
libtropic

C library for TROPIC devices

Version: 0.0.1

Git tag: v0.0.1

Tropic Square

July 25, 2024



tropicsquare



1 libtropic	1
1.1 Introduction	1
1.2 Dependencies	1
1.3 Examples	1
1.4 Running tests	1
1.5 Library configuration	1
1.5.1 Cryptography support	1
1.6 Library overview	2
2 Module Index	3
2.1 Modules	3
3 File Index	3
3.1 File List	3
4 Module Documentation	4
4.1 [PUBLIC API]	4
4.1.1 Detailed Description	4
4.2 ts_init	4
4.2.1 Detailed Description	5
4.2.2 Function Documentation	5
4.3 ts_deinit	5
4.3.1 Detailed Description	5
4.3.2 Function Documentation	5
4.4 ts_handshake	6
4.4.1 Detailed Description	6
4.4.2 Function Documentation	6
4.5 ts_ping	7
4.5.1 Detailed Description	7
4.5.2 Function Documentation	7
4.6 ts_random_get	7
4.6.1 Detailed Description	8
4.6.2 Function Documentation	8
4.7 ECC functions	8
4.7.1 Detailed Description	10
4.8 ts_ecc_key_generate	10
4.8.1 Detailed Description	10
4.8.2 Function Documentation	10
4.9 ts_ecc_key_read	11
4.9.1 Detailed Description	11
4.9.2 Function Documentation	11
4.10 ts_eddsa_sign	12
4.10.1 Detailed Description	12
4.10.2 Function Documentation	12
4.11 ts_ecdsa_sign	12



4.11.1 Detailed Description	13
4.11.2 Function Documentation	13
4.12 ts_ecc_key_erase	13
4.12.1 Detailed Description	13
4.12.2 Function Documentation	14
4.13 ts_get_info_cert	14
4.13.1 Detailed Description	15
4.13.2 Function Documentation	15
4.14 [L2 functions]	15
4.14.1 Detailed Description	16
4.14.2 Macro Definition Documentation	16
4.14.3 Function Documentation	17
4.15 [PRIVATE API]	18
4.15.1 Detailed Description	18
4.16 [L2 API]	18
4.16.1 Detailed Description	19
4.17 get_info_req	19
4.17.1 Detailed Description	19
4.17.2 Data Structure Documentation	19
4.18 handshake_req	20
4.18.1 Detailed Description	20
4.18.2 Data Structure Documentation	20
4.19 encrypted_cmd_req	21
4.19.1 Detailed Description	21
4.19.2 Data Structure Documentation	21
4.20 [L3 functions]	22
4.20.1 Detailed Description	22
4.20.2 Macro Definition Documentation	22
4.20.3 Function Documentation	23
4.21 [L3 API]	24
4.21.1 Detailed Description	24
4.22 Ping	24
4.22.1 Detailed Description	24
4.22.2 Data Structure Documentation	24
4.23 Random_Value_Get	25
4.23.1 Detailed Description	25
4.23.2 Data Structure Documentation	25
4.24 ECC_Key_Generate	26
4.24.1 Detailed Description	26
4.24.2 Data Structure Documentation	26
4.25 ECC_Key_Read	27
4.25.1 Detailed Description	27



4.25.2 Data Structure Documentation	27
4.26 ECC_Key_Erase	28
4.26.1 Detailed Description	28
4.26.2 Data Structure Documentation	28
4.27 EDDSA_Sign	29
4.27.1 Detailed Description	29
4.27.2 Data Structure Documentation	29
4.28 ECDSA_Sign	30
4.28.1 Detailed Description	30
4.28.2 Data Structure Documentation	30
5 File Documentation	31
5.1 libtropic.h File Reference	31
5.1.1 Detailed Description	34
5.1.2 Function Documentation	34
5.2 ts_aesgcm.h File Reference	39
5.2.1 Detailed Description	40
5.2.2 Data Structure Documentation	40
5.2.3 Function Documentation	40
5.3 ts_common.h File Reference	42
5.3.1 Detailed Description	43
5.3.2 Data Structure Documentation	43
5.3.3 Macro Definition Documentation	44
5.3.4 Typedef Documentation	45
5.3.5 Enumeration Type Documentation	45
5.3.6 Function Documentation	45
5.4 ts_crc16.h File Reference	46
5.4.1 Detailed Description	46
5.4.2 Function Documentation	46
5.5 ts_hkdf.h File Reference	47
5.5.1 Detailed Description	47
5.5.2 Function Documentation	47
5.6 ts_hmac_sha256.h File Reference	48
5.6.1 Detailed Description	48
5.6.2 Function Documentation	48
5.7 ts_l1.h File Reference	49
5.7.1 Detailed Description	50
5.7.2 Function Documentation	50
5.8 ts_l2.h File Reference	53
5.8.1 Detailed Description	54
5.8.2 Macro Definition Documentation	54
5.8.3 Function Documentation	55
5.9 ts_l2_api.h File Reference	56



5.9.1 Detailed Description	57
5.10 ts_l3.h File Reference	57
5.10.1 Detailed Description	57
5.10.2 Macro Definition Documentation	58
5.10.3 Function Documentation	58
5.11 ts_l3_api.h File Reference	59
5.11.1 Detailed Description	61
5.11.2 Macro Definition Documentation	61
5.12 ts_random.h File Reference	61
5.12.1 Detailed Description	62
5.12.2 Function Documentation	62
5.13 ts_sha256.h File Reference	62
5.13.1 Detailed Description	63
5.13.2 Data Structure Documentation	63
5.13.3 Function Documentation	63
5.14 ts_x25519.h File Reference	64
5.14.1 Detailed Description	65
5.14.2 Function Documentation	65
Index	67



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 cryptography is `vendor/trezor_crypto`.

```
# Use trezor_crypto library:
option(TS_CRYPTOTREZOR "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 I2 request/response functions.

Related code:

- [ts_l2.c](#)
- [ts_l2.h](#)

Tropic layer 3

This layer is preparing and parsing I3 commands/results, it uses I2 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:

[PUBLIC API]	4
ts_init	4
ts_deinit	5
ts_handshake	6
ts_ping	7
ts_random_get	7
ECC functions	8
ts_ecc_key_generate	10
ts_ecc_key_read	11
ts_eddsa_sign	12
ts_ecdsa_sign	12
ts_ecc_key_erase	13
ts_get_info_cert	14
[PRIVATE API]	18
[L2 functions]	15
[L2 API]	18
get_info_req	19
handshake_req	20
encrypted_cmd_req	21
[L3 functions]	22
[L3 API]	24
Ping	24
Random_Value_Get	25
ECC_Key_Generate	26
ECC_Key_Read	27
ECC_Key_Erase	28
EDDSA_Sign	29
ECDSA_Sign	30

3 File Index

3.1 File List

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

libtropic.h	31
Libtropic header file	
ts_aesgcm.h	39
AESGCM function declarations	
ts_common.h	42
Shared definitions and functions commonly used by more layers	
ts_crc16.h	46
CRC16 functions are defined here	
ts_hkdf.h	47
HKDF function declaration	
ts_hmac_sha256.h	48
HMAC SHA256 function declarations	



ts_l1.h	Layer 1 interfaces	49
ts_l2.h	Layer 2 interfaces	53
ts_l2_api.h	Layer 2 API structures for various requests	56
ts_l3.h	This file contains interfaces related to layer 3	57
ts_l3_api.h	Layer 3 API structures for various requests	59
ts_random.h	API for providing random numbers from host platform RNG	61
ts_sha256.h	SHA256 function declarations	62
ts_x25519.h	X25519 function declarations	64

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

<code>h</code>	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

<code>h</code>	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

<i>h</i>	Device's handle
<i>buff</i>	Buffer
<i>len</i>	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

<i>h</i>	Device's handle
<i>slot</i>	Slot number ECC_SLOT_1 - ECC_SLOT_32
<i>curve</i>	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

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

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

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

<i>h</i>	Device's handle
<i>slot</i>	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 RISC-V 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 256-383 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

<i>h</i>	Device's handle
<i>cert</i>	Certificate's buffer
<i>max_len</i>	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

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

Returns

TS_OK if success, otherwise returns other error code.

4.14 [L2 functions]

Functions controlling I2 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 incoming 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 previously 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

<code>frame</code>	
--------------------	--

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

<code>h</code>	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

<code>h</code>	Chip's handle
----------------	---------------

Returns

TS_OK if success, otherwise returns other error code.

4.15 [PRIVATE API]

Modules

- [\[L2 functions\]](#)
Functions controlling I2 transmission.
- [\[L2 API\]](#)
Data Link Layer.
- [\[L3 functions\]](#)
Functions controlling I3 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

<code>h</code>	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 successful reception of L3 response. , uint16_t cmd_len,

Parameters

<code>h</code>	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

<code>h</code>	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

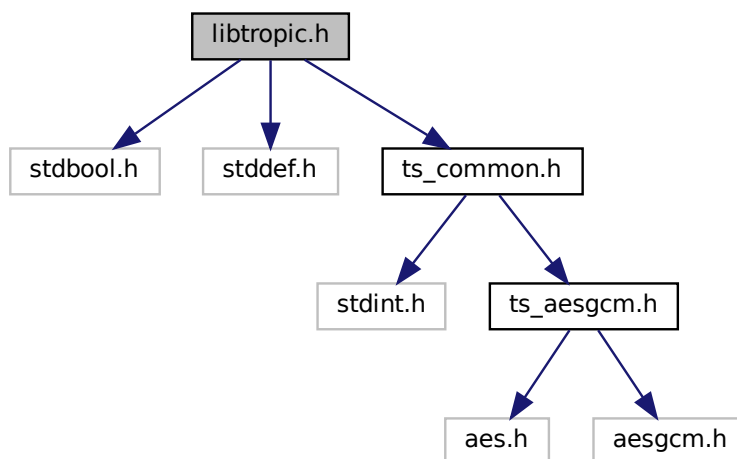
4.28.2.2 struct ts_l3_ecdsa_sign_res_t

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 256-383 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

<i>h</i>	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

<i>h</i>	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

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

<i>h</i>	Device's handle
<i>msg_out</i>	Ping message going out
<i>msg_in</i>	Ping message going in
<i>len</i>	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

<i>h</i>	Device's handle
<i>buff</i>	Buffer
<i>len</i>	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

<i>h</i>	Device's handle
<i>slot</i>	Slot number ECC_SLOT_1 - ECC_SLOT_32
<i>curve</i>	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

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

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

<i>h</i>	Device's handle
<i>slot</i>	Slot containing a private key, ECC_SLOT_1 - ECC_SLOT_32



Parameters

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

<i>h</i>	Device's handle
<i>slot</i>	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

<i>h</i>	Device's handle
<i>cert</i>	Certificate's buffer
<i>max_len</i>	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

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

Returns

TS_OK if success, otherwise returns other error code.

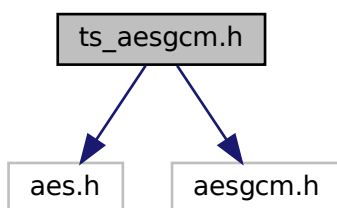
5.2 ts_aesgcm.h File Reference

AESGCM function declarations.

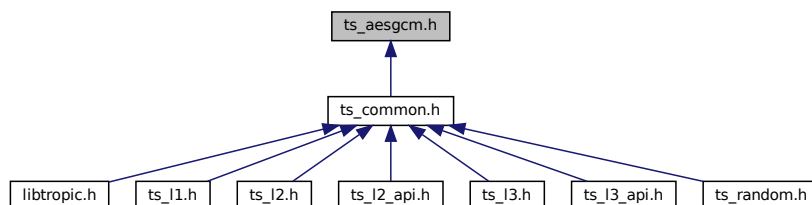
```
#include "aes.h"
```

```
#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

<i>ctx</i>	AESGCM context structure
<i>key</i>	The key value
<i>key_len</i>	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

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

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

<code>ctx</code>	AESGCM context structure
------------------	--------------------------

Returns

TS_OK if success, otherwise returns other error code.

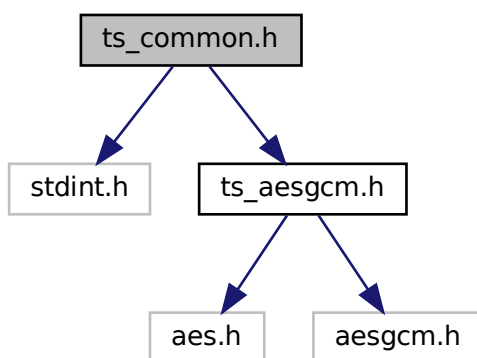
5.3 ts_common.h File Reference

Shared definitions and functions commonly used by more layers.

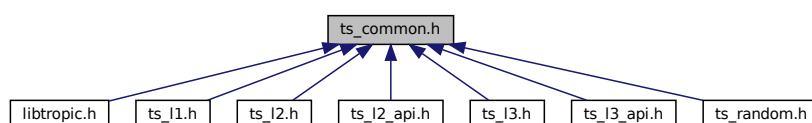
```
#include "stdint.h"
```

```
#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	I2_buff[1+L2_CHUNK_MAX_FRAME_SIZE]	
uint8_t	I3_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 I3 ciphertext (or decrypted I3 packet)

5.3.3.5 L3_FRAME_MAX_SIZE #define L3_FRAME_MAX_SIZE (L3_PACKET_SIZE_SIZE + L3_PACKET_MAX_SIZE + L3_CMD_TAG_SIZE)

Max size of one unit of transport on I3 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

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

<i>error</i>	<code>ts_ret_t</code> error type
--------------	----------------------------------

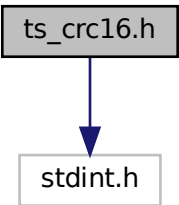
Returns

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

<i>buf</i>	
<i>size</i>	

Returns

uint16_t



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

Parameters

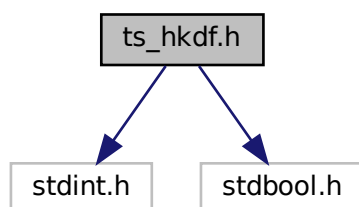
<i>req</i>	
------------	--

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

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

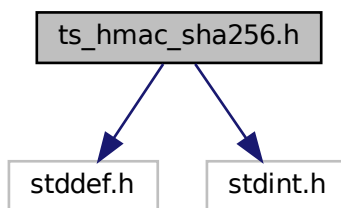
5.6 ts_hmac_sha256.h File Reference

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

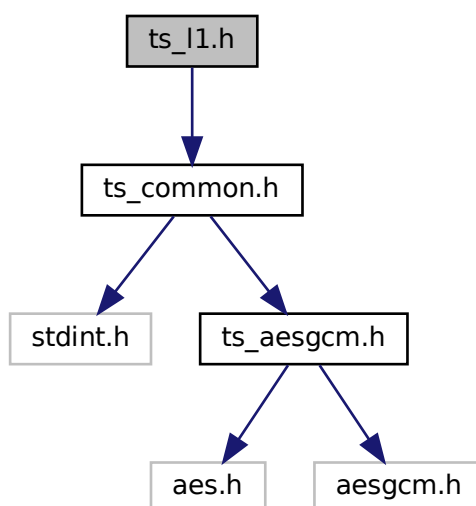
<i>key</i>	Key data buffer
<i>keylen</i>	Length of data in key data buffer
<i>input</i>	Input data buffer
<i>ilen</i>	Length of data in input data buffer
<i>output</i>	Output buffer

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 I1 transfer.
- `ts_ret_t ts_l1_init (ts_handle_t *h)`
Platform agnostic 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 agnostic 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

<code>h</code>	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

<i>h</i>	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

<i>h</i>	Chip's handle
<i>tx_len</i>	The length of data to be transferred
<i>timeout</i>	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

<i>h</i>	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

<i>h</i>	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

<i>h</i>	Chip's handle
<i>max_len</i>	Max len of receive buffer
<i>timeout</i>	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

<i>h</i>	Chip's handle
<i>len</i>	Length of data to send
<i>timeout</i>	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

<i>h</i>	Chip's handle
<i>ms</i>	Time to wait in milliseconds



Returns

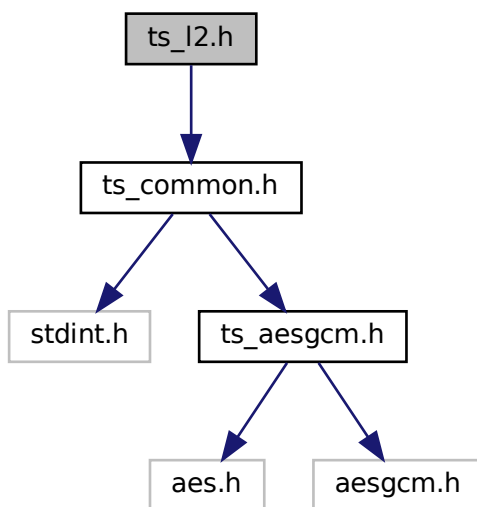
TS_OK if success, otherwise returns other error code.

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 incoming 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 previously 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

<i>frame</i>	
--------------	--

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

<i>h</i>	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

<i>h</i>	Chip's handle
----------	---------------



Returns

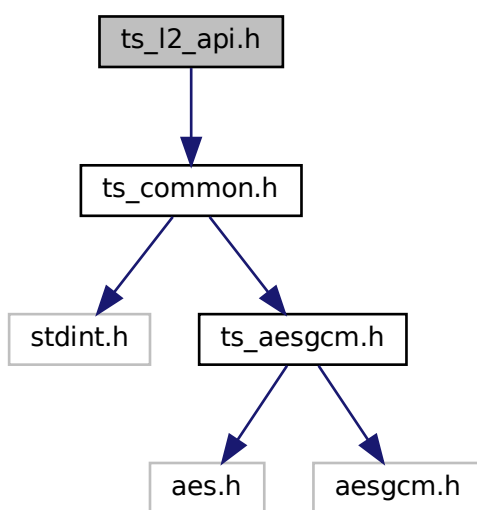
TS_OK if success, otherwise returns other error code.

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

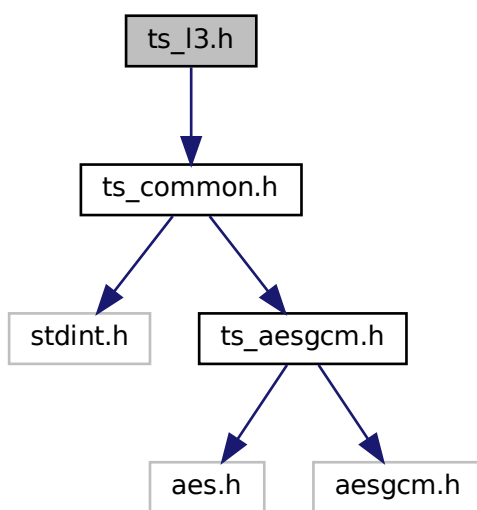
Tropic Square s.r.o.

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

<code>h</code>	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 successful reception of L3 response. , uint16_t cmd_len,



Parameters

<i>h</i>	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

<i>h</i>	Chip's handle
----------	---------------

Returns

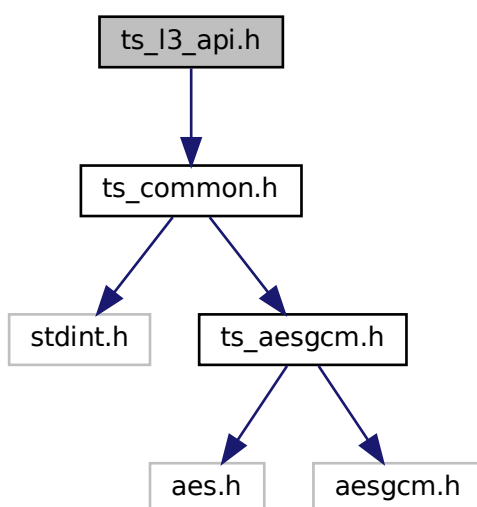
TS_OK if success, otherwise returns other error code.

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`

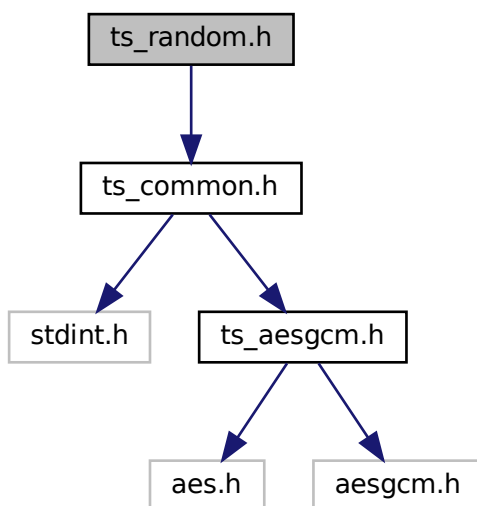
The origin of the key

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

<i>buff</i>	Buffer to be filled with random bytes
<i>len</i>	Number of random bytes

Returns

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

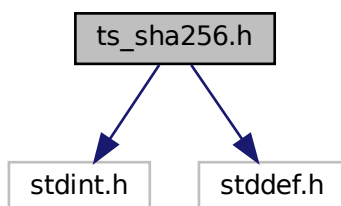
5.13 ts_sha256.h File Reference

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

<code>ctx</code>	Hash context
------------------	--------------



5.13.3.2 ts_sha256_start() `void ts_sha256_start (`
`void * ctx)`

This function starts hash context

Parameters

<i>ctx</i>	
------------	--

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

<i>ctx</i>	Hash context
<i>input</i>	Input data
<i>len</i>	Length of input data

5.13.3.4 ts_sha256_finish() `void ts_sha256_finish (`
`void * ctx,`
`uint8_t * output)`

Parameters

<i>ctx</i>	Hash context
<i>output</i>	Hash digest

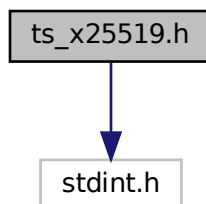
5.14 ts_x25519.h File Reference

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

<i>priv</i>	Private key 32B long
<i>pub</i>	Public key 32B long
<i>secret</i>	Shared secret 32B long

5.14.2.2 ts_X25519_scalarmult() void ts_X25519_scalarmult (



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

Parameters

<i>sk</i>	Secret key
<i>pk</i>	Public key

Index

[L2 API], [18](#)
[L2 functions], [15](#)
 L2_STATUS_CRC_ERR, [17](#)
 L2_STATUS_GEN_ERR, [17](#)
 L2_STATUS_HSK_ERR, [16](#)
 L2_STATUS_NO_RESP, [17](#)
 L2_STATUS_NO_SESSION, [17](#)
 L2_STATUS_REQUEST_CONT, [16](#)
 L2_STATUS_REQUEST_OK, [16](#)
 L2_STATUS_RESULT_CONT, [16](#)
 L2_STATUS_RESULT_OK, [16](#)
 L2_STATUS_TAG_ERR, [17](#)
 L2_STATUS_UNKNOWN_ERR, [17](#)
 ts_l2_encrypted_cmd, [18](#)
 ts_l2_frame_check, [17](#)
 ts_l2_transfer, [17](#)
[L3 API], [24](#)
[L3 functions], [22](#)
 L3_RESULT_FAIL, [22](#)
 L3_RESULT_INVALID_CMD, [22](#)
 L3_RESULT_OK, [22](#)
 L3_RESULT_UNAUTHORIZED, [22](#)
 ts_l3_cmd, [23](#)
 ts_l3_nonce_increase, [23](#)
 ts_l3_nonce_init, [23](#)
[PRIVATE API], [18](#)
[PUBLIC API], [4](#)

add_crc
 ts_crc16.h, [46](#)

crc16
 ts_crc16.h, [46](#)

ECC functions, [8](#)
ECC_Key_Erase, [28](#)
ECC_Key_Generate, [26](#)
ECC_Key_Read, [27](#)
ECDSA_Sign, [30](#)
EDDSA_Sign, [29](#)
encrypted_cmd_req, [21](#)

get_info_req, [19](#)

handshake_req, [20](#)

L2_CHUNK_MAX_DATA_SIZE
 ts_common.h, [44](#)
l2_encrypted_cmd_req_t, [21](#)
l2_encrypted_cmd_rsp_t, [21](#)
l2_get_info_req_t, [19](#)
l2_get_info_rsp_t, [19](#)
l2_handshake_req_t, [20](#)
l2_handshake_rsp_t, [20](#)
L2_STATUS_CRC_ERR
 [L2 functions], [17](#)
 ts_l2.h, [54](#)
L2_STATUS_GEN_ERR
 [L2 functions], [17](#)
 ts_l2.h, [54](#)
L2_STATUS_HSK_ERR
 [L2 functions], [16](#)
 ts_l2.h, [54](#)
L2_STATUS_NO_RESP
 [L2 functions], [17](#)
 ts_l2.h, [55](#)
L2_STATUS_NO_SESSION
 [L2 functions], [17](#)
 ts_l2.h, [54](#)
L2_STATUS_REQUEST_CONT
 [L2 functions], [16](#)
 ts_l2.h, [54](#)
L2_STATUS_REQUEST_OK
 [L2 functions], [16](#)
 ts_l2.h, [54](#)
L2_STATUS_RESULT_CONT
 [L2 functions], [16](#)
 ts_l2.h, [54](#)
L2_STATUS_RESULT_OK
 [L2 functions], [16](#)
 ts_l2.h, [54](#)
L2_STATUS_TAG_ERR
 [L2 functions], [17](#)
 ts_l2.h, [54](#)
L2_STATUS_UNKNOWN_ERR
 [L2 functions], [17](#)
 ts_l2.h, [54](#)
L3_FRAME_MAX_SIZE
 ts_common.h, [44](#)
l3_frame_t, [43](#)
L3_PACKET_MAX_SIZE
 ts_common.h, [44](#)
L3_RESULT_FAIL
 [L3 functions], [22](#)
 ts_l3.h, [58](#)
L3_RESULT_INVALID_CMD
 [L3 functions], [22](#)
 ts_l3.h, [58](#)
L3_RESULT_OK
 [L3 functions], [22](#)



- ts_l3.h, [58](#)
- L3_RESULT_UNAUTHORIZED
 - [L3 functions], [22](#)
 - ts_l3.h, [58](#)
- libtropic.h, [31](#)
 - ts_cert_verify_and_parse, [38](#)
 - ts_deinit, [35](#)
 - ts_ecc_key_erase, [38](#)
 - ts_ecc_key_generate, [36](#)
 - ts_ecc_key_read, [36](#)
 - ts_ecdsa_sign, [37](#)
 - ts_eddsa_sign, [37](#)
 - ts_get_info_cert, [38](#)
 - ts_handshake, [35](#)
 - ts_init, [34](#)
 - ts_ping, [35](#)
 - ts_random_get, [36](#)
- Ping, [24](#)
- Random_Value_Get, [25](#)
- ts_aes_gcm_ctx, [40](#)
- ts_aesgcm.h, [39](#)
 - ts_aesgcm_decrypt, [41](#)
 - ts_aesgcm_encrypt, [40](#)
 - ts_aesgcm_end, [42](#)
 - ts_aesgcm_init_and_key, [40](#)
- ts_aesgcm_decrypt
 - ts_aesgcm.h, [41](#)
- ts_aesgcm_encrypt
 - ts_aesgcm.h, [40](#)
- ts_aesgcm_end
 - ts_aesgcm.h, [42](#)
- ts_aesgcm_init_and_key
 - ts_aesgcm.h, [40](#)
- ts_cert_verify_and_parse
 - libtropic.h, [38](#)
 - ts_get_info_cert, [15](#)
- ts_common.h, [42](#)
 - L2_CHUNK_MAX_DATA_SIZE, [44](#)
 - L3_FRAME_MAX_SIZE, [44](#)
 - L3_PACKET_MAX_SIZE, [44](#)
 - TS_FAIL, [45](#)
 - ts_handle_t, [45](#)
 - TS_L1_CHIP_ALARM_MODE, [45](#)
 - TS_L1_CHIP_BUSY, [45](#)
 - TS_L1_CHIP_STARTUP_MODE, [45](#)
 - TS_L1_DATA_LEN_ERROR, [45](#)
 - TS_L1_LEN_MAX, [44](#)
 - TS_L1_LEN_MIN, [44](#)
 - TS_L1_SPI_ERROR, [45](#)
 - TS_L2_DATA_LEN_ERROR, [45](#)
 - TS_L2_IN_CRC_ERR, [45](#)
 - TS_OK, [45](#)
 - TS_PARAM_ERR, [45](#)
 - ts_ret_t, [45](#)
 - ts_ret_verbose, [45](#)
- ts_crc16.h, [46](#)
 - add_crc, [46](#)
 - crc16, [46](#)
- ts_deinit, [5](#)
 - libtropic.h, [35](#)
 - ts_deinit, [5](#)
- ts_ecc_key_erase, [13](#)
 - libtropic.h, [38](#)
 - ts_ecc_key_erase, [14](#)
- ts_ecc_key_generate, [10](#)
 - libtropic.h, [36](#)
 - ts_ecc_key_generate, [10](#)
- ts_ecc_key_read, [11](#)
 - libtropic.h, [36](#)
 - ts_ecc_key_read, [11](#)
- ts_ecdsa_sign, [12](#)
 - libtropic.h, [37](#)
 - ts_ecdsa_sign, [13](#)
- ts_eddsa_sign, [12](#)
 - libtropic.h, [37](#)
 - ts_eddsa_sign, [12](#)
- TS_FAIL
 - ts_common.h, [45](#)
- ts_get_info_cert, [14](#)
 - libtropic.h, [38](#)
 - ts_cert_verify_and_parse, [15](#)
 - ts_get_info_cert, [15](#)
- ts_handle_t, [44](#)
 - ts_common.h, [45](#)
- ts_handshake, [6](#)
 - libtropic.h, [35](#)
 - ts_handshake, [6](#)
- ts_hkdf
 - ts_hkdf.h, [47](#)
- ts_hkdf.h, [47](#)
 - ts_hkdf, [47](#)
- ts_hmac_sha256
 - ts_hmac_sha256.h, [48](#)
- ts_hmac_sha256.h, [48](#)
 - ts_hmac_sha256, [48](#)
- ts_init, [4](#)
 - libtropic.h, [34](#)
 - ts_init, [5](#)
- ts_l1.h, [49](#)
 - ts_l1_deinit, [51](#)
 - ts_l1_delay, [52](#)
 - ts_l1_init, [51](#)
 - ts_l1_read, [52](#)
 - ts_l1_spi_csn_high, [50](#)
 - ts_l1_spi_csn_low, [50](#)
 - ts_l1_spi_transfer, [51](#)
 - ts_l1_write, [52](#)
- TS_L1_CHIP_ALARM_MODE
 - ts_common.h, [45](#)
- TS_L1_CHIP_BUSY
 - ts_common.h, [45](#)
- TS_L1_CHIP_STARTUP_MODE
 - ts_common.h, [45](#)
- TS_L1_DATA_LEN_ERROR



- ts_common.h, [45](#)
- ts_l1_deinit
 - ts_l1.h, [51](#)
- ts_l1_delay
 - ts_l1.h, [52](#)
- ts_l1_init
 - ts_l1.h, [51](#)
- TS_L1_LEN_MAX
 - ts_common.h, [44](#)
- TS_L1_LEN_MIN
 - ts_common.h, [44](#)
- ts_l1_read
 - ts_l1.h, [52](#)
- ts_l1_spi_csn_high
 - ts_l1.h, [50](#)
- ts_l1_spi_csn_low
 - ts_l1.h, [50](#)
- TS_L1_SPI_ERROR
 - ts_common.h, [45](#)
- ts_l1_spi_transfer
 - ts_l1.h, [51](#)
- ts_l1_write
 - ts_l1.h, [52](#)
- ts_l2.h, [53](#)
 - L2_STATUS_CRC_ERR, [54](#)
 - L2_STATUS_GEN_ERR, [54](#)
 - L2_STATUS_HSK_ERR, [54](#)
 - L2_STATUS_NO_RESP, [55](#)
 - L2_STATUS_NO_SESSION, [54](#)
 - L2_STATUS_REQUEST_CONT, [54](#)
 - L2_STATUS_REQUEST_OK, [54](#)
 - L2_STATUS_RESULT_CONT, [54](#)
 - L2_STATUS_RESULT_OK, [54](#)
 - L2_STATUS_TAG_ERR, [54](#)
 - L2_STATUS_UNKNOWN_ERR, [54](#)
 - ts_l2_encrypted_cmd, [55](#)
 - ts_l2_frame_check, [55](#)
 - ts_l2_transfer, [55](#)
- ts_l2_api.h, [56](#)
- TS_L2_DATA_LEN_ERROR
 - ts_common.h, [45](#)
- ts_l2_encrypted_cmd
 - [L2 functions], [18](#)
 - ts_l2.h, [55](#)
- ts_l2_frame_check
 - [L2 functions], [17](#)
 - ts_l2.h, [55](#)
- TS_L2_IN_CRC_ERR
 - ts_common.h, [45](#)
- ts_l2_transfer
 - [L2 functions], [17](#)
 - ts_l2.h, [55](#)
- ts_l3.h, [57](#)
 - L3_RESULT_FAIL, [58](#)
 - L3_RESULT_INVALID_CMD, [58](#)
 - L3_RESULT_OK, [58](#)
 - L3_RESULT_UNAUTHORIZED, [58](#)
 - ts_l3_cmd, [59](#)
 - ts_l3_nonce_increase, [58](#)
 - ts_l3_nonce_init, [58](#)
- ts_l3_api.h, [59](#)
 - TS_L3_ECC_KEY_READ_CURVE_P256, [61](#)
 - TS_L3_ECC_KEY_READ_ORIGIN_ECC_KEY_GENERATE, [61](#)
- ts_l3_cmd
 - [L3 functions], [23](#)
 - ts_l3.h, [59](#)
- ts_l3_ecc_key_erase_cmd_t, [28](#)
- ts_l3_ecc_key_erase_res_t, [28](#)
- ts_l3_ecc_key_generate_cmd_t, [26](#)
- ts_l3_ecc_key_generate_res_t, [27](#)
- ts_l3_ecc_key_read_cmd_t, [27](#)
- TS_L3_ECC_KEY_READ_CURVE_P256
 - ts_l3_api.h, [61](#)
- TS_L3_ECC_KEY_READ_ORIGIN_ECC_KEY_GENERATE
 - ts_l3_api.h, [61](#)
- ts_l3_ecc_key_read_res_t, [27](#)
- ts_l3_ecdsa_sign_cmd_t, [30](#)
- ts_l3_ecdsa_sign_res_t, [30](#)
- ts_l3_eddsa_sign_cmd_t, [29](#)
- ts_l3_eddsa_sign_res_t, [29](#)
- ts_l3_nonce_increase
 - [L3 functions], [23](#)
 - ts_l3.h, [58](#)
- ts_l3_nonce_init
 - [L3 functions], [23](#)
 - ts_l3.h, [58](#)
- ts_l3_ping_cmd_t, [24](#)
- ts_l3_ping_res_t, [25](#)
- ts_l3_random_value_get_cmd_t, [25](#)
- ts_l3_random_value_get_res_t, [26](#)
- TS_OK
 - ts_common.h, [45](#)
- TS_PARAM_ERR
 - ts_common.h, [45](#)
- ts_ping, [7](#)
 - libtropic.h, [35](#)
 - ts_ping, [7](#)
- ts_random.h, [61](#)
 - ts_random_bytes, [62](#)
- ts_random_bytes
 - ts_random.h, [62](#)
- ts_random_get, [7](#)
 - libtropic.h, [36](#)
 - ts_random_get, [8](#)
- ts_ret_t
 - ts_common.h, [45](#)
- ts_ret_verbose
 - ts_common.h, [45](#)
- ts_sha256.h, [62](#)
 - ts_sha256_finish, [64](#)
 - ts_sha256_init, [63](#)
 - ts_sha256_start, [63](#)
 - ts_sha256_update, [64](#)
- ts_sha256_ctx, [63](#)
- ts_sha256_finish



ts_sha256.h, [64](#)
ts_sha256_init
ts_sha256.h, [63](#)
ts_sha256_start
ts_sha256.h, [63](#)
ts_sha256_update
ts_sha256.h, [64](#)
ts_X25519
ts_x25519.h, [65](#)
ts_x25519.h, [64](#)
ts_X25519, [65](#)
ts_X25519_scalarmult, [65](#)
ts_X25519_scalarmult
ts_x25519.h, [65](#)