**

It returns SE3_OK on success, otherwise see se3c1def.h

# Index

AES data structures, 10
AES functions, 11
    B5_Aes256_Finit, 11
    B5_Aes256_Init, 11
    B5_Aes256_SetIV, 11
    B5_Aes256_Update, 13
AES Key, IV, Block Sizes, 8
    B5_AES_128, 8
    B5_AES_192, 8
    B5_AES_256, 8
    B5_AES_BLK_SIZE, 8
    B5_AES_IV_SIZE, 8
AES modes, 9
    B5_AES256_CBC_DEC, 9
    B5_AES256_CBC_ENC, 9
    B5_AES256_CFB_DEC, 9
    B5_AES256_CFB_ENC, 9
    B5_AES256_CTR, 9
    B5_AES256_ECB_DEC, 9
    B5_AES256_ECB_ENC, 9
    B5_AES256_OFB, 9
AES return values, 7
AccessLogin, 19
AlgorithmAvail, 21
    SE3_ALGO_AES_HMACSHA256, 21
    SE3_ALGO_AES_HMAC, 21
    SE3_ALGO_AES, 21
    SE3_ALGO_HMACSHA256, 21
    SE3_ALGO_SHA256, 21

B5_AES256_CBC_DEC
    AES modes, 9
B5_AES256_CBC_ENC
    AES modes, 9
B5_AES256_CFB_DEC
    AES modes, 9
B5_AES256_CFB_ENC
    AES modes, 9
B5_AES256_CTR
    AES modes, 9
B5_AES256_ECB_DEC
    AES modes, 9
B5_AES256_ECB_ENC
    AES modes, 9
B5_AES256_OFB
    AES modes, 9
B5_AES_128
    AES Key, IV, Block Sizes, 8
B5_AES_192
    AES Key, IV, Block Sizes, 8
B5_AES_256
    AES Key, IV, Block Sizes, 8
B5_AES_BLK_SIZE
    AES Key, IV, Block Sizes, 8
B5_AES_IV_SIZE
    AES Key, IV, Block Sizes, 8
B5_Aes256_Finit
    AES functions, 11
B5_Aes256_Init
    AES functions, 11
B5_Aes256_SetIV
    AES functions, 11
B5_Aes256_Update
    AES functions, 13
B5_CMAC_AES_128
    CMAC-AES Key, Blk Sizes, 14
B5_CMAC_AES_192
    CMAC-AES Key, Blk Sizes, 14
B5_CMAC_AES_256
    CMAC-AES Key, Blk Sizes, 14
B5_CMAC_AES_BLK_SIZE
    CMAC-AES Key, Blk Sizes, 14
B5_CmacAes256_Finit
    CMAC-AES functions, 17
B5_CmacAes256_Init
    CMAC-AES functions, 17
B5_CmacAes256_Reset
    CMAC-AES functions, 17
B5_CmacAes256_Sign
    CMAC-AES functions, 18
B5_CmacAes256_Update
    CMAC-AES functions, 18
B5_HmacSha256_Finit
    HMAC-SHA256 functions, 29
B5_HmacSha256_Init
    HMAC-SHA256 functions, 29
B5_HmacSha256_Update
    HMAC-SHA256 functions, 29
B5_Sha256_Finit
    SHA256 functions, 25
B5_Sha256_Init
    SHA256 functions, 25
B5_Sha256_Update
    SHA256 functions, 25
B5_tAesCtx, 31
    InitVector, 31
    mode, 31
    Nr, 31
    rk, 31

B5_tCmacAesCtx, 32
B5_tHmacSha256Ctx, 32
B5_tSha256Ctx, 32

CMAC-AES data structures, 16
CMAC-AES functions, 17
  B5_CmacAes256_Finit, 17
  B5_CmacAes256_Init, 17
  B5_CmacAes256_Reset, 17
  B5_CmacAes256_Sign, 18
  B5_CmacAes256_Update, 18
CMAC-AES Key, Blk Sizes, 14
  B5_CMAC_AES_128, 14
  B5_CMAC_AES_192, 14
  B5_CMAC_AES_256, 14
  B5_CMAC_AES_BLK_SIZE, 14
CMAC-AES return values, 15
crc16.h
  se3_crc16_update, 37

HMAC-SHA256 data structures, 28
HMAC-SHA256 functions, 29
  B5_HmacSha256_Finit, 29
  B5_HmacSha256_Init, 29
  B5_HmacSha256_Update, 29
HMAC-SHA256 return values, 27

InitVector
  B5_tAesCtx, 31

KeyOpEdit, 20
  SE3_KEY_OP_DELETE, 20
  SE3_KEY_OP_INSERT, 20
  SE3_KEY_OP_UPSERT, 20

L0.h
  L0_TXRX, 48
  L0_close, 46
  L0_discover_init, 46
  L0_discover_next, 46
  L0_discover_serialno, 46
  L0_echo, 47
  L0_factoryinit, 47
  L0_open, 47
L0_TXRX
  L0.h, 48
L0_close
  L0.h, 46
L0_discover_init
  L0.h, 46
L0_discover_next
  L0.h, 46
L0_discover_serialno
  L0.h, 46
L0_echo
  L0.h, 47
L0_factoryinit
  L0.h, 47
L0_open
  L0.h, 47
L1.h
  L1_crypto_init, 50
  L1_crypto_set_time, 50
  L1_crypto_update, 51
  L1_decrypt, 51
  L1_digest, 51
  L1_encrypt, 52
  L1_find_key, 52
  L1_get_algorithms, 53
  L1_key_edit, 53
  L1_key_list, 53
  L1_login, 54
  L1_logout, 54
  L1_set_admin_PIN, 54
  L1_set_user_PIN, 55
L1_crypto_init
  L1.h, 50
L1_crypto_set_time
  L1.h, 50
L1_crypto_update
  L1.h, 51
L1_decrypt
  L1.h, 51
L1_digest
  L1.h, 51
L1_encrypt
  L1.h, 52
L1_find_key
  L1.h, 52
L1_get_algorithms
  L1.h, 53
L1_key_edit
  L1.h, 53
L1_key_list
  L1.h, 53
L1_login
  L1.h, 54
L1_logout
  L1.h, 54
L1_set_admin_PIN
  L1.h, 54
L1_set_user_PIN
  L1.h, 55

mode
  B5_tAesCtx, 31

Nr
  B5_tAesCtx, 31

rk
  B5_tAesCtx, 31

SE3_ALGO_AES_HMACSHA256
  AlgorithmAvail, 21
SE3_ALGO_AES_HMAC
  AlgorithmAvail, 21
SE3_ALGO_AES

  AlgorithmAvail, 21
SE3_ALGO_HMACSHA256
  AlgorithmAvail, 21
SE3_ALGO_SHA256
  AlgorithmAvail, 21
SE3_ERR_ACCESS
  se3c1def.h, 44
SE3_ERR_AUTH
  se3c1def.h, 44
SE3_ERR_EXPIRED
  se3c1def.h, 44
SE3_ERR_MEMORY
  se3c1def.h, 44
SE3_ERR_PIN
  se3c1def.h, 44
SE3_ERR_RESOURCE
  se3c1def.h, 44
SE3_KEY_OP_DELETE
  KeyOpEdit, 20
SE3_KEY_OP_INSERT
  KeyOpEdit, 20
SE3_KEY_OP_UPSERT
  KeyOpEdit, 20
SHA256 data structures, 24
SHA256 digest and block sizes, 23
SHA256 functions, 25
  B5_Sha256_Finit, 25
  B5_Sha256_Init, 25
  B5_Sha256_Update, 25
SHA256 return values, 22
se3_algo_, 33
se3_common.h
  se3_nblocks, 38
  se3_req_len_data, 39
  se3_req_len_data_and_headers, 39
  se3_resp_len_data, 39
  se3_resp_len_data_and_headers, 39
se3_crc16_update
  crc16.h, 37
se3_device_, 33
se3_device_info_, 34
se3_disco_it_, 34
se3_discover_info_, 35
se3_drive_it_, 35
se3_file, 35
se3_key_, 35
se3_nblocks
  se3_common.h, 38
se3_payload_cryptoctx_, 36
se3_req_len_data
  se3_common.h, 39
se3_req_len_data_and_headers
  se3_common.h, 39
se3_resp_len_data
  se3_common.h, 39
se3_resp_len_data_and_headers
  se3_common.h, 39
se3_session_, 36
se3c1def.h
  SE3_ERR_ACCESS, 44
  SE3_ERR_AUTH, 44
  SE3_ERR_EXPIRED, 44
  SE3_ERR_MEMORY, 44
  SE3_ERR_PIN, 44
  SE3_ERR_RESOURCE, 44
src/Common/crc16.h, 37
src/Common/se3_common.h, 38
src/Common/se3c1def.h, 40
src/Host/L0.h, 45
src/Host/L1.h, 48