libtropic

C library for TROPIC devices

Version: 0.0.1

Git tag: v0.0.1

Tropic Square

July 25, 2024

tropicsquare

1 libtropic

1.1 Introduction

This library implements functionalities for interfacing applications with TROPIC01 device. It comes without any security claims and shall be used for evaluation purpose only.

Supported TROPIC devices:

Device	Comment
TROPIC01	First generation TROPIC01

1.2 Dependencies

Used build system is **cmake 3.21**:
```
$ sudo apt install cmake
```

Ceedling is used for running tests and creating code coverage report, install it like this:
```
# Install Ruby
$ sudo apt-get install ruby-full
# Ceedling install
$ gem install ceedling
# Code coverage tool used by Ceedling
$ pip install gcovr
```

1.3 Examples

A few examples of library's usage are placed in `examples/` folder.

1.4 Running tests

Make sure you have Ceedling installed (as described in Dependencies).

Expected version:
```
$ ceedling version
   Ceedling:: 0.31.1
      Unity:: 2.5.4
      CMock:: 2.5.4
 CException:: 1.3.3
```

Running tests and creating code coverage report:
```
$ ceedling gcov:all utils:gcov
```

1.5 Library configuration

See option() calls in root **CMakelists.txt** and check also how CMakeLists.txt looks in example projects.

1.5.1 Cryptography support

For certain operations on application's side, libtropic needs cryptography support. It is possible to choose a cryptography provider, because definitions of crypto functions are chosen during compilation.

Current default provider of cryptogprahy is `vendor/trezor_crypto`.
```
# Use trezor_crypto library:
option(TS_CRYPTO_TREZOR "Use trezor_crypto as a cryptography provider" ON)
```

1.6 Library overview

Tropic layer 1

This layer is processing raw data transfers.

Available L1 implementations are:

- SPI (libtropic on embedded target and Physical chip, or FPGA)
- TCP (libtropic on Unix and TROPIC01's model on Unix)
- Serialport (libtropic on embedded target and TROPIC01's model on Unix)

Use `option()` switch to enable libtropic support for a certain platform, have a look in examples.

If there is no support for a platform, user is expected to provide own implementation for weak functions in this layer.

Related code:

- ts_l1.c
- ts_l1.h

Tropic layer 2

This layer is responsible for executing l2 request/response functions.

Related code:

- ts_l2.c
- ts_l2.h

Tropic layer 3

This layer is preparing and parsing l3 commands/results, it uses l2 functions to send and receive payloads.

Related code:

- ts_l3.c
- ts_l3.h

libtropic

This is a highest abstraction of tropic chip functionalities. Library offers various calls to simplify tropic chip usage on a target platform:

Related code:

- libtropic.c
- libtropic.h

2 Module Index

2.1 Modules

Here is a list of all modules:

3 File Index

3.1 File List

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

4 Module Documentation

4.1 [PUBLIC API]

Modules

- ts_init
  Initialize device handle.
- ts_deinit
  Deinitialize device handle.
- ts_handshake
  Establish a secure session.
- ts_ping
  Test secure session.
- ts_random_get
  Get random bytes from TROPIC01.
- ECC functions
  Group of ECC commands.
- ts_get_info_cert
  Get device's certificate.

4.1.1 Detailed Description

Dear users, please use this API. It contains all functions you need to interface with TROPIC01 device.

4.2 ts_init

Initialize device handle.

Functions

- ts_ret_t ts_init (ts_handle_t *h)
  Initialize handle and transport layer.

4.2.1 Detailed Description

No more details available.

4.2.2 Function Documentation

4.2.2.1 ts_init() `ts_ret_t ts_init ( ts_handle_t * h )`

Parameters

h	Device's handle

Returns

TS_OK if success, otherwise returns other error code.

4.3 ts_deinit

Deinitialize device handle.

Functions

- ts_ret_t ts_deinit (ts_handle_t *h)

  Deinitialize handle and transport layer.

4.3.1 Detailed Description

No more details available.

4.3.2 Function Documentation

4.3.2.1 ts_deinit() `ts_ret_t ts_deinit ( ts_handle_t * h )`

Parameters

h	Device's handle

Returns

TS_OK if success, otherwise returns other error code.

4.4 ts_handshake

Establish a secure session.

Macros

- #define TS_L2_HANDSHAKE_REQ_PKEY_INDEX_PAIRING_KEY_SLOT_0 0

  Corresponds to S_{H0Pub}.
- #define TS_L2_HANDSHAKE_REQ_PKEY_INDEX_PAIRING_KEY_SLOT_1 1

  Corresponds to S_{H1Pub}.
- #define TS_L2_HANDSHAKE_REQ_PKEY_INDEX_PAIRING_KEY_SLOT_2 2

  Corresponds to S_{H2Pub}.
- #define TS_L2_HANDSHAKE_REQ_PKEY_INDEX_PAIRING_KEY_SLOT_3 3

  Corresponds to S_{H3Pub}.

Functions

- ts_ret_t ts_handshake (ts_handle_t *h, const uint8_t *stpub, const uint8_t pkey_index, const uint8_t *shipriv, const uint8_t *shipub)

  This function provides secure handshake functionality.

4.4.1 Detailed Description

After succesfull execution, gcm contexts in passed handle will have kcmd and kres keys set. From this point, device will accept L3 commands.

4.4.2 Function Documentation

4.4.2.1 ts_handshake()

```
ts_ret_t ts_handshake (
        ts_handle_t * h,
        const uint8_t * stpub,
        const uint8_t pkey_index,
        const uint8_t * shipriv,
        const uint8_t * shipub )
```

Parameters

h	Device's handle
stpub	STPUB from device's certificate
pkey_index	Index of pairing public key
shipriv	Secure host private key
shipub	Secure host public key

Returns

TS_OK if success, otherwise returns other error code.

4.5 ts_ping

Test secure session.

Macros

- #define PING_LEN_MAX L3_CMD_DATA_SIZE_MAX
  Maximal length of Ping command message.

Functions

- ts_ret_t ts_ping (ts_handle_t *h, const uint8_t *msg_out, uint8_t *msg_in, const uint16_t len)
  Test secure session by exchanging a message with chip.

4.5.1 Detailed Description

Message passed to this function will be encrypted with session keys and sent/received through secure channel.

4.5.2 Function Documentation

4.5.2.1 ts_ping()

```
ts_ret_t ts_ping (
        ts_handle_t * h,
        const uint8_t * msg_out,
        uint8_t * msg_in,
        const uint16_t len )
```

Parameters

h	Device's handle
msg_out	Ping message going out
msg_in	Ping message going in
len	Length of ping message

Returns

TS_OK if success, otherwise returns other error code.

4.6 ts_random_get

Get random bytes from TROPIC01.

Macros

- #define RANDOM_VALUE_GET_LEN_MAX L2_CHUNK_MAX_DATA_SIZE
  Maximum number of random bytes requested at once.

Functions

- ts_ret_t ts_random_get (ts_handle_t *h, uint8_t *buff, const uint16_t len)
  Get number of random bytes.

4.6.1 Detailed Description

This function provides access to random bytes generated by TROPIC01's random number generator

4.6.2 Function Documentation

4.6.2.1 ts_random_get()

```
ts_ret_t ts_random_get (
        ts_handle_t * h,
        uint8_t * buff,
        const uint16_t len )
```

Parameters

h	Device's handle
buff	Buffer
len	Number of random bytes

Returns

TS_OK if success, otherwise returns other error code.

4.7 ECC functions

Group of ECC commands.

Modules

- ts_ecc_key_generate
  Generate ECC key.
- ts_ecc_key_read
  Read ECC public key.
- ts_eddsa_sign
  Sign with TROPIC01.
- ts_ecdsa_sign
  Sign with TROPIC01.
- ts_ecc_key_erase
  Erase ECC key.

Macros

- #define ECC_SLOT_0 0

  ECC key slot 0.
- #define ECC_SLOT_1 1

  ECC key slot 1.
- #define ECC_SLOT_2 2

  ECC key slot 2.
- #define ECC_SLOT_3 3

  ECC key slot 3.
- #define ECC_SLOT_4 4

  ECC key slot 4.
- #define ECC_SLOT_5 5

  ECC key slot 5.
- #define ECC_SLOT_6 6

  ECC key slot 6.
- #define ECC_SLOT_7 7

  ECC key slot 7.
- #define ECC_SLOT_8 8

  ECC key slot 8.
- #define ECC_SLOT_9 9

  ECC key slot 9.
- #define ECC_SLOT_10 10

  ECC key slot 10.
- #define ECC_SLOT_11 11

  ECC key slot 11.
- #define ECC_SLOT_12 12

  ECC key slot 12.
- #define ECC_SLOT_13 13

  ECC key slot 13.
- #define ECC_SLOT_14 14

  ECC key slot 14.
- #define ECC_SLOT_15 15

  ECC key slot 15.
- #define ECC_SLOT_16 16

  ECC key slot 16.
- #define ECC_SLOT_17 17

  ECC key slot 17.
- #define ECC_SLOT_18 18

  ECC key slot 18.
- #define ECC_SLOT_19 19

  ECC key slot 19.
- #define ECC_SLOT_20 20

  ECC key slot 20.
- #define ECC_SLOT_21 21

  ECC key slot 21.
- #define ECC_SLOT_22 22

  ECC key slot 22.
- #define ECC_SLOT_23 23

  ECC key slot 23.
- #define ECC_SLOT_24 24

ECC key slot 24.

- #define ECC_SLOT_25 25
  ECC key slot 25.
- #define ECC_SLOT_26 26
  ECC key slot 26.
- #define ECC_SLOT_27 27
  ECC key slot 27.
- #define ECC_SLOT_28 28
  ECC key slot 28.
- #define ECC_SLOT_29 29
  ECC key slot 29.
- #define ECC_SLOT_30 30
  ECC key slot 30.
- #define ECC_SLOT_31 31
  ECC key slot 31.
- #define TS_L3_ECC_KEY_GENERATE_CURVE_P256 1
  P256 Curve - 64-byte long public key.
- #define TS_L3_ECC_KEY_GENERATE_CURVE_ED25519 2
  Ed25519 Curve - 32-byte long public key.

4.7.1 Detailed Description

4.8 ts_ecc_key_generate

Generate ECC key.

Functions

- ts_ret_t ts_ecc_key_generate (ts_handle_t *h, const uint8_t slot, const uint8_t curve)
  Generate ECC key in the device's ECC key slot.

4.8.1 Detailed Description

Generate ECC private key in internal slots inside of TROPIC01

4.8.2 Function Documentation

4.8.2.1 ts_ecc_key_generate()

```
ts_ret_t ts_ecc_key_generate (
        ts_handle_t * h,
        const uint8_t slot,
        const uint8_t curve )
```

Parameters

h	Device's handle
slot	Slot number ECC_SLOT_1 - ECC_SLOT_32
curve	Type of ECC curve. Use L3_ECC_KEY_GENERATE_CURVE_ED25519 or L3_ECC_KEY_GENERATE_CURVE_P256

Returns

TS_OK if success, otherwise returns other error code.

4.9 ts_ecc_key_read

Read ECC public key.

Functions

- ts_ret_t ts_ecc_key_read (ts_handle_t ∗h, const uint8_t slot, uint8_t ∗key, const int8_t keylen, uint8_t ∗curve, uint8_t ∗origin)
  Read ECC public key corresponding to a private key in device's slot.

4.9.1 Detailed Description

Read ECC public key corresponding to private key in TROPIC01's slot.

4.9.2 Function Documentation

4.9.2.1 ts_ecc_key_read()

```
ts_ret_t ts_ecc_key_read (
        ts_handle_t * h,
        const uint8_t slot,
        uint8_t * key,
        const int8_t keylen,
        uint8_t * curve,
        uint8_t * origin )
```

Parameters

h	Device's handle
slot	Slot number ECC_SLOT_1 - ECC_SLOT_32
key	Buffer for retrieving a key
keylen	Length of the key's buffer
curve	Will be filled by curve byte
origin	Will be filled by origin byte

Returns

TS_OK if success, otherwise returns other error code.

4.10 ts_eddsa_sign

Sign with TROPIC01.

Functions

- ts_ret_t ts_eddsa_sign (ts_handle_t *h, const uint8_t slot, const uint8_t *msg, const int16_t msg_len, uint8_t *rs, const int8_t rs_len)

  EdDSA sign message with a private key stored in TROPIC01 device.

4.10.1 Detailed Description

Use TROPIC01 to EdDSA sign a message with a private key in its slot

4.10.2 Function Documentation

4.10.2.1 ts_eddsa_sign()

```
ts_ret_t ts_eddsa_sign (
    ts_handle_t * h,
    const uint8_t slot,
    const uint8_t * msg,
    const int16_t msg_len,
    uint8_t * rs,
    const int8_t rs_len )
```

Parameters

h	Chip's handle
slot	Slot containing a private key, ECC_SLOT_1 - ECC_SLOT_32
msg	Buffer containing a message to sign, max length is 4096B
msg_len	Length of a message
rs	Buffer for storing a signature in a form of R and S bytes
rs_len	Length of rs buffer should be 64B

Returns

TS_OK if success, otherwise returns other error code.

4.11 ts_ecdsa_sign

Sign with TROPIC01.

Functions

- ts_ret_t ts_ecdsa_sign (ts_handle_t *h, const uint8_t slot, const uint8_t *msg_hash, const int16_t msg_hash_len, uint8_t *rs, const int8_t rs_len)

  ECDSA sign message with a private key stored in TROPIC01 device.

4.11.1 Detailed Description

Use TROPIC01 to ECDSA sign a message with a private key in its slot

4.11.2 Function Documentation

4.11.2.1 ts_ecdsa_sign()

```
ts_ret_t ts_ecdsa_sign (
        ts_handle_t * h,
        const uint8_t slot,
        const uint8_t * msg_hash,
        const int16_t msg_hash_len,
        uint8_t * rs,
        const int8_t rs_len )
```

Parameters

h	Device's handle
slot	Slot containing a private key, ECC_SLOT_1 - ECC_SLOT_32
msg	Buffer containing hash of a message
msg_len	Length of hash's buffer should be 32B
rs	Buffer for storing a signature in a form of R and S bytes
rs_len	Length of rs buffer should be 64B

Returns

TS_OK if success, otherwise returns other error code.

4.12 ts_ecc_key_erase

Erase ECC key.

Functions

- ts_ret_t ts_ecc_key_erase (ts_handle_t *h, const uint8_t slot)

  Erase ECC key from chip-s slot.

4.12.1 Detailed Description

Erase ECC private key in a corresponding slot inside of TROPIC01

4.12.2 Function Documentation

4.12.2.1 ts_ecc_key_erase()

```
ts_ret_t ts_ecc_key_erase (
    ts_handle_t * h,
    const uint8_t slot )
```

Parameters

h	Device's handle
slot	Slot number ECC_SLOT_1 - ECC_SLOT_32

Returns

TS_OK if success, otherwise returns other error code.

4.13 ts_get_info_cert

Get device's certificate.

Macros

- #define TS_L2_GET_INFO_REQ_OBJECT_ID_X509_CERTIFICATE 0
  The X.509 chip certificate read from I-Memory and signed by Tropic Square (max length of 512B)
- #define TS_L2_GET_INFO_REQ_OBJECT_ID_CHIP_ID 1
  The chip ID - the chip silicon revision and unique device ID (max length of 128B)
- #define TS_L2_GET_INFO_REQ_OBJECT_ID_RISCV_FW_VERSION 2
  The RISCV current running FW version (4 Bytes)
- #define TS_L2_GET_INFO_REQ_OBJECT_ID_SPECT_FW_VERSION 4
  The SPECT FW version (4 Bytes)
- #define TS_L2_GET_INFO_REQ_OBJECT_ID_FW_BANK 176
  The FW header read from the selected bank id (shown as an index). Supported only in Start-up mode.
- #define TS_L2_GET_INFO_REQ_BLOCK_INDEX_DATA_CHUNK_0_127 0
  Request for data bytes 0-127 of the object.
- #define TS_L2_GET_INFO_REQ_BLOCK_INDEX_DATA_CHUNK_128_255 1
  Request for data bytes 128-255 of the object (only needed for the X.509 certificate)
- #define TS_L2_GET_INFO_REQ_BLOCK_INDEX_DATA_CHUNK_256_383 2
  Request for data bytes 128-383 of object (only needed for the X.509 certificate)
- #define TS_L2_GET_INFO_REQ_BLOCK_INDEX_DATA_CHUNK_384_511 3
  Request for data bytes 384-511 of object (only needed for the X.509 certificate)
- #define TS_L2_GET_INFO_REQ_CERT_SIZE 512
  Maximal size of certificate.

Functions

- ts_ret_t ts_get_info_cert (ts_handle_t *h, uint8_t *cert, const int16_t max_len)
  Get device's certificate.
- ts_ret_t ts_cert_verify_and_parse (const uint8_t *cert, const int16_t max_len, uint8_t *stpub)
  Verify certificate chain and parse STPUB.

4.13.1 Detailed Description

Obtain X509 device's certificate from TROPIC01's I-config memory.

4.13.2 Function Documentation

4.13.2.1 ts_get_info_cert() `ts_ret_t ts_get_info_cert (`

```
ts_handle_t * h,
uint8_t * cert,
const int16_t max_len )
```

Parameters

h	Device's handle
cert	Certificate's buffer
max_len	Length of certificate's buffer

Returns

TS_OK if success, otherwise returns other error code.

4.13.2.2 ts_cert_verify_and_parse() `ts_ret_t ts_cert_verify_and_parse (`

```
const uint8_t * cert,
const int16_t max_len,
uint8_t * stpub )
```

Parameters

cert	Certificate in DER format
max_len	Len of certificate buffer
stpub	TROPIC01 STPUB, unique for each device

Returns

TS_OK if success, otherwise returns other error code.

4.14 [L2 functions]

Functions controlling l2 transmission.

Macros

- #define L2_STATUS_REQUEST_OK 0x01
- #define L2_STATUS_RESULT_OK 0x02
- #define L2_STATUS_REQUEST_CONT 0x03
- #define L2_STATUS_RESULT_CONT 0x04
- #define L2_STATUS_HSK_ERR 0x79
- #define L2_STATUS_NO_SESSION 0x7A
- #define L2_STATUS_TAG_ERR 0x7B
- #define L2_STATUS_CRC_ERR 0x7C
- #define L2_STATUS_UNKNOWN_ERR 0x7E
- #define L2_STATUS_GEN_ERR 0x7F
- #define L2_STATUS_NO_RESP 0xFF

Functions

- ts_ret_t ts_l2_frame_check (const uint8_t *frame)

  This function checks if incomming L2 frame is valid.
- ts_ret_t ts_l2_transfer (ts_handle_t *h)
- ts_ret_t ts_l2_encrypted_cmd (ts_handle_t *h)

  This function executes generic L3 command. It expects command's data correctly encrypted using keys created during previsously called ts_l2_handshake_req()

4.14.1 Detailed Description

Function used during l2 operation.

4.14.2 Macro Definition Documentation

4.14.2.1 L2_STATUS_REQUEST_OK `#define L2_STATUS_REQUEST_OK 0x01`

STATUS field value

4.14.2.2 L2_STATUS_RESULT_OK `#define L2_STATUS_RESULT_OK 0x02`

STATUS field value

4.14.2.3 L2_STATUS_REQUEST_CONT `#define L2_STATUS_REQUEST_CONT 0x03`

STATUS field value

4.14.2.4 L2_STATUS_RESULT_CONT `#define L2_STATUS_RESULT_CONT 0x04`

STATUS field value

4.14.2.5 L2_STATUS_HSK_ERR `#define L2_STATUS_HSK_ERR 0x79`

STATUS field value

4.14.2.6 L2_STATUS_NO_SESSION `#define L2_STATUS_NO_SESSION 0x7A`

STATUS field value

4.14.2.7 L2_STATUS_TAG_ERR `#define L2_STATUS_TAG_ERR 0x7B`

STATUS field value

4.14.2.8 L2_STATUS_CRC_ERR `#define L2_STATUS_CRC_ERR 0x7C`

STATUS field value

4.14.2.9 L2_STATUS_UNKNOWN_ERR `#define L2_STATUS_UNKNOWN_ERR 0x7E`

STATUS field value

4.14.2.10 L2_STATUS_GEN_ERR `#define L2_STATUS_GEN_ERR 0x7F`

STATUS field value

4.14.2.11 L2_STATUS_NO_RESP `#define L2_STATUS_NO_RESP 0xFF`

STATUS field value

4.14.3 Function Documentation

4.14.3.1 ts_l2_frame_check() `ts_ret_t ts_l2_frame_check ( const uint8_t * frame )`

Parameters

frame	

Returns

TS_OK if success, otherwise returns other error code.

4.14.3.2 ts_l2_transfer() `ts_ret_t ts_l2_transfer ( ts_handle_t * h )`

Parameters

h	Chip's handle

Returns

TS_OK if success, otherwise returns other error code.

4.14.3.3 ts_l2_encrypted_cmd() `ts_ret_t ts_l2_encrypted_cmd ( ts_handle_t * h )`

Parameters

h	Chip's handle

Returns

TS_OK if success, otherwise returns other error code.

4.15 [PRIVATE API]

Modules

- [L2 functions]
  Functions controlling l2 transmission.
- [L2 API]
  Data Link Layer.
- [L3 functions]
  Functions controlling l3 transmission.
- [L3 API]
  Secure Session Layer.

4.15.1 Detailed Description

Dear users, please DO NOT USE this API. Private API is used by libtropic internally.

4.16 [L2 API]

Data Link Layer.

Modules

- get_info_req
  Get some info.
- handshake_req
  Establish a secure session.
- encrypted_cmd_req
  Request to execute encrypted command.

4.16.1 Detailed Description

This layer uses unencrypted Request/Response packets

4.17 get_info_req

Get some info.

Data Structures

- struct l2_get_info_req_t
- struct l2_get_info_rsp_t

Macros

- #define TS_L2_GET_INFO_REQ_ID 0x01
  Command ID.
- #define TS_L2_GET_INFO_REQ_LEN 0x02
  Length of this request.

4.17.1 Detailed Description

Get some info from the device

4.17.2 Data Structure Documentation

Data Fields

u8	req_id	
u8	req_len	
u8	obj_id	
u8	block_index	
uint8_t	crc[2]	

4.17.2.1 struct l2_get_info_req_t

Data Fields

u8	chip_status	
u8	status	
u8	rsp_len	
u8	data[128]	
u8	crc[2]	

4.17.2.2 struct l2_get_info_rsp_t

4.18 handshake_req

Establish a secure session.

Data Structures

- struct l2_handshake_req_t
- struct l2_handshake_rsp_t

Macros

- #define TS_L2_HANDSHAKE_REQ_ID 0x02

  Command ID.
- #define TS_L2_HANDSHAKE_REQ_LEN 0x21

  Length of this request.

4.18.1 Detailed Description

Request to execute a Secure Channel Handshake and establish a new Secure Channel Session (TROPIC01 moves to Secure Channel Mode).

4.18.2 Data Structure Documentation

Data Fields

u8	req_id	
u8	req_len	
uint8_t	e_hpub[32]	
uint8_t	pkey_index	
uint8_t	crc[2]	

4.18.2.1 struct l2_handshake_req_t

Data Fields

u8	chip_status	
u8	status	
u8	rsp_len	
uint8_t	e_tpub[32]	
uint8_t	t_auth[16]	
u8	crc[2]	

4.18.2.2 struct l2_handshake_rsp_t

4.19 encrypted_cmd_req

Request to execute encrypted command.

Data Structures

- struct l2_encrypted_cmd_req_t
- struct l2_encrypted_cmd_rsp_t

Macros

- #define TS_L2_ENCRYPTED_CMD_REQ_ID 0x04

  Command ID.

4.19.1 Detailed Description

Transmission of one encrypted command may consist of multiple encrypted_cmd_req requests/responses

4.19.2 Data Structure Documentation

Data Fields

u8	req_id	
u8	req_len	
uint8_t	body[L2_CHUNK_MAX_DATA_SIZE]	
uint8_t	crc[2]	

4.19.2.1 struct l2_encrypted_cmd_req_t

Data Fields

u8	chip_status	
u8	status	

Data Fields

u8	rsp_len	
uint8_t	body[L2_CHUNK_MAX_DATA_SIZE]	
u8	crc[2]	

4.19.2.2 struct l2_encrypted_cmd_rsp_t

4.20 [L3 functions]

Functions controlling l3 transmission.

Macros

- #define L3_RESULT_OK 0xC3u
- #define L3_RESULT_FAIL 0x3Cu
- #define L3_RESULT_UNAUTHORIZED 0x01u
- #define L3_RESULT_INVALID_CMD 0x02u

Functions

- ts_ret_t ts_l3_nonce_init (ts_handle_t *h)
- ts_ret_t ts_l3_nonce_increase (ts_handle_t *h)
- ts_ret_t ts_l3_cmd (ts_handle_t *h)

4.20.1 Detailed Description

Function used during l3 operation.

4.20.2 Macro Definition Documentation

4.20.2.1 L3_RESULT_OK `#define L3_RESULT_OK 0xC3u`

L3 RESULT field Value

4.20.2.2 L3_RESULT_FAIL `#define L3_RESULT_FAIL 0x3Cu`

L3 RESULT field Value

4.20.2.3 L3_RESULT_UNAUTHORIZED `#define L3_RESULT_UNAUTHORIZED 0x01u`

L3 RESULT field Value

4.20.2.4 L3_RESULT_INVALID_CMD `#define L3_RESULT_INVALID_CMD 0x02u`

L3 RESULT field Value

4.20.3 Function Documentation

4.20.3.1 ts_l3_nonce_init() `ts_ret_t ts_l3_nonce_init ( ts_handle_t * h )`

Initializes nonce in handle to 0. This function is used during secure handshake.

Parameters

h	Chip's handle

Returns

TS_OK if success, otherwise returns other error code.

4.20.3.2 ts_l3_nonce_increase() `ts_ret_t ts_l3_nonce_increase ( ts_handle_t * h )`

Increases by one nonce stored in handle. This function is used after successfull reception of L3 response. , uint16_t cmd_len,

Parameters

h	Chip's handle

Returns

TS_OK if success, otherwise returns other error code.

4.20.3.3 ts_l3_cmd() `ts_ret_t ts_l3_cmd ( ts_handle_t * h )`

Perform l3 encrypted command operation.

Parameters

h	Chip's handle

Returns

TS_OK if success, otherwise returns other error code.

4.21 [L3 API]

Secure Session Layer.

Modules

- Ping
  Ping command.
- Random_Value_Get
  Get random bytes from TROPIC01.
- ECC_Key_Generate
  Generate ECC key.
- ECC_Key_Read
  Read ECC key.
- ECC_Key_Erase
  Erase ECC key.
- EDDSA_Sign
  Sign with EDDSA key.
- ECDSA_Sign
  Sign with ECDSA key.

4.21.1 Detailed Description

This layer uses encrypted Command/Result packets

4.22 Ping

Ping command.

Data Structures

- struct ts_l3_ping_cmd_t
- struct ts_l3_ping_res_t

Macros

- #define TS_L3_PING_CMD 0x01
  Command ID.

4.22.1 Detailed Description

Run arbitrary message through the secure session to verify that all works.

4.22.2 Data Structure Documentation

Data Fields

u16	packet_size	L3 packet size
u8	command	L3 Command Identifier
u8	data[4096]	Data transferred from host into chip
u8	tag[16]	L3 tag

4.22.2.1 struct ts_l3_ping_cmd_t

Data Fields

u16	packet_size	L3 packet size
u8	result	L3 Result status indication
u8	data[4096]	Data transferred from chip into host
u8	tag[16]	

4.22.2.2 struct ts_l3_ping_res_t

4.23 Random_Value_Get

Get random bytes from TROPIC01.

Data Structures

- struct ts_l3_random_value_get_cmd_t
- struct ts_l3_random_value_get_res_t

Macros

- #define TS_L3_RANDOM_VALUE_GET_CMD 0x50
  Command ID.
- #define TS_L3_RANDOM_VALUE_GET_CMD_SIZE 2
  Command length.

4.23.1 Detailed Description

Uses TRNG2 inside of TROPIC01

4.23.2 Data Structure Documentation

Data Fields

u16	packet_size	
u8	command	L3 Command Identifier
u8	n_bytes	The number of random bytes to get
u8	tag[16]	L3 tag

4.23.2.1 struct ts_l3_random_value_get_cmd_t

Data Fields

u16	packet_size	L3 packet size
u8	result	L3 Result status indication
u8	padding[3]	The padding by dummy data
u8	random_data[255]	The random data from TRNG2 in the number of bytes specified in the N_BYTES L3 Field
u8	tag[16]	L3 tag

4.23.2.2 struct ts_l3_random_value_get_res_t

4.24 ECC_Key_Generate

Generate ECC key.

Data Structures

- struct ts_l3_ecc_key_generate_cmd_t
- struct ts_l3_ecc_key_generate_res_t

Macros

- #define TS_L3_ECC_KEY_GENERATE_CMD 0x60

  ECC_Key_generate command ID.
- #define TS_L3_ECC_KEY_GENERATE_CMD_SIZE 0x04u

  ECC_Key_generate command size.
- #define TS_L3_ECC_KEY_GENERATE_SLOT_MIN 0

  ECC_Key_generate min slot number.
- #define TS_L3_ECC_KEY_GENERATE_SLOT_MAX 31

  ECC_Key_generate max slot number.

4.24.1 Detailed Description

Generate ECC private key inside of TROPIC01's memory slot.

4.24.2 Data Structure Documentation

Data Fields

u16	packet_size	L3 packet size
u8	command	L3 Command Identifier
u16	slot	The slot to write the generated key. Valid values are 0 - 31
u8	curve	The Elliptic Curve the key is generated from
u8	tag[16]	L3 tag

4.24.2.1 struct ts_l3_ecc_key_generate_cmd_t

Data Fields

u16	packet_size	L3 packet size
u8	result	L3 Result status indication
u8	tag[16]	L3 tag

4.24.2.2 struct ts_l3_ecc_key_generate_res_t

4.25 ECC_Key_Read

Read ECC key.

Data Structures

- struct ts_l3_ecc_key_read_cmd_t
- struct ts_l3_ecc_key_read_res_t

Macros

- #define TS_L3_ECC_KEY_READ_CMD 0x62

  Command ID.
- #define TS_L3_ECC_KEY_READ_CMD_SIZE 0x03

  Command length.

4.25.1 Detailed Description

Read ECC public key which corresponds to TROPIC01's memory slot.

4.25.2 Data Structure Documentation

Data Fields

u16	packet_size	L3 packet size
u8	command	L3 Command Identifier
u16	slot	ECC Key slot
u8	tag[16]	L3 tag

4.25.2.1 struct ts_l3_ecc_key_read_cmd_t

Data Fields

u16	packet_size	L3 packet size
u8	result	L3 Result status indication
u8	curve	
u8	origin	
u8	padding[13]	Padding
u8	pub_key[64]	The public key from the ECC Key slot as specified in the SLOT L3 Field
u8	tag[16]	L3 tag

4.25.2.2 struct ts_l3_ecc_key_read_res_t

4.26 ECC_Key_Erase

Erase ECC key.

Data Structures

- struct ts_l3_ecc_key_erase_cmd_t
- struct ts_l3_ecc_key_erase_res_t

Macros

- #define TS_L3_ECC_KEY_ERASE_CMD 0x63u

  Command ID.
- #define TS_L3_ECC_KEY_ERASE_CMD_SIZE 0x03u

  Command length.

4.26.1 Detailed Description

Erase ECC private key in a given TROIC01's memory slot

4.26.2 Data Structure Documentation

Data Fields

u16	packet_size	L3 packet size
u8	command	L3 Command Identifier
u16	slot	ECC Key slot
u8	tag[16]	L3 tag

4.26.2.1 struct ts_l3_ecc_key_erase_cmd_t

Data Fields

u16	packet_size	L3 packet size
u8	result	L3 Result status indication
u8	tag[16]	L3 tag

4.26.2.2 struct ts_l3_ecc_key_erase_res_t

4.27 EDDSA_Sign

Sign with EDDSA key.

Data Structures

- struct ts_l3_eddsa_sign_cmd_t
- struct ts_l3_eddsa_sign_res_t

Macros

- #define TS_L3_EDDSA_SIGN_CMD 0x71

  Command ID.
- #define TS_L3_EDDSA_SIGN_CMD_SIZE 0x10u

  Command length.
- #define **TS_L3_EDDSA_SIGN_MSG_LEN_MIN** 0x01
- #define **TS_L3_EDDSA_SIGN_MSG_LEN_MAX** 4096

4.27.1 Detailed Description

Use private key from a given slot to sign a message

4.27.2 Data Structure Documentation

Data Fields

u16	packet_size	L3 packet size
u8	command	L3 Command Identifier
u16	slot	ECC Key slot
u8	padding[13]	Padding
u8	msg[4096]	Message to sign
u8	tag[16]	L3 tag

4.27.2.1 struct ts_l3_eddsa_sign_cmd_t

Data Fields

u16	packet_size	L3 packet size
u8	result	L3 Result status indication
u8	padding[15]	Padding
u8	r[32]	EdDSA signature - The R part
u8	s[32]	EdDSA signature - The S part
u8	tag[16]	L3 tag

4.27.2.2 struct ts_l3_eddsa_sign_res_t

4.28 ECDSA_Sign

Sign with ECDSA key.

Data Structures

- struct ts_l3_ecdsa_sign_cmd_t
- struct ts_l3_ecdsa_sign_res_t

Macros

- #define TS_L3_ECDSA_SIGN 0x70

  Command ID.
- #define TS_L3_ECDSA_SIGN_CMD_SIZE 0x30u

  Command length.
- #define **TS_L3_ECDSA_SIGN_MSG_HASH_LEN** 0x20

4.28.1 Detailed Description

Use private key from a given slot to sign a message

4.28.2 Data Structure Documentation

Data Fields

u16	packet_size	L3 packet size
u8	command	L3 Command Identifier
u16	slot	ECC Key slot
u8	padding[13]	Padding
u8	msg_hash[32]	Message to sign
u8	tag[16]	L3 tag

4.28.2.1 struct ts_l3_ecdsa_sign_cmd_t

Data Fields

u16	packet_size	L3 packet size
u8	result	L3 Result status indication
u8	padding[15]	Padding
u8	r[32]	EdDSA signature - The R part
u8	s[32]	EdDSA signature - The S part
u8	tag[16]	L3 tag

4.28.2.2 struct ts_l3_ecdsa_sign_res_t

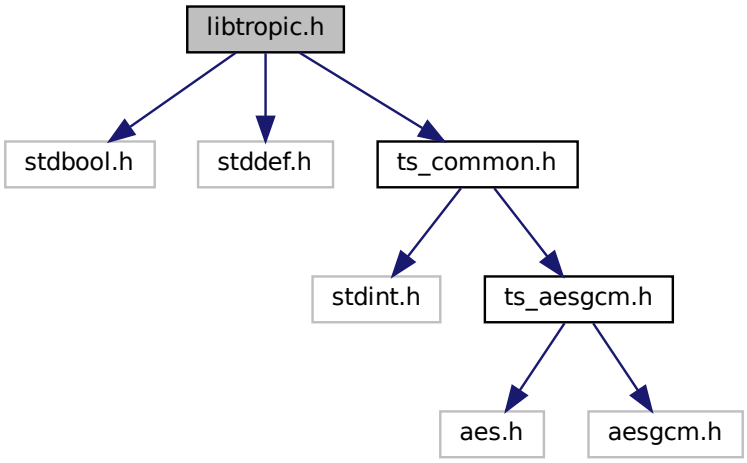
5 File Documentation

5.1 libtropic.h File Reference

libtropic header file

```
#include <stdbool.h>
#include <stddef.h>
#include "ts_common.h"
```

Include dependency graph for libtropic.h:



Macros

- #define TS_L2_HANDSHAKE_REQ_PKEY_INDEX_PAIRING_KEY_SLOT_0 0

  Corresponds to S_{H0Pub}.
- #define TS_L2_HANDSHAKE_REQ_PKEY_INDEX_PAIRING_KEY_SLOT_1 1

  Corresponds to S_{H1Pub}.
- #define TS_L2_HANDSHAKE_REQ_PKEY_INDEX_PAIRING_KEY_SLOT_2 2

Corresponds to S_{H2Pub}.

- #define TS_L2_HANDSHAKE_REQ_PKEY_INDEX_PAIRING_KEY_SLOT_3 3

  Corresponds to S_{H3Pub}.
- #define PING_LEN_MAX L3_CMD_DATA_SIZE_MAX

  Maximal length of Ping command message.
- #define RANDOM_VALUE_GET_LEN_MAX L2_CHUNK_MAX_DATA_SIZE

  Maximum number of random bytes requested at once.
- #define ECC_SLOT_0 0

  ECC key slot 0.
- #define ECC_SLOT_1 1

  ECC key slot 1.
- #define ECC_SLOT_2 2

  ECC key slot 2.
- #define ECC_SLOT_3 3

  ECC key slot 3.
- #define ECC_SLOT_4 4

  ECC key slot 4.
- #define ECC_SLOT_5 5

  ECC key slot 5.
- #define ECC_SLOT_6 6

  ECC key slot 6.
- #define ECC_SLOT_7 7

  ECC key slot 7.
- #define ECC_SLOT_8 8

  ECC key slot 8.
- #define ECC_SLOT_9 9

  ECC key slot 9.
- #define ECC_SLOT_10 10

  ECC key slot 10.
- #define ECC_SLOT_11 11

  ECC key slot 11.
- #define ECC_SLOT_12 12

  ECC key slot 12.
- #define ECC_SLOT_13 13

  ECC key slot 13.
- #define ECC_SLOT_14 14

  ECC key slot 14.
- #define ECC_SLOT_15 15

  ECC key slot 15.
- #define ECC_SLOT_16 16

  ECC key slot 16.
- #define ECC_SLOT_17 17

  ECC key slot 17.
- #define ECC_SLOT_18 18

  ECC key slot 18.
- #define ECC_SLOT_19 19

  ECC key slot 19.
- #define ECC_SLOT_20 20

  ECC key slot 20.
- #define ECC_SLOT_21 21

  ECC key slot 21.

- #define ECC_SLOT_22 22

  ECC key slot 22.
- #define ECC_SLOT_23 23

  ECC key slot 23.
- #define ECC_SLOT_24 24

  ECC key slot 24.
- #define ECC_SLOT_25 25

  ECC key slot 25.
- #define ECC_SLOT_26 26

  ECC key slot 26.
- #define ECC_SLOT_27 27

  ECC key slot 27.
- #define ECC_SLOT_28 28

  ECC key slot 28.
- #define ECC_SLOT_29 29

  ECC key slot 29.
- #define ECC_SLOT_30 30

  ECC key slot 30.
- #define ECC_SLOT_31 31

  ECC key slot 31.
- #define TS_L3_ECC_KEY_GENERATE_CURVE_P256 1

  P256 Curve - 64-byte long public key.
- #define TS_L3_ECC_KEY_GENERATE_CURVE_ED25519 2

  Ed25519 Curve - 32-byte long public key.
- #define TS_L2_GET_INFO_REQ_OBJECT_ID_X509_CERTIFICATE 0

  The X.509 chip certificate read from I-Memory and signed by Tropic Square (max length of 512B)
- #define TS_L2_GET_INFO_REQ_OBJECT_ID_CHIP_ID 1

  The chip ID - the chip silicon revision and unique device ID (max length of 128B)
- #define TS_L2_GET_INFO_REQ_OBJECT_ID_RISCV_FW_VERSION 2

  The RISCV current running FW version (4 Bytes)
- #define TS_L2_GET_INFO_REQ_OBJECT_ID_SPECT_FW_VERSION 4

  The SPECT FW version (4 Bytes)
- #define TS_L2_GET_INFO_REQ_OBJECT_ID_FW_BANK 176

  The FW header read from the selected bank id (shown as an index). Supported only in Start-up mode.
- #define TS_L2_GET_INFO_REQ_BLOCK_INDEX_DATA_CHUNK_0_127 0

  Request for data bytes 0-127 of the object.
- #define TS_L2_GET_INFO_REQ_BLOCK_INDEX_DATA_CHUNK_128_255 1

  Request for data bytes 128-255 of the object (only needed for the X.509 certificate)
- #define TS_L2_GET_INFO_REQ_BLOCK_INDEX_DATA_CHUNK_256_383 2

  Request for data bytes 128-383 of object (only needed for the X.509 certificate)
- #define TS_L2_GET_INFO_REQ_BLOCK_INDEX_DATA_CHUNK_384_511 3

  Request for data bytes 384-511 of object (only needed for the X.509 certificate)
- #define TS_L2_GET_INFO_REQ_CERT_SIZE 512

  Maximal size of certificate.

Functions

- ts_ret_t ts_init (ts_handle_t *h)

  Initialize handle and transport layer.
- ts_ret_t ts_deinit (ts_handle_t *h)

  Deinitialize handle and transport layer.
- ts_ret_t ts_handshake (ts_handle_t *h, const uint8_t *stpub, const uint8_t pkey_index, const uint8_t *shipriv, const uint8_t *shipub)

  This function provides secure handshake functionality.
- ts_ret_t ts_ping (ts_handle_t *h, const uint8_t *msg_out, uint8_t *msg_in, const uint16_t len)

  Test secure session by exchanging a message with chip.
- ts_ret_t ts_random_get (ts_handle_t *h, uint8_t *buff, const uint16_t len)

  Get number of random bytes.
- ts_ret_t ts_ecc_key_generate (ts_handle_t *h, const uint8_t slot, const uint8_t curve)

  Generate ECC key in the device's ECC key slot.
- ts_ret_t ts_ecc_key_read (ts_handle_t *h, const uint8_t slot, uint8_t *key, const int8_t keylen, uint8_t *curve, uint8_t *origin)

  Read ECC public key corresponding to a private key in device's slot.
- ts_ret_t ts_eddsa_sign (ts_handle_t *h, const uint8_t slot, const uint8_t *msg, const int16_t msg_len, uint8_t *rs, const int8_t rs_len)

  EdDSA sign message with a private key stored in TROPIC01 device.
- ts_ret_t ts_ecdsa_sign (ts_handle_t *h, const uint8_t slot, const uint8_t *msg_hash, const int16_t msg_hash_len, uint8_t *rs, const int8_t rs_len)

  ECDSA sign message with a private key stored in TROPIC01 device.
- ts_ret_t ts_ecc_key_erase (ts_handle_t *h, const uint8_t slot)

  Erase ECC key from chip-s slot.
- ts_ret_t ts_get_info_cert (ts_handle_t *h, uint8_t *cert, const int16_t max_len)

  Get device's certificate.
- ts_ret_t ts_cert_verify_and_parse (const uint8_t *cert, const int16_t max_len, uint8_t *stpub)

  Verify certificate chain and parse STPUB.

5.1.1 Detailed Description

Author

Tropic Square s.r.o.

5.1.2 Function Documentation

5.1.2.1 ts_init() `ts_ret_t ts_init ( ts_handle_t * h )`

Parameters

h	Device's handle

Returns

TS_OK if success, otherwise returns other error code.

5.1.2.2 ts_deinit()

```
ts_ret_t ts_deinit (
        ts_handle_t * h )
```

Parameters

h	Device's handle

Returns

TS_OK if success, otherwise returns other error code.

5.1.2.3 ts_handshake()

```
ts_ret_t ts_handshake (
        ts_handle_t * h,
        const uint8_t * stpub,
        const uint8_t pkey_index,
        const uint8_t * shipriv,
        const uint8_t * shipub )
```

Parameters

h	Device's handle
stpub	STPUB from device's certificate
pkey_index	Index of pairing public key
shipriv	Secure host private key
shipub	Secure host public key

Returns

TS_OK if success, otherwise returns other error code.

5.1.2.4 ts_ping()

```
ts_ret_t ts_ping (
        ts_handle_t * h,
        const uint8_t * msg_out,
        uint8_t * msg_in,
        const uint16_t len )
```

Parameters

h	Device's handle
msg_out	Ping message going out
msg_in	Ping message going in
len	Length of ping message

Returns

TS_OK if success, otherwise returns other error code.

5.1.2.5 ts_random_get()

```
ts_ret_t ts_random_get (
        ts_handle_t * h,
        uint8_t * buff,
        const uint16_t len )
```

Parameters

h	Device's handle
buff	Buffer
len	Number of random bytes

Returns

TS_OK if success, otherwise returns other error code.

5.1.2.6 ts_ecc_key_generate()

```
ts_ret_t ts_ecc_key_generate (
        ts_handle_t * h,
        const uint8_t slot,
        const uint8_t curve )
```

Parameters

h	Device's handle
slot	Slot number ECC_SLOT_1 - ECC_SLOT_32
curve	Type of ECC curve. Use L3_ECC_KEY_GENERATE_CURVE_ED25519 or L3_ECC_KEY_GENERATE_CURVE_P256

Returns

TS_OK if success, otherwise returns other error code.

5.1.2.7 ts_ecc_key_read()

```
ts_ret_t ts_ecc_key_read (
        ts_handle_t * h,
        const uint8_t slot,
        uint8_t * key,
        const int8_t keylen,
        uint8_t * curve,
        uint8_t * origin )
```

Parameters

h	Device's handle
slot	Slot number ECC_SLOT_1 - ECC_SLOT_32
key	Buffer for retrieving a key
keylen	Length of the key's buffer
curve	Will be filled by curve byte
origin	Will be filled by origin byte

Returns

TS_OK if success, otherwise returns other error code.

5.1.2.8 ts_eddsa_sign()

```
ts_ret_t ts_eddsa_sign (
        ts_handle_t * h,
        const uint8_t slot,
        const uint8_t * msg,
        const int16_t msg_len,
        uint8_t * rs,
        const int8_t rs_len )
```

Parameters

h	Chip's handle
slot	Slot containing a private key, ECC_SLOT_1 - ECC_SLOT_32
msg	Buffer containing a message to sign, max length is 4096B
msg_len	Length of a message
rs	Buffer for storing a signature in a form of R and S bytes
rs_len	Length of rs buffer should be 64B

Returns

TS_OK if success, otherwise returns other error code.

5.1.2.9 ts_ecdsa_sign()

```
ts_ret_t ts_ecdsa_sign (
        ts_handle_t * h,
        const uint8_t slot,
        const uint8_t * msg_hash,
        const int16_t msg_hash_len,
        uint8_t * rs,
        const int8_t rs_len )
```

Parameters

h	Device's handle
slot	Slot containing a private key, ECC_SLOT_1 - ECC_SLOT_32

Parameters

msg	Buffer containing hash of a message
msg_len	Length of hash's buffer should be 32B
rs	Buffer for storing a signature in a form of R and S bytes
rs_len	Length of rs buffer should be 64B

Returns

TS_OK if success, otherwise returns other error code.

5.1.2.10 ts_ecc_key_erase()

```
ts_ret_t ts_ecc_key_erase (
        ts_handle_t * h,
        const uint8_t slot )
```

Parameters

h	Device's handle
slot	Slot number ECC_SLOT_1 - ECC_SLOT_32

Returns

TS_OK if success, otherwise returns other error code.

5.1.2.11 ts_get_info_cert()

```
ts_ret_t ts_get_info_cert (
        ts_handle_t * h,
        uint8_t * cert,
        const int16_t max_len )
```

Parameters

h	Device's handle
cert	Certificate's buffer
max_len	Length of certificate's buffer

Returns

TS_OK if success, otherwise returns other error code.

5.1.2.12 ts_cert_verify_and_parse()

```
ts_ret_t ts_cert_verify_and_parse (
        const uint8_t * cert,
        const int16_t max_len,
        uint8_t * stpub )
```

Parameters

cert	Certificate in DER format
max_len	Len of certificate buffer
stpub	TROPIC01 STPUB, unique for each device

Returns

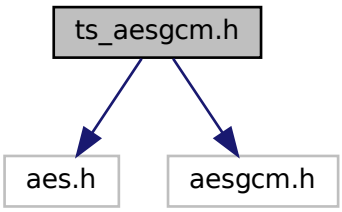
TS_OK if success, otherwise returns other error code.

5.2 ts_aesgcm.h File Reference

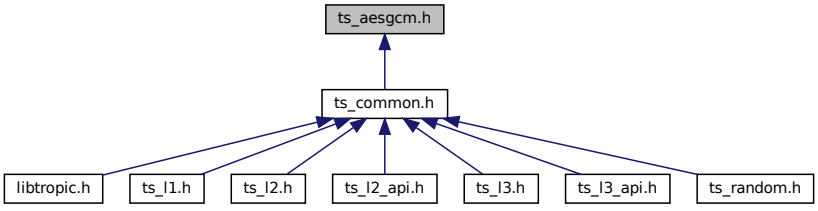
AESGCM function declarations.

```
#include "aes.h"
#include "aesgcm.h"
```

Include dependency graph for ts_aesgcm.h:



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



Data Structures

- struct ts_aes_gcm_ctx

Typedefs

- typedef struct ts_aes_gcm_ctx **ts_aes_gcm_ctx_t**

Functions

- int ts_aesgcm_init_and_key (void *ctx, const uint8_t *key, uint32_t key_len)
- int ts_aesgcm_encrypt (void *ctx, const uint8_t *iv, uint32_t iv_len, const uint8_t *aad, uint32_t aad_len, uint8_t *msg, uint32_t msg_len, uint8_t *tag, uint32_t tag_len)
- int ts_aesgcm_decrypt (void *ctx, const uint8_t *iv, uint32_t iv_len, const uint8_t *aad, uint32_t aad_len, uint8_t *msg, uint32_t msg_len, const uint8_t *tag, uint32_t tag_len)
- int ts_aesgcm_end (void *ctx)

5.2.1 Detailed Description

Author

Tropic Square s.r.o.

5.2.2 Data Structure Documentation

5.2.2.1 struct ts_aes_gcm_ctx

5.2.3 Function Documentation

5.2.3.1 ts_aesgcm_init_and_key()

```
int ts_aesgcm_init_and_key (
        void * ctx,
        const uint8_t * key,
        uint32_t key_len )
```

This function initializes AES GCM context with keys

Parameters

ctx	AESGCM context structure
key	The key value
key_len	Length of key in bytes

Returns

TS_OK if success, otherwise returns other error code.

5.2.3.2 ts_aesgcm_encrypt()

```
int ts_aesgcm_encrypt (
        void * ctx,
        const uint8_t * iv,
        uint32_t iv_len,
        const uint8_t * aad,
        uint32_t aad_len,
        uint8_t * msg,
        uint32_t msg_len,
        uint8_t * tag,
        uint32_t tag_len )
```

This function decrypts data. It expect initialized context with valid keys.

Parameters

ctx	AESGCM context structure
iv	The initialisation vector
iv_len	Length of initialization vector in bytes
aad	The header buffer
aad_len	Length of header buffer in bytes
msg	Message buffer
msg_len	Length of message in bytes
tag	The tag buffer
tag_len	Length of tag buffer in bytes

Returns

TS_OK if success, otherwise returns other error code.

5.2.3.3 ts_aesgcm_decrypt()

```
int ts_aesgcm_decrypt (
        void * ctx,
        const uint8_t * iv,
        uint32_t iv_len,
        const uint8_t * aad,
        uint32_t aad_len,
        uint8_t * msg,
        uint32_t msg_len,
        const uint8_t * tag,
        uint32_t tag_len )
```

This function decrypts data. It expect initialized context with valid keys.

Parameters

ctx	AESGCM context structure
iv	The initialisation vector
iv_len	Length of initialization vector in bytes
aad	The header buffer
aad_len	Length of header buffer in bytes
msg	Message buffer
msg_len	Length of message in bytes
tag	The tag buffer
tag_len	Length of tag buffer in bytes

Returns

TS_OK if success, otherwise returns other error code.

5.2.3.4 ts_aesgcm_end() `int ts_aesgcm_end ( void * ctx )`

This function clears AES GCM context

Parameters

ctx	AESGCM context structure

Returns

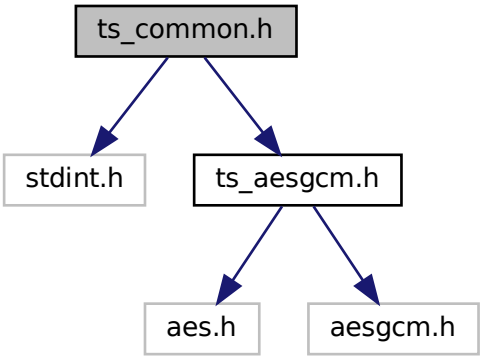
TS_OK if success, otherwise returns other error code.

5.3 ts_common.h File Reference

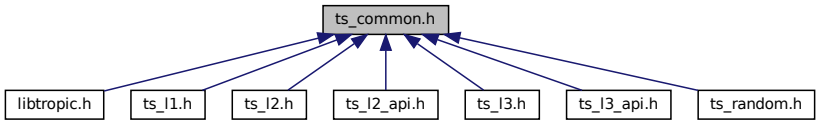
Shared definitions and functions commonly used by more layers.

```
#include "stdint.h"
#include "ts_aesgcm.h"
```

Include dependency graph for ts_common.h:



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



Data Structures

- struct l3_frame_t
- struct ts_handle_t

Macros

- #define **UNUSED**(x) (void)(x)
- #define **SESSION_ON** 0xAA55AA55
- #define **SESSION_OFF** 0x00000000
- #define **L3_ID_SIZE** 1u
- #define **L3_TAG_SIZE** 16u
- #define **L3_KEY_SIZE** 32u
- #define **L3_IV_SIZE** 12u
- #define **L3_PACKET_SIZE_SIZE** sizeof(uint16_t)
- #define **L3_CMD_ID_SIZE** (1)
- #define **L3_CMD_DATA_SIZE_MAX** (4095)
- #define L2_CHUNK_MAX_DATA_SIZE 252u
- #define **L2_CHUNK_MAX_FRAME_SIZE** (1 + 1 + L2_CHUNK_MAX_DATA_SIZE + 2)
- #define TS_L1_LEN_MIN 1
- #define TS_L1_LEN_MAX (1 + 1 + 1 + L2_CHUNK_MAX_DATA_SIZE + 2)
- #define L3_PACKET_MAX_SIZE (L3_CMD_ID_SIZE + L3_CMD_DATA_SIZE_MAX)
- #define L3_FRAME_MAX_SIZE (L3_PACKET_SIZE_SIZE + L3_PACKET_MAX_SIZE + L3_TAG_SIZE)

Typedefs

- typedef uint8_t **u8**
- typedef uint16_t **u16**
- typedef struct ts_handle_t ts_handle_t

Enumerations

- enum ts_ret_t {
  TS_OK , TS_FAIL , TS_PARAM_ERR , TS_L1_SPI_ERROR ,
  TS_L1_DATA_LEN_ERROR , TS_L1_CHIP_STARTUP_MODE , TS_L1_CHIP_ALARM_MODE , TS_L1_CHIP_BUSY ,
  TS_L2_IN_CRC_ERR , **TS_L2_REQ_CONT** , **TS_L2_RES_CONT** , **TS_L2_HSK_ERR** ,
  TS_L2_NO_SESSION , **TS_L2_TAG_ERR** , **TS_L2_CRC_ERR** , **TS_L2_GEN_ERR** ,
  TS_L2_NO_RESP , **TS_L2_UNKNOWN_REQ** , TS_L2_DATA_LEN_ERROR , **TS_L3_OK** ,
  TS_L3_FAIL , **TS_L3_UNAUTHORIZED** , **TS_L3_INVALID_CMD** , **TS_HOST_NO_SESSION** }

  Enum return type.

Functions

- const char * ts_ret_verbose (ts_ret_t ret)

5.3.1 Detailed Description

Author

Tropic Square s.r.o.

5.3.2 Data Structure Documentation

5.3.2.1 struct l3_frame_t L3 command and result packet

Data Fields

uint16_t	packet_size	
uint8_t	data[L3_FRAME_MAX_SIZE - L3_PACKET_SIZE_SIZE]	RES_SIZE or CMD_SIZE value

5.3.2.2 struct ts_handle_t

This structure holds data related to one physical chip. Contains AESGCM contexts for encrypting and decrypting L3 commands, nonce and device void pointer, which can be used for passing arbitrary data.

Data Fields

void *	device	
uint32_t	session	
uint8_t	IV[12]	
ts_aes_gcm_ctx_t	encrypt	
ts_aes_gcm_ctx_t	decrypt	
uint8_t	l2_buff[1+L2_CHUNK_MAX_FRAME_SIZE]	
uint8_t	l3_buff[L3_FRAME_MAX_SIZE]	

5.3.3 Macro Definition Documentation

5.3.3.1 L2_CHUNK_MAX_DATA_SIZE

```
#define L2_CHUNK_MAX_DATA_SIZE 252u
```

Maximal size of data field in one L2 transfer

5.3.3.2 TS_L1_LEN_MIN

```
#define TS_L1_LEN_MIN 1
```

Maximal number of data bytes in one L1 transfer

5.3.3.3 TS_L1_LEN_MAX

```
#define TS_L1_LEN_MAX (1 + 1 + 1 + L2_CHUNK_MAX_DATA_SIZE + 2)
```

Maximal number of data bytes in one L1 transfer

5.3.3.4 L3_PACKET_MAX_SIZE

```
#define L3_PACKET_MAX_SIZE (L3_CMD_ID_SIZE + L3_CMD_DATA_SIZE_MAX)
```

Maximum size of l3 ciphertext (or decrypted l3 packet)

5.3.3.5 L3_FRAME_MAX_SIZE

```
#define L3_FRAME_MAX_SIZE (L3_PACKET_SIZE_SIZE + L3_PACKET_MAX_SIZE + L3_↩
TAG_SIZE)
```

Max size of one unit of transport on l3 layer

5.3.4 Typedef Documentation

5.3.4.1 ts_handle_t `typedef struct ts_handle_t ts_handle_t`

This structure holds data related to one physical chip. Contains AESGCM contexts for encrypting and decrypting L3 commands, nonce and device void pointer, which can be used for passing arbitrary data.

5.3.5 Enumeration Type Documentation

5.3.5.1 ts_ret_t `enum ts_ret_t`

Enumerator

TS_OK	Operation was successful
TS_FAIL	Operation was not succesfull
TS_PARAM_ERR	Some parameter was not accepted by function
TS_L1_SPI_ERROR	Spi transfer returned error
TS_L1_DATA_LEN_ERROR	Data does not have an expected length
TS_L1_CHIP_STARTUP_MODE	Chip is in STARTUP mode
TS_L1_CHIP_ALARM_MODE	Chip is in ALARM mode
TS_L1_CHIP_BUSY	Chip is BUSY - typically chip is still booting
TS_L2_IN_CRC_ERR	Return values based on STATUS field Indicates that there are more l2 frames to be received
TS_L2_DATA_LEN_ERROR	L2 data does not have an expected length

5.3.6 Function Documentation

5.3.6.1 ts_ret_verbose() `const char* ts_ret_verbose ( ts_ret_t ret )`

Helper function for printing out error's associated name

Parameters

error	ts_ret_t error type

Returns

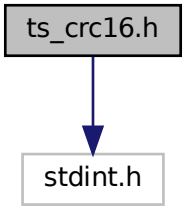
const char* name of error.

5.4 ts_crc16.h File Reference

CRC16 functions are defined here.

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

Include dependency graph for ts_crc16.h:



Functions

- uint16_t crc16 (const uint8_t *buf, int16_t size)
- void add_crc (void *req)

5.4.1 Detailed Description

Author

Tropic Square s.r.o.

5.4.2 Function Documentation

5.4.2.1 crc16()

```
uint16_t crc16 (
        const uint8_t * buf,
        int16_t size )
```

Parameters

buf	
size	

Returns

uint16_t

5.4.2.2 add_crc() `void add_crc ( void * req )`

Parameters

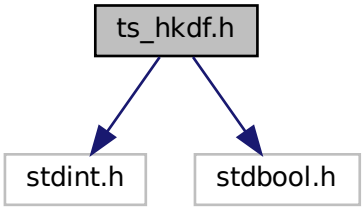
req	

5.5 ts_hkdf.h File Reference

HKDF function declaration.

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

Include dependency graph for ts_hkdf.h:



Functions

- void ts_hkdf (uint8_t *ck, uint32_t ck_size, uint8_t *input, uint32_t input_size, uint8_t nouts, uint8_t *output_1, uint8_t *output_2)

5.5.1 Detailed Description

Author

Tropic Square s.r.o.

5.5.2 Function Documentation

5.5.2.1 ts_hkdf() `void ts_hkdf ( uint8_t * ck, uint32_t ck_size, uint8_t * input, uint32_t input_size, uint8_t nouts, uint8_t * output_1, uint8_t * output_2 )`

The HMAC key derivation function as described in datasheet

Parameters

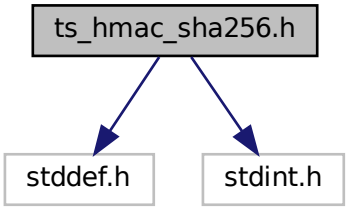
ck	CK parameter
ck_len	Length of CK parameter
input	Input data
input_size	Size of input data
nouts	Number of outputs
output_1	Output data 1
output_2	Output data 2

5.6 ts_hmac_sha256.h File Reference

HMAC SHA256 function declarations.

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

Include dependency graph for ts_hmac_sha256.h:



Functions

- void ts_hmac_sha256 (const uint8_t *key, size_t keylen, const uint8_t *input, size_t ilen, uint8_t *output)

5.6.1 Detailed Description

Author

Tropic Square s.r.o.

5.6.2 Function Documentation

5.6.2.1 ts_hmac_sha256()

```
void ts_hmac_sha256 (
        const uint8_t * key,
        size_t keylen,
        const uint8_t * input,
        size_t ilen,
        uint8_t * output )
```

This function computes HMAC SHA256 algorithm

Parameters

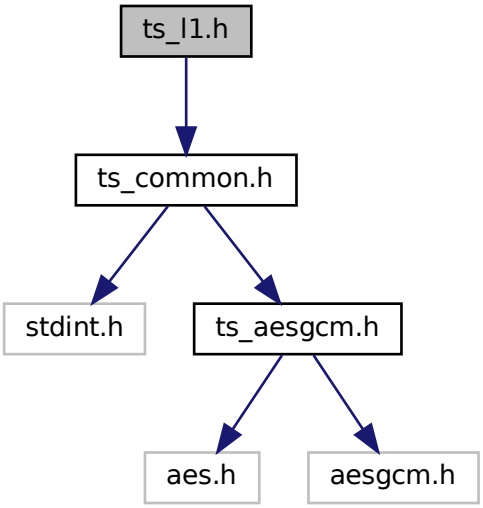
key	Key data buffer
keylen	Length of data in key data buffer
input	Input data buffer
ilen	Length of data in input data buffer
output	Output buffer

5.7 ts_l1.h File Reference

Layer 1 interfaces.

```
#include "ts_common.h"
```

Include dependency graph for ts_l1.h:



Macros

- #define **CHIP_MODE_READY_bit** 0x01
- #define **CHIP_MODE_ALARM_bit** 0x02
- #define **CHIP_MODE_STARTUP_bit** 0x04
- #define **TS_L1_READ_MAX_TRIES** 10
- #define **TS_L1_READ_RETRY_DELAY** 25
- #define **TS_L1_TIMEOUT_MS_MIN** 5
- #define **TS_L1_TIMEOUT_MS_DEFAULT** 70
- #define **TS_L1_TIMEOUT_MS_MAX** 150
- #define **TS_L1_DELAY_MS_MAX** 500
- #define **GET_INFO_REQ_ID** 0xAA

Functions

- ts_ret_t ts_l1_spi_csn_low (ts_handle_t *h)

  Set chip select pin low.
- ts_ret_t ts_l1_spi_csn_high (ts_handle_t *h)

  Set chip select pin high.
- ts_ret_t ts_l1_spi_transfer (ts_handle_t *h, uint8_t offset, uint16_t tx_len, uint32_t timeout)

  Do l1 transfer.
- ts_ret_t ts_l1_init (ts_handle_t *h)

  Platform agonostic init function, configurable during build. Check libtropic's documentation for more info about platform configuration.
- ts_ret_t ts_l1_deinit (ts_handle_t *h)

  Platform agonostic deinit function, configurable during build. Check libtropic's documentation for more info about platform configuration.
- ts_ret_t ts_l1_read (ts_handle_t *h, const uint32_t max_len, const uint32_t timeout)

  Read data from Tropic chip into host platform.
- ts_ret_t ts_l1_write (ts_handle_t *h, const uint16_t len, const uint32_t timeout)

  Write data from host platform into Tropic chip.
- ts_ret_t ts_l1_delay (ts_handle_t *h, uint32_t ms)

  Platform's definition for delay, specifies what host platform should do when libtropic's functions need some delay.

5.7.1 Detailed Description

Author

Tropic Square s.r.o.

5.7.2 Function Documentation

5.7.2.1 ts_l1_spi_csn_low()

```
ts_ret_t ts_l1_spi_csn_low (
        ts_handle_t * h )
```

Parameters

h	Chip's handle

Returns

TS_OK if success, otherwise returns other error code.

5.7.2.2 ts_l1_spi_csn_high()

```
ts_ret_t ts_l1_spi_csn_high (
        ts_handle_t * h )
```

Parameters

h	Chip's handle

Returns

TS_OK if success, otherwise returns other error code.

5.7.2.3 ts_l1_spi_transfer()

```
ts_ret_t ts_l1_spi_transfer (
        ts_handle_t * h,
        uint8_t offset,
        uint16_t tx_len,
        uint32_t timeout )
```

Parameters

h	Chip's handle
tx_len	The length of data to be transferred
timeout	Timeout

Returns

TS_OK if success, otherwise returns other error code.

5.7.2.4 ts_l1_init()

```
ts_ret_t ts_l1_init (
        ts_handle_t * h )
```

Parameters

h	Chip's handle

Returns

TS_OK if success, otherwise returns other error code.

5.7.2.5 ts_l1_deinit()

```
ts_ret_t ts_l1_deinit (
        ts_handle_t * h )
```

Parameters

h	Chip's handle

Returns

TS_OK if success, otherwise returns other error code.

5.7.2.6 ts_l1_read()

```
ts_ret_t ts_l1_read (
        ts_handle_t * h,
        const uint32_t max_len,
        const uint32_t timeout )
```

Parameters

h	Chip's handle
max_len	Max len of receive buffer
timeout	Timeout - how long function will wait for response

Returns

TS_OK if success, otherwise returns other error code.

5.7.2.7 ts_l1_write()

```
ts_ret_t ts_l1_write (
        ts_handle_t * h,
        const uint16_t len,
        const uint32_t timeout )
```

Parameters

h	Chip's handle
len	Length of data to send
timeout	Timeout

Returns

TS_OK if success, otherwise returns other error code.

5.7.2.8 ts_l1_delay()

```
ts_ret_t ts_l1_delay (
        ts_handle_t * h,
        uint32_t ms )
```

Parameters

h	Chip's handle
ms	Time to wait in miliseconds

Returns

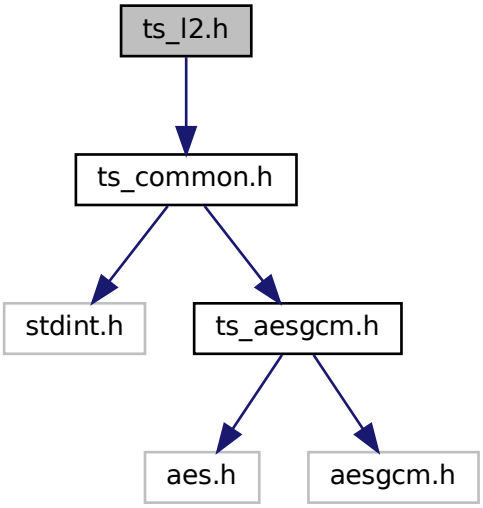
TS_OK if success, otherwise returns other error code.

5.8 ts_l2.h File Reference

Layer 2 interfaces.

```
#include "ts_common.h"
```

Include dependency graph for ts_l2.h:



Macros

- #define L2_STATUS_REQUEST_OK 0x01
- #define L2_STATUS_RESULT_OK 0x02
- #define L2_STATUS_REQUEST_CONT 0x03
- #define L2_STATUS_RESULT_CONT 0x04
- #define L2_STATUS_HSK_ERR 0x79
- #define L2_STATUS_NO_SESSION 0x7A
- #define L2_STATUS_TAG_ERR 0x7B
- #define L2_STATUS_CRC_ERR 0x7C
- #define L2_STATUS_UNKNOWN_ERR 0x7E
- #define L2_STATUS_GEN_ERR 0x7F
- #define L2_STATUS_NO_RESP 0xFF

Functions

- ts_ret_t ts_l2_frame_check (const uint8_t *frame)

  This function checks if incomming L2 frame is valid.
- ts_ret_t ts_l2_transfer (ts_handle_t *h)
- ts_ret_t ts_l2_encrypted_cmd (ts_handle_t *h)

  This function executes generic L3 command. It expects command's data correctly encrypted using keys created during previsously called ts_l2_handshake_req()

5.8.1 Detailed Description

Author

Tropic Square s.r.o.

5.8.2 Macro Definition Documentation

5.8.2.1 L2_STATUS_REQUEST_OK `#define L2_STATUS_REQUEST_OK 0x01`

STATUS field value

5.8.2.2 L2_STATUS_RESULT_OK `#define L2_STATUS_RESULT_OK 0x02`

STATUS field value

5.8.2.3 L2_STATUS_REQUEST_CONT `#define L2_STATUS_REQUEST_CONT 0x03`

STATUS field value

5.8.2.4 L2_STATUS_RESULT_CONT `#define L2_STATUS_RESULT_CONT 0x04`

STATUS field value

5.8.2.5 L2_STATUS_HSK_ERR `#define L2_STATUS_HSK_ERR 0x79`

STATUS field value

5.8.2.6 L2_STATUS_NO_SESSION `#define L2_STATUS_NO_SESSION 0x7A`

STATUS field value

5.8.2.7 L2_STATUS_TAG_ERR `#define L2_STATUS_TAG_ERR 0x7B`

STATUS field value

5.8.2.8 L2_STATUS_CRC_ERR `#define L2_STATUS_CRC_ERR 0x7C`

STATUS field value

5.8.2.9 L2_STATUS_UNKNOWN_ERR `#define L2_STATUS_UNKNOWN_ERR 0x7E`

STATUS field value

5.8.2.10 L2_STATUS_GEN_ERR `#define L2_STATUS_GEN_ERR 0x7F`

STATUS field value

5.8.2.11 L2_STATUS_NO_RESP `#define L2_STATUS_NO_RESP 0xFF`

STATUS field value

5.8.3 Function Documentation

5.8.3.1 ts_l2_frame_check() `ts_ret_t ts_l2_frame_check ( const uint8_t * frame )`

Parameters

frame	

Returns

TS_OK if success, otherwise returns other error code.

5.8.3.2 ts_l2_transfer() `ts_ret_t ts_l2_transfer ( ts_handle_t * h )`

Parameters

h	Chip's handle

Returns

TS_OK if success, otherwise returns other error code.

5.8.3.3 ts_l2_encrypted_cmd() `ts_ret_t ts_l2_encrypted_cmd ( ts_handle_t * h )`

Parameters

h	Chip's handle

Returns

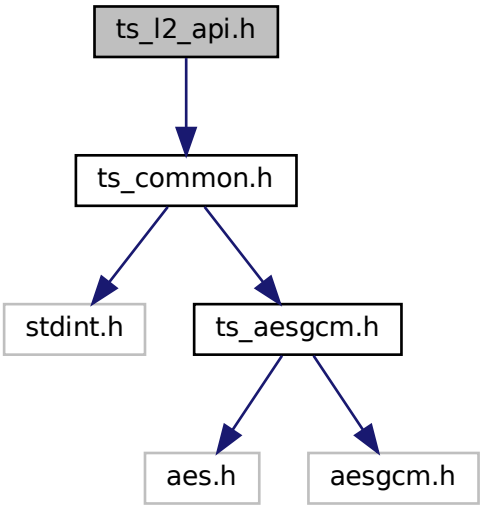
TS_OK if success, otherwise returns other error code.

5.9 ts_l2_api.h File Reference

Layer 2 API structures for various requests.

```
#include "ts_common.h"
```

Include dependency graph for ts_l2_api.h:



Data Structures

- struct l2_get_info_req_t
- struct l2_get_info_rsp_t
- struct l2_handshake_req_t
- struct l2_handshake_rsp_t
- struct l2_encrypted_cmd_req_t
- struct l2_encrypted_cmd_rsp_t

Macros

- #define TS_L2_GET_INFO_REQ_ID 0x01

  Command ID.
- #define TS_L2_GET_INFO_REQ_LEN 0x02

  Length of this request.
- #define TS_L2_HANDSHAKE_REQ_ID 0x02

  Command ID.
- #define TS_L2_HANDSHAKE_REQ_LEN 0x21

  Length of this request.
- #define TS_L2_ENCRYPTED_CMD_REQ_ID 0x04

  Command ID.

5.9.1 Detailed Description

Author

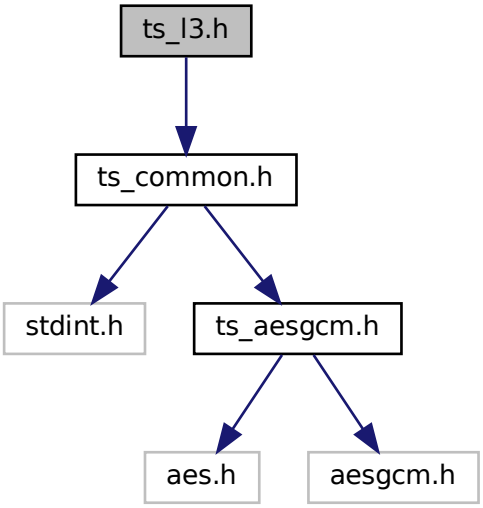
Tropic Square s.r.o.

5.10 ts_l3.h File Reference

This file contains interfaces related to layer 3.

```
#include "ts_common.h"
```

Include dependency graph for ts_l3.h:



Macros

- #define L3_RESULT_OK 0xC3u
- #define L3_RESULT_FAIL 0x3Cu
- #define L3_RESULT_UNAUTHORIZED 0x01u
- #define L3_RESULT_INVALID_CMD 0x02u

Functions

- ts_ret_t ts_l3_nonce_init (ts_handle_t *h)
- ts_ret_t ts_l3_nonce_increase (ts_handle_t *h)
- ts_ret_t ts_l3_cmd (ts_handle_t *h)

5.10.1 Detailed Description

Author

Tropic Square s.r.o.

5.10.2 Macro Definition Documentation

5.10.2.1 L3_RESULT_OK `#define L3_RESULT_OK 0xC3u`

L3 RESULT field Value

5.10.2.2 L3_RESULT_FAIL `#define L3_RESULT_FAIL 0x3Cu`

L3 RESULT field Value

5.10.2.3 L3_RESULT_UNAUTHORIZED `#define L3_RESULT_UNAUTHORIZED 0x01u`

L3 RESULT field Value

5.10.2.4 L3_RESULT_INVALID_CMD `#define L3_RESULT_INVALID_CMD 0x02u`

L3 RESULT field Value

5.10.3 Function Documentation

5.10.3.1 ts_l3_nonce_init() `ts_ret_t ts_l3_nonce_init ( ts_handle_t * h )`

Initializes nonce in handle to 0. This function is used during secure handshake.

Parameters

h	Chip's handle

Returns

TS_OK if success, otherwise returns other error code.

5.10.3.2 ts_l3_nonce_increase() `ts_ret_t ts_l3_nonce_increase ( ts_handle_t * h )`

Increases by one nonce stored in handle. This function is used after successfull reception of L3 response. , uint16_t cmd_len,

Parameters

h	Chip's handle

Returns

TS_OK if success, otherwise returns other error code.

5.10.3.3 ts_l3_cmd() `ts_ret_t ts_l3_cmd ( ts_handle_t * h )`

Perform l3 encrypted command operation.

Parameters

h	Chip's handle

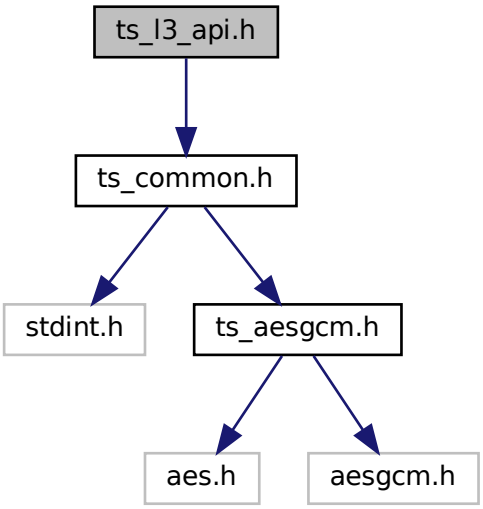
Returns

TS_OK if success, otherwise returns other error code.

5.11 ts_l3_api.h File Reference

Layer 3 API structures for various requests.

`#include "ts_common.h"`
Include dependency graph for ts_l3_api.h:



Data Structures

- struct ts_l3_ping_cmd_t
- struct ts_l3_ping_res_t
- struct ts_l3_random_value_get_cmd_t
- struct ts_l3_random_value_get_res_t
- struct ts_l3_ecc_key_generate_cmd_t
- struct ts_l3_ecc_key_generate_res_t
- struct ts_l3_ecc_key_read_cmd_t
- struct ts_l3_ecc_key_read_res_t
- struct ts_l3_ecc_key_erase_cmd_t
- struct ts_l3_ecc_key_erase_res_t
- struct ts_l3_eddsa_sign_cmd_t
- struct ts_l3_eddsa_sign_res_t
- struct ts_l3_ecdsa_sign_cmd_t
- struct ts_l3_ecdsa_sign_res_t

Macros

- #define TS_L3_PING_CMD 0x01

  Command ID.
- #define TS_L3_RANDOM_VALUE_GET_CMD 0x50

  Command ID.
- #define TS_L3_RANDOM_VALUE_GET_CMD_SIZE 2

  Command length.
- #define TS_L3_ECC_KEY_GENERATE_CMD 0x60

  ECC_Key_generate command ID.
- #define TS_L3_ECC_KEY_GENERATE_CMD_SIZE 0x04u

  ECC_Key_generate command size.
- #define TS_L3_ECC_KEY_GENERATE_SLOT_MIN 0

  ECC_Key_generate min slot number.
- #define TS_L3_ECC_KEY_GENERATE_SLOT_MAX 31

  ECC_Key_generate max slot number.
- #define TS_L3_ECC_KEY_READ_CMD 0x62

  Command ID.
- #define TS_L3_ECC_KEY_READ_CMD_SIZE 0x03

  Command length.
- #define TS_L3_ECC_KEY_READ_CURVE_P256 1
- #define **TS_L3_ECC_KEY_READ_CURVE_ED25519** 2
- #define TS_L3_ECC_KEY_READ_ORIGIN_ECC_KEY_GENERATE 1
- #define **TS_L3_ECC_KEY_READ_ORIGIN_ECC_KEY_STORE** 2
- #define TS_L3_ECC_KEY_ERASE_CMD 0x63u

  Command ID.
- #define TS_L3_ECC_KEY_ERASE_CMD_SIZE 0x03u

  Command length.
- #define TS_L3_EDDSA_SIGN_CMD 0x71

  Command ID.
- #define TS_L3_EDDSA_SIGN_CMD_SIZE 0x10u

  Command length.
- #define **TS_L3_EDDSA_SIGN_MSG_LEN_MIN** 0x01
- #define **TS_L3_EDDSA_SIGN_MSG_LEN_MAX** 4096
- #define TS_L3_ECDSA_SIGN 0x70

  Command ID.
- #define TS_L3_ECDSA_SIGN_CMD_SIZE 0x30u

  Command length.
- #define **TS_L3_ECDSA_SIGN_MSG_HASH_LEN** 0x20

5.11.1 Detailed Description

Author

Tropic Square s.r.o.

5.11.2 Macro Definition Documentation

5.11.2.1 TS_L3_ECC_KEY_READ_CURVE_P256 `#define TS_L3_ECC_KEY_READ_CURVE_P256 1`

Elliptic Curve

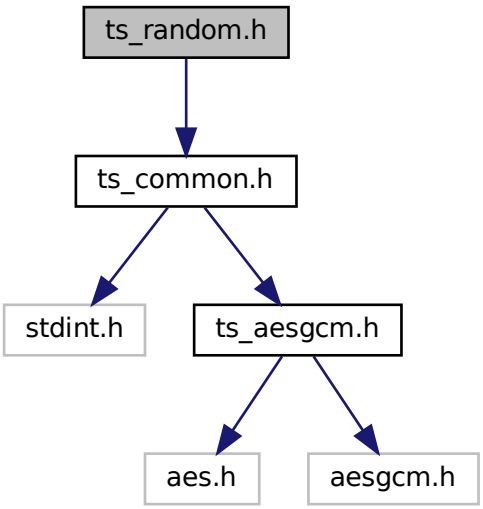
5.11.2.2 TS_L3_ECC_KEY_READ_ORIGIN_ECC_KEY_GENERATE `#define TS_L3_ECC_KEY_READ_ORIGIN_ECC_KEY_GENERATE 1`

The origin of the key

5.12 ts_random.h File Reference

API for providing random numbers from host platform RNG.

```
#include "ts_common.h"
```
Include dependency graph for ts_random.h:



Functions

- ts_ret_t ts_random_bytes (uint8_t *buff, uint16_t len)
  Get a number of random bytes from the host platform RNG.

5.12.1 Detailed Description

Author

Tropic Square s.r.o.

5.12.2 Function Documentation

5.12.2.1 ts_random_bytes()

```
ts_ret_t ts_random_bytes (
        uint8_t * buff,
        uint16_t len )
```

Parameters

buff	Buffer to be filled with random bytes
len	Number of random bytes

Returns

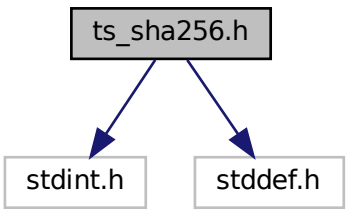
ts_ret_t TS_OK if all went OK, or other return value.

5.13 ts_sha256.h File Reference

SHA256 function declarations.

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

Include dependency graph for ts_sha256.h:



Data Structures

- struct ts_sha256_ctx

Macros

- #define **SHA256_DIGEST_LENGTH** 32

Typedefs

- typedef struct ts_sha256_ctx **ts_sha256_ctx_t**

Functions

- void ts_sha256_init (void *ctx)
- void ts_sha256_start (void *ctx)
- void ts_sha256_update (void *ctx, const uint8_t *input, size_t len)
- void ts_sha256_finish (void *ctx, uint8_t *output)

  This function finalizes hashing and outputs a digest.

5.13.1 Detailed Description

Author

Tropic Square s.r.o.

5.13.2 Data Structure Documentation

5.13.2.1 struct ts_sha256_ctx

5.13.3 Function Documentation

5.13.3.1 ts_sha256_init() `void ts_sha256_init ( void * ctx )`

This function initializes hash context

Parameters

ctx	Hash context

5.13.3.2 ts_sha256_start() `void ts_sha256_start (`

```
    void * ctx )
```

This function starts hash context

Parameters

ctx	

5.13.3.3 ts_sha256_update() `void ts_sha256_update (`

```
    void * ctx,
    const uint8_t * input,
    size_t len )
```

This function add data to hashing context

Parameters

ctx	Hash context
input	Input data
len	Length of input data

5.13.3.4 ts_sha256_finish() `void ts_sha256_finish (`

```
    void * ctx,
    uint8_t * output )
```

Parameters

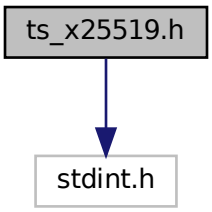
ctx	Hash context
output	Hash digest

5.14 ts_x25519.h File Reference

X25519 function declarations.

```
#include "stdint.h"
```

Include dependency graph for ts_x25519.h:



Functions

- void ts_X25519 (const uint8_t *priv, const uint8_t *pub, uint8_t *secret)
- void ts_X25519_scalarmult (const uint8_t *sk, uint8_t *pk)

  X25519 scalar multiplication with a base point.

5.14.1 Detailed Description

Author

Tropic Square s.r.o.

5.14.2 Function Documentation

5.14.2.1 ts_X25519()

```
void ts_X25519 (
    const uint8_t * priv,
    const uint8_t * pub,
    uint8_t * secret )
```

This function computes x25519 shared secret

Parameters

priv	Private key 32B long
pub	Public key 32B long
secret	Shared secret 32B long

5.14.2.2 ts_X25519_scalarmult()

```
void ts_X25519_scalarmult (
```

```
    const uint8_t * sk,
    uint8_t * pk )
```

Parameters

sk	Secret key
pk	Public key

Index