

Trusted Application Programming Reference on Portable TEE

The National Institute of Advanced Industrial Science and Technology

2021-12-02

1 Overview of ta-ref	1
1.1 Features	1
1.1.1 What we did on RISC-V	1
1.1.2 Separate GP TEE Internal API	2
1.2 Diagram	3
1.2.1 Dependency of category	3
2 How to Program on ta-ref	3
2.1 Time Functions	3
2.2 Random Functions	4
2.3 Hash Functions	4
2.4 Symmetric Crypto Functions	4
2.5 Asymmetric Crypto Functions	5
2.6 Symmetric Crypto Gcm Functions	6
2.7 Open, Read, Write, Close On Secure Storage	8
3 Preparation before building	8
3.1 Keystone(RISC-V Unleashed)	8
3.1.1 Required Packages	9
3.1.2 Build Keystone	9
3.1.3 Run Keystone examples	9
3.2 OPTEE (ARM64 RPI3)	9
3.2.1 Required Packages	10
3.2.2 Build OPTEE v3.9.0	10
3.2.3 Run OPTEE Examples	11
3.3 SGX (Intel NUC)	11
3.3.1 List of machines which are confirmed to work	12
3.3.2 BIOS Versions which are failed or succeeded in IAS Test	12
3.3.3 BIOS Settings	12
3.3.4 Required Packages	12
3.3.5 Build SGX	12
3.3.6 Run sgx-ra-sample	14
3.4 Investigation of MbedTLS Configuration file	16
3.4.1 mbedtls configuration file (config.h)	16
3.4.2 Supplement Investigation information	18
4 Building	18
4.1 Install Doxygen-1.9.2	18
4.2 Install Required Packages	18
4.3 Build and Install	19
4.4 ta-ref with Keystone	19
4.4.1 Cloning source and building	19
4.4.2 Check ta-ref by running test_gp, test_hello, on QEMU	19

4.5 ta-ref with OPTEE	21
4.5.1 Cloning source and building	21
4.5.2 Check ta-ref by running test_gp, test_hello, on QEMU	21
4.6 ta-ref with SGX	23
4.6.1 Cloning source and building	23
4.6.2 Check ta-ref by running test_gp, test_hello, simulation mode on any pc	23
5 Running on Dev Boards	25
5.1 Keystone, Unleashed	25
5.1.1 Preparation of rootfs on SD Card	25
5.1.2 Copying binaries of test_hello and test_gp	26
5.1.3 Check test_hello and test_gp on Unleashed	26
5.2 OPTEE, RPI3	28
5.2.1 Preparation of rootfs on SD Card	28
5.2.2 Copying binaries of test_hello and test_gp to rootfs partition	29
5.2.3 Check test_hello and test_gp	29
5.3 SGX, NUC	30
5.3.1 Copying binaries of test_hello and test_gp to NUC machine	30
5.3.2 Check test_hello and test_gp	30
6 API Compare With Full-Set of GP API	33
6.1 GP API	33
7 Class Index	35
7.1 Class List	35
8 File Index	36
8.1 File List	36
9 Class Documentation	36
9.1 __TEE_ObjectHandle Struct Reference	36
9.1.1 Member Data Documentation	37
9.2 __TEE_OperationHandle Struct Reference	38
9.2.1 Member Data Documentation	38
9.3 addrinfo Struct Reference	39
9.3.1 Member Data Documentation	40
9.4 enclave_report Struct Reference	40
9.4.1 Member Data Documentation	41
9.5 pollfd Struct Reference	41
9.5.1 Member Data Documentation	41
9.6 report Struct Reference	42
9.6.1 Member Data Documentation	42
9.7 sm_report Struct Reference	43
9.7.1 Member Data Documentation	43

9.8 TEE.Attribute Struct Reference	43
9.8.1 Member Data Documentation	44
9.9 TEE.Identity Struct Reference	45
9.9.1 Member Data Documentation	45
9.10 TEE.ObjectInfo Struct Reference	46
9.10.1 Member Data Documentation	46
9.11 TEE.OperationInfo Struct Reference	47
9.11.1 Member Data Documentation	47
9.12 TEE.OperationInfoKey Struct Reference	48
9.12.1 Member Data Documentation	49
9.13 TEE.OperationInfoMultiple Struct Reference	49
9.13.1 Member Data Documentation	49
9.14 TEE.Param Union Reference	50
9.14.1 Member Data Documentation	51
9.15 TEE.SEAID Struct Reference	51
9.15.1 Member Data Documentation	52
9.16 TEE.SERedReaderProperties Struct Reference	52
9.16.1 Member Data Documentation	52
9.17 TEE.Time Struct Reference	53
9.17.1 Member Data Documentation	53
9.18 TEE.UUID Struct Reference	53
9.18.1 Member Data Documentation	53
9.19 TEEC.Context Struct Reference	54
9.19.1 Detailed Description	54
9.19.2 Member Data Documentation	54
9.20 TEEC.Operation Struct Reference	55
9.20.1 Detailed Description	55
9.20.2 Member Data Documentation	55
9.21 TEEC.Parameter Union Reference	56
9.21.1 Detailed Description	56
9.21.2 Member Data Documentation	57
9.22 TEEC.RegisteredMemoryReference Struct Reference	57
9.22.1 Detailed Description	58
9.22.2 Member Data Documentation	58
9.23 TEEC.Session Struct Reference	59
9.23.1 Detailed Description	59
9.23.2 Member Data Documentation	59
9.24 TEEC.SharedMemory Struct Reference	59
9.24.1 Detailed Description	60
9.24.2 Member Data Documentation	60
9.25 TEEC.TempMemoryReference Struct Reference	61
9.25.1 Detailed Description	61

9.25.2 Member Data Documentation	61
9.26 TEEC.UUID Struct Reference	62
9.26.1 Detailed Description	62
9.26.2 Member Data Documentation	62
9.27 TEEC.Value Struct Reference	63
9.27.1 Detailed Description	63
9.27.2 Member Data Documentation	63
10 File Documentation	64
10.1 ta-ref/api/include/compiler.h File Reference	64
10.2 compiler.h	64
10.3 ta-ref/api/include/report.h File Reference	67
10.4 report.h	67
10.5 ta-ref/api/include/tee-common.h File Reference	68
10.5.1 Detailed Description	68
10.6 tee-common.h	69
10.7 ta-ref/api/include/tee-ta-internal.h File Reference	69
10.7.1 Detailed Description	72
10.7.2 Function Documentation	72
10.8 tee-ta-internal.h	94
10.9 ta-ref/api/include/tee_api_defines.h File Reference	96
10.10 tee_api_defines.h	96
10.11 ta-ref/api/include/tee_api_defines_extensions.h File Reference	102
10.12 tee_api_defines_extensions.h	102
10.13 ta-ref/api/include/tee_api_types.h File Reference	104
10.13.1 Typedef Documentation	105
10.13.2 Enumeration Type Documentation	107
10.14 tee_api_types.h	107
10.15 ta-ref/api/include/tee_client_api.h File Reference	110
10.15.1 Typedef Documentation	112
10.15.2 Function Documentation	112
10.16 tee_client_api.h	116
10.17 ta-ref/api/include/tee_internal_api.h File Reference	119
10.18 tee_internal_api.h	119
10.19 ta-ref/api/include/tee_internal_api_extensions.h File Reference	119
10.19.1 Function Documentation	120
10.20 tee_internal_api_extensions.h	121
10.21 ta-ref/api/include/tee_ta_api.h File Reference	122
10.21.1 Function Documentation	122
10.22 tee_ta_api.h	123
10.23 ta-ref/api/include/test_dev_key.h File Reference	126
10.23.1 Variable Documentation	126

10.24 test_dev_key.h	127
10.25 ta-ref/api/include/trace.h File Reference	127
10.25.1 Function Documentation	128
10.25.2 Variable Documentation	129
10.26 trace.h	129
10.27 ta-ref/api/include/trace_levels.h File Reference	132
10.28 trace_levels.h	132
10.29 ta-ref/api/keystone/tee-internal-api-machine.c File Reference	133
10.29.1 Function Documentation	133
10.30 ta-ref/api/keystone/tee-internal-api.c File Reference	135
10.30.1 Function Documentation	136
10.30.2 Variable Documentation	145
10.31 ta-ref/api/sgx/tee-internal-api.c File Reference	145
10.31.1 Function Documentation	147
10.31.2 Variable Documentation	154
10.32 ta-ref/api/keystone/tee_api_tee_types.h File Reference	154
10.33 tee_api_tee_types.h	155
10.34 ta-ref/api/optee/tee_api_tee_types.h File Reference	157
10.35 tee_api_tee_types.h	157
10.36 ta-ref/api/sgx/tee_api_tee_types.h File Reference	157
10.37 tee_api_tee_types.h	158
10.38 ta-ref/api/keystone/teeec_stub.c File Reference	160
10.38.1 Function Documentation	160
10.39 ta-ref/api/keystone/trace.c File Reference	163
10.39.1 Function Documentation	164
10.40 ta-ref/api/tee-internal-api-cryptlib.c File Reference	165
10.40.1 Function Documentation	167
10.41 ta-ref/docs/building.md File Reference	179
10.42 ta-ref/docs/gp_api.md File Reference	179
10.43 ta-ref/docs/how_to_program_on_ta-ref.md File Reference	179
10.44 ta-ref/docs/overview_of_ta-ref.md File Reference	179
10.45 ta-ref/docs/preparation.md File Reference	179
10.46 ta-ref/docs/running_on_dev_boards.md File Reference	179

Index

181

1 Overview of ta-ref

1.1 Features

1.1.1 What we did on RISC-V

- We designed the GP internal API library to be portable.

- Keystone SDK is utilized because of runtime "Eyrie".
- The library is ported to Intel SGX as well as RISC-V Keystone.
- Implementation Challenge
 - The combination of GP internal API and cipher suite is big.
 - * We pick up some important GP internal APIs.
 - Some APIs depend on CPU architecture.
 - * We separate APIs into CPU architecture dependent / independent.
 - Integrate GP TEE Internal API to Keystone SDK.
 - * Keystone SDK includes EDL (Enclave Definition Language) named "keedger".
 - * Keedger creates the code for OCALL (request from TEE to REE) to check the pointer and boundary.

1.1.2 Separate GP TEE Internal API

- CPU architecture dependent
 - Random Generator, Time, Secure Storage, Transient Object(TEE_GenerateKey)
- CPU architecture independent(Crypto)
 - Transient Object(exclude TEE_GenerateKey), Crypto Common, Authenticated Encryption, Symmetric/Asymmetric Cipher, Message Digest

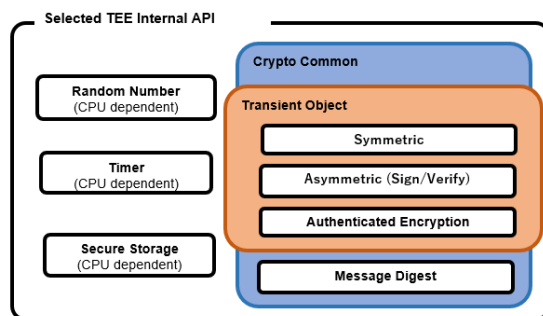
Category	CPU (In)Dependent	Functions
Random Number	Dependent	TEE_GenerateRandom
Time	Dependent	TEE_GetREETime, TEE_GetSystemTime
Secure Storage	Dependent	TEE_CreatePersistentObject, TEE_OpenPersistentObject, TEE_ReadObjectData, TEE_WriteObjectData, TEE_CloseObject
Transient Object	Dependent Independent	TEE_GenerateKey, TEE_AllocateTransientObject, TEE_FreeTransientObject, TEE_InitRefAttribute, TEE_InitValueAttribute, TEE_SetOperationKey
Crypto Common	Independent	TEE_AllocateOperation, TEE_FreeOperation
Authenticated Encryption	Independent	TEE_AEInit, TEE_AEUpdateAAD, TEE_AEUpdate, TEE_AEEncryptFinal, TEE_AEDecryptFinal
Symmetric Cipher	Independent	TEE_CipherInit, TEE_CipherUpdate, TEE_CipherDoFinal
Asymmetric Cipher	Independent	TEE_AsymmetricSignDigest, TEE_AsymmetricVerifyDigest
Message Digest	Independent	TEE_DigestUpdate, TEE_DigestDoFinal

1.2 Diagram

1.2.1 Dependency of category

Dependency of category

- Some categories have dependency.
 - Crypto Common
 - Cipher suite must be registered before use.
 - Transient Object
 - The space for a key must be prepared before use.



Sample Program

```
// Allocate a transient object for keypair
TEE_AllocateTransientObject(TEE_TYPE_ECDSA_KEYPAIR,
    KEY_SIZE, &keypair);
// Assemble an attribute for ecc key
TEE_InitValueAttribute(&attr, TEE_ATTR_ECC_CURVE,
    TEE_ECC_CURVE_NIST_P256, KEY_SIZE);
// Generate a keypair having that attribute
TEE_GenerateKey(keypair, KEY_SIZE, &attr, 1);
```

```
// Allocate sign operation
TEE_AllocateOperation(&handle, TEE_ALG_ECDSA_P256,
    TEE_MODE_SIGN, KEY_SIZE);
```

```
// Set the generated key to the sign operation
TEE_SetOperationKey(handle, keypair);
```

```
// Sign
uint32_t siglen = SIG_LENGTH;
TEE_AsymmetricSignDigest(handle, NULL, 0, hash,
    hashlen, sig, &siglen);
```

```
// Free handle for the sign operation
TEE_FreeOperation(handle);
```

Legend:

- Crypto Common
- Transient Object
- Asymmetric (Sign/Verify)

2 How to Program on ta-ref

2.1 Time Functions

This function retrieves the current time as seen from the point of view of the REE, which expressed in the number of seconds and prints the "GP REE second and millisecond".

```
--- start Ree time ---
void gp_ree_time_test(void)
{
    TEE_Time time;
    /* REE time */
    TEE_GetREETime(&time);
    tee_printf("@GP REE time %u sec %u millis\n", time.seconds, time.millis);
}
--- end Ree Time ---
```

This function retrieves the current system time as seen from the point of view of the TA, which expressed in the number of seconds and print the "GP System time second and millisecond".

```
--- start System time ---
void gp_trusted_time_test(void)
{
    TEE_Time time;
    /* System time */
    TEE_GetSystemTime(&time);
    tee_printf("@GP System time %u sec %u millis\n", time.seconds, time.millis);
}
--- end System time ---
```


2.2 Random Functions

This function generates the random data by invoking TEE_GenerateRandom function and it prints the generated random data.

```
--- start Random ---
void gp_random_test(void)
{
    unsigned char rbuf[16];
    TEE_GenerateRandom(rbuf, sizeof(rbuf));
    tee_printf("@random: ");
    for (int i = 0; i < sizeof(rbuf); i++) {
        tee_printf ("%02x", rbuf[i]);
    }
    tee_printf("\n");
}
--- end Random ---
```

2.3 Hash Functions

Pseudo code of how to use Message Digest Functions. Keystone uses sha3.c which is almost identical. Ultimate question is whether this should be done in 'Enclave (U-Mode) or Runtime (S-Mode) the library used in keystone.↔ The function performs many operations to achieve message data hash techniques to allocate the handle for a new cryptographic operation. And then finalize the message digest operation to produce the message hash. It prints the hash message.

```
--- start Message Digest ---
void gp_message_digest_test(void)
{
    static unsigned char data[256] = {
        // 0x00,0x01,...,0xff
#include "test.dat"
    };
    unsigned char hash[SHA_LENGTH];
    TEE_OperationHandle handle;
    uint32_t hashlen = SHA_LENGTH;
    TEE_Result rv;
    // Take hash of test data
    /* sha3_init() in sha3.c */
    rv = TEE_AllocateOperation(&handle, TEE_ALG_SHA256, TEE_MODE_DIGEST, SHA_LENGTH);
    GP_ASSERT(rv, "TEE_AllocateOperation fails");
    /* sha3_update() in sha3.c */
    TEE_DigestUpdate(handle, data, sizeof(data));

    /* sha3_final() in sha3.c */
    rv = TEE_DigestDoFinal(handle, NULL, 0, hash, &hashlen);
    GP_ASSERT(rv, "TEE_DigestDoFinal fails");
    TEE_FreeOperation(handle);
    /* hash value is ready */
    // Dump hashed data
    tee_printf("hash: ");
    for (int i = 0; i < SHA_LENGTH; i++) {
        tee_printf ("%02x", hash[i]);
    }
    tee_printf("\n");
}
--- end Message Digest ---
```

2.4 Symmetric Crypto Functions

Crypto, Authenticated Encryption with Symmetric Key Verification Functions. This function allocates an uninitialized transient object, i.e. a container for attributes. Transient objects are used to hold a cryptographic object (key or key-pair). With the generation of a key, a new cryptographic operation for encrypt and decrypt data is initiated with a symmetric cipher operation. The original data is compared with decrypted data by checking the data and its length.

```
--- start Symmetric Key Encryption ---
void gp_symmetric_key_enc_verify_test(void)
{
    TEE_OperationHandle handle;
    static unsigned char data[CIPHER_LENGTH] = {
        // 0x00,0x01,...,0xff
#include "test.dat"
    };
    /* ... */
}
```

```

uint8_t iv[16];
unsigned char out[CIPHER_LENGTH];
uint32_t outlen;
TEE_ObjectHandle key;
TEE_Result rv;
// Generate key
rv = TEE_AllocateTransientObject(TEE_TYPE_AES, 32*8, &key);
GP_ASSERT(rv, "TEE_AllocateTransientObject fails");
rv = TEE_GenerateKey(key, 256, NULL, 0);
GP_ASSERT(rv, "TEE_GenerateKey fails");
// Encrypt test data
rv = TEE_AllocateOperation(&handle, TEE_ALG_AES_CBC_NOPAD, TEE_MODE_ENCRYPT, 256);
GP_ASSERT(rv, "TEE_AllocateOperation fails");
rv = TEE_SetOperationKey(handle, key);
GP_ASSERT(rv, "TEE_SetOperationKey fails");
TEE_GenerateRandom(iv, sizeof(iv));
TEE_CipherInit(handle, iv, sizeof(iv));
//GP_ASSERT(rv, "TEE_AEInit fails");
outlen = CIPHER_LENGTH;
rv = TEE_CipherUpdate(handle, data, CIPHER_LENGTH, out, &outlen);
GP_ASSERT(rv, "TEE_CipherUpdate fails");
TEE_FreeOperation(handle);
// Dump encrypted data
tee_printf("@cipher: ");
for (int i = 0; i < CIPHER_LENGTH; i++) {
    tee_printf("%02x", out[i]);
}
tee_printf("\n");
// Decrypt it
rv = TEE_AllocateOperation(&handle, TEE_ALG_AES_CBC_NOPAD, TEE_MODE_DECRYPT, 256);
GP_ASSERT(rv, "TEE_AllocateOperation fails");
rv = TEE_SetOperationKey(handle, key);
GP_ASSERT(rv, "TEE_SetOperationKey fails");
TEE_CipherInit(handle, iv, sizeof(iv));
//GP_ASSERT(rv, "TEE_AEInit fails");
outlen = CIPHER_LENGTH;
rv = TEE_CipherUpdate(handle, out, CIPHER_LENGTH, out, &outlen);
GP_ASSERT(rv, "TEE_CipherUpdate fails");
TEE_FreeOperation(handle);
TEE_FreeTransientObject(key);
// Dump data
tee_printf("decrypted to: ");
for (int i = 0; i < CIPHER_LENGTH; i++) {
    tee_printf("%02x", out[i]);
}
tee_printf("\n");
// Verify decrypted data against original one
int verify_ok;
verify_ok = !memcmp(out, data, CIPHER_LENGTH);
if (verify_ok) {
    tee_printf("verify ok\n");
} else {
    tee_printf("verify fails\n");
}
}
--- end Symmetric Key Encryption ---

```

2.5 Asymmetric Crypto Functions

Crypto, Sign and Verify with Asymmetric Key Verification Functions. Cryptographic Operations for API Message Digest Functions. The function performs cryptographic operation for API Message. To achieve this, the function allocates a handle for a new cryptographic operation, to finalize the message digest operation and to produce the message hash. The Hashed data is signed with signature key within an asymmetric operation. The original Hashed Data and Signed hashed data is compared for ok status.

```

--- start Asymmetric Key Signed ---
void gp_asymmetric_key_sign_test(void)
{
    static unsigned char data[256] = {
        // 0x00, 0x01, ..., 0xff
#include "test.dat"
    };
    unsigned char hash[SHA_LENGTH];
    unsigned char sig[SIG_LENGTH];
    TEE_OperationHandle handle;
    uint32_t hashlen = SHA_LENGTH;
    TEE_Result rv;

    // Take hash of test data
    /* Calculate hash */
    /* sha3_init() in sha3.c */

```

```

rv = TEE.AllocateOperation(&handle, TEE.ALG_SHA256, TEE.MODE_DIGEST, SHA_LENGTH);
GP_ASSERT(rv, "TEE.AllocateOperation fails");
/* sha3.update() in sha3.c */
TEE.DigestUpdate(handle, data, sizeof(data));

/* sha3.final() in sha3.c */
rv = TEE.DigestDoFinal(handle, NULL, 0, hash, &hashlen);
GP_ASSERT(rv, "TEE.DigestDoFinal fails");
/* free up */
TEE.FreeOperation(handle);
/* Get the signature */
// Dump hashed data
tee.printf("@digest: ");
for (int i = 0; i < SHA_LENGTH; i++) {
    tee.printf ("%02x", hash[i]);
}
tee.printf("\n");
uint32_t siglen = SIG_LENGTH;
TEE.ObjectHandle keypair;
// Sign hashed data with the generated keys
/* set ecdsa.p256 key */
rv = TEE.AllocateOperation(&handle, TEE.ALG_ECDSA_P256, TEE.MODE_SIGN, 256);
GP_ASSERT(rv, "TEE.AllocateOperation fails");
// Generate keypair
rv = TEE.AllocateTransientObject(TEE.TYPE_ECDSA_KEYPAIR, 256, &keypair);
GP_ASSERT(rv, "TEE.AllocateTransientObject fails");
TEE.Attribute attr;
TEE.InitValueAttribute(&attr,
    TEE.ATTR_ECC_CURVE,
    TEE.ECC_CURVE_NIST_P256,
    256);
rv = TEE.GenerateKey(keypair, 256, &attr, 1);
GP_ASSERT(rv, "TEE.GenerateKey fails");
rv = TEE.SetOperationKey(handle, keypair);
GP_ASSERT(rv, "TEE.SetOperationKey fails");
/* Keystone has ecDSA.p256.sign() Equivalent in openssl is EVP.DigestSign() */
rv = TEE.AsymmetricSignDigest(handle, NULL, 0, hash, hashlen, sig, &siglen);
GP_ASSERT(rv, "TEE.AsymmetricSignDigest fails");

/* free up */
TEE.FreeOperation(handle);
/* Get the signature */
// Dump signature
tee.printf("@signature: ");
for (uint32_t i = 0; i < siglen; i++) {
    tee.printf ("%02x", sig[i]);
}
tee.printf("\n");
// Verify signature against hashed data
/* set ecDSA.p256 key */
rv = TEE.AllocateOperation(&handle, TEE.ALG_ECDSA_P256, TEE.MODE_VERIFY, 256);
GP_ASSERT(rv, "TEE.AllocateOperation fails");
rv = TEE.SetOperationKey(handle, keypair);
GP_ASSERT(rv, "TEE.SetOperationKey fails");
/* Keystone has ecDSA.p256.verify() Equivalent in openssl is EVP.DigestVerify() */
TEE.Result verify_ok;
verify_ok = TEE.AsymmetricVerifyDigest(handle, NULL, 0, hash, hashlen, sig, siglen);
/* free up */
TEE.FreeOperation(handle);
tee.printf("@@TEE.FreeOperation: \n");
TEE.FreeTransientObject(keypair);

if (verify_ok == TEE.SUCCESS) {
    tee.printf("verify ok\n");
} else {
    tee.printf("verify fails\n");
}
}
/* Check verify_ok for success of verification */
--- end Asymmetric Key Signed ---

```

2.6 Symmetric Crypto Gcm Functions

This function encrypt and decrypt the test data. The function allocates an uninitialized transient object, i.e. a container for attributes. Transient objects are used to hold a cryptographic object (key or key-pair). With the generation of a key, a new cryptographic operation for encrypt and decrypt data is initiated with a symmetric cipher operation. The data is also checked whether it is completely encrypted or decrypted. The original data is compared with decrypted data by checking the data and cipher length.

```

--- start Symmetric Key GCM ---
void gp_symmetric_key_gcm_verify_test(void)

```

```

{
    TEEOperationHandle handle;
    static unsigned char data[CIPHER_LENGTH] = {
        // 0x00,0x01,...,0xff
#include "test.dat"
    };
    uint8_t iv[16];
    unsigned char out[CIPHER_LENGTH];
    uint32_t outlen;
    unsigned char tag[16];
    TEEObjectHandle key;
    TEEResult rv;
    // Generate key
    rv = TEE.AllocateTransientObject(TEE.TYPE_AES, 256, &key);
    GP_ASSERT(rv, "TEE.AllocateTransientObject fails");
    rv = TEE.GenerateKey(key, 256, NULL, 0);
    GP_ASSERT(rv, "TEE.GenerateKey fails");
    // Encrypt test data
    rv = TEE.AllocateOperation(&handle, TEE.ALG_AES_GCM, TEE.MODE_ENCRYPT, 256);
    GP_ASSERT(rv, "TEE.AllocateOperation fails");
    rv = TEE.SetOperationKey(handle, key);
    GP_ASSERT(rv, "TEE.SetOperationKey fails");
    TEE.GenerateRandom(iv, sizeof(iv));
    /* Equivalent in openssl is EVP_EncryptInit_ex() */
    rv = TEE.AEInit(handle, iv, sizeof(iv), 16*8, 16, 16);
    GP_ASSERT(rv, "TEE.AEInit fails");
    /* Equivalent in openssl is EVP_EncryptUpdate() */
    // rv = TEE.AEUpdateAAD(handle, aad, 16);
    // GP_ASSERT(rv, "TEE.AEUpdateAAD fails");
    unsigned int taglen = 16;
    memset(tag, 0, 16);
    outlen = CIPHER_LENGTH;
    /* Equivalent in openssl is EVP_EncryptFinal() */
    rv = TEE.AEEncryptFinal(handle, data, 256, out, &outlen, tag, &taglen);
    TEE.FreeOperation(handle);
    /* Get the auth.tag */
    // Dump encrypted data and tag
    tee_printf("@cipher: ");
    for (int i = 0; i < CIPHER_LENGTH; i++) {
        tee_printf ("%02x", out[i]);
    }
    tee_printf("\n");
    tee_printf("@tag: ");
    for (int i = 0; i < 16; i++) {
        tee_printf ("%02x", tag[i]);
    }
    tee_printf("\n");
    // Decrypt it
    rv = TEE.AllocateOperation(&handle, TEE.ALG_AES_GCM, TEE.MODE_DECRYPT, 256);
    GP_ASSERT(rv, "TEE.AllocateOperation fails");
    rv = TEE.SetOperationKey(handle, key);
    GP_ASSERT(rv, "TEE.SetOperationKey fails");
    /* Equivalent in openssl is EVP_DecryptInit_ex() */
    rv = TEE.AEInit(handle, iv, sizeof(iv), 16*8, 16, 16);
    GP_ASSERT(rv, "TEE.AEInit fails");
    // rv = TEE.AEUpdateAAD(handle, aad, 16);
    // GP_ASSERT(rv, "TEE.AEUpdateAAD fails");
    unsigned char decode[CIPHER_LENGTH];
    outlen = 256;
    /* Equivalent in openssl require two functions
       EVP_CIPHER_CTX_ctrl(tag) and EVP_DecryptFinal(others) */
    rv = TEE.AEDecryptFinal(handle, out, 256, decode, &outlen, tag, 16);
    GP_ASSERT(rv, "TEE.AEDecryptFinal fails");
    TEE.FreeOperation(handle);
    TEE.FreeTransientObject(key);
    // Dump data and tag
    tee_printf("decrypted to: ");
    for (int i = 0; i < CIPHER_LENGTH; i++) {
        tee_printf ("%02x", decode[i]);
    }
    tee_printf("\n");

    // Verify decrypted data against original one
    /* Check verify_ok for success of decrypting and authentication */
    int verify_ok;
    verify_ok = !memcmp(decode, data, CIPHER_LENGTH);
    if (verify_ok) {
        tee_printf("verify ok\n");
    } else {
        tee_printf("verify fails\n");
    }
}
--- end Symmetric Key GCM ---

```

2.7 Open, Read, Write, Close On Secure Storage

Core Functions, Secure Storage Functions. Pseudo code of how to use Secure Storage. These could be implemented using ocall on Keystone. Almost identical to open(), clone(), read(), write() in POSIX API. The function creates a persistent object for reading and writing the data. The created data individually for read and write are compared for data length. If the length of both the objects are same, the function prints "verify ok" and prints "verify fails" if it is not the same.

```

--- start Secure storage ---
void gp.secure_storage.test(void)
{
    static unsigned char data[] = {
        // 0x00,0x01,...,0xff
#include "test.dat"
    };
    static unsigned char buf[DATA_LENGTH];
    TEE.Result rv;
    /* write */
    TEE.ObjectHandle object;
    rv = TEE.CreatePersistentObject(TEE.STORAGE_PRIVATE,
                                   "FileOne", strlen("FileOne"),
                                   (TEE.DATA_FLAG_ACCESS.WRITE
                                    | TEE.DATA_FLAG_OVERWRITE),
                                   TEE.HANDLE_NULL,
                                   NULL, 0,
                                   &object);
    GP_ASSERT(rv, "TEE.CreatePersistentObject fails");
    memcpy(buf, data, DATA_LENGTH);
    /* fill the date in buffer */
    rv = TEE.WriteObjectData(object, (const char *)data, DATA_LENGTH);
    GP_ASSERT(rv, "TEE.WriteObjectData fails");
    TEE.CloseObject(object);
--- write file end ---
    /* clear buf */
    memset(buf, 0, DATA_LENGTH);
--- read file start ---
    /* read */
    rv = TEE.OpenPersistentObject(TEE.STORAGE_PRIVATE,
                                   "FileOne", strlen("FileOne"),
                                   TEE.DATA_FLAG_ACCESS.READ,
                                   &object);
    GP_ASSERT(rv, "TEE.OpenPersistentObject fails");
    uint32_t count;
    rv = TEE.ReadObjectData(object, (char *)buf, DATA_LENGTH, &count);

    GP_ASSERT(rv, "TEE.ReadObjectData fails");
    TEE.CloseObject(object);
    /* use the date in buffer */
    tee_printf("%d bytes read: ", count);
    for (uint32_t i = 0; i < count; i++) {
        tee_printf ("%02x", buf[i]);
    }
    tee_printf("\n");
    /* Compare read data with written data */
    int verify_ok;
    verify_ok = !memcmp(buf, data, DATA_LENGTH);
    if (verify_ok) {
        tee_printf("verify ok\n");
    } else {
        tee_printf("verify fails\n");
    }
}
--- end Secure storage ---

```

3 Preparation before building

3.1 Keystone(RISC-V Unleashed)

Keystone is an open-source TEE framework for RISC-V processors. For more details check,

- <http://docs.keystone-enclave.org/en/latest>

3.1.1 Required Packages

Install following Packages

```
apt-get update
apt-get install -y autoconf automake autotools-dev bc bison build-essential curl expat libexpat1-dev flex
gawk gcc git gperf libgmp-dev libmpc-dev libmpfr-dev libtool texinfo tmux patchutils zlib1g-dev wget
bzip2 patch vim-common lbzip2 python pkg-config libglib2.0-dev libpixmap-1-dev libssl-dev screen
device-tree-compiler expect makeelf unzip cpio rsync cmake
```

3.1.2 Build Keystone

Download the keystone sources

```
git clone https://github.com/keystone-enclave/keystone.git
cd keystone
git checkout v0.3
./fast-setup.sh
make
source source.sh
./sdk/scripts/init.sh
./sdk/examples/hello/vault.sh
./sdk/examples/hello-native/vault.sh
./tests/tests/vault.sh
make image
```

RISC-V Toolchain:

- When you execute `./fast-setup.sh`, the toolchain for RISC-V has been installed at `$KEYSTONE_DIR/riscv/bin` and it adds to your PATH.

3.1.3 Run Keystone examples

Launch QEMU console

```
./scripts/run-qemu.sh
Welcome to Buildroot
```

Login to console with user=root, passwd=sifive

```
buildroot login: root
Password:
$
```

Run hello example

```
$ insmod keystone-driver.ko
[ 365.354299] keystone_driver: loading out-of-tree module taints kernel.
[ 365.364279] keystone_enclave: keystone enclave v0.2
$
$ ./hello/hello.ke
Verifying archive integrity... 100% All good.
Uncompressing Keystone vault archive 100%
hello, world!
```

Poweroff the console incase, if you want to exit.

```
$ poweroff
```

3.2 OPTEE (ARM64 RPI3)

OP-TEE is a Trusted Execution Environment (TEE) designed as companion to a non-secure Linux kernel running on Arm. Lets build OPTEE for QEMU and Raspberry Pi3 Model B development board. For more details check,

- <https://optee.readthedocs.io/en/latest/>

3.2.1 Required Packages

Install following packages on Ubuntu 18.04

```
sudo dpkg --add-architecture i386
sudo apt-get update -y
sudo apt-get install -y android-tools-adb android-tools-fastboot autoconf \
    automake bc bison build-essential ccache cscope curl device-tree-compiler \
    expect flex ftp-upload gdisk iasl libattr1-dev libc6:i386 libcap-dev \
    libfdt-dev libftdi-dev libglib2.0-dev libhidapi-dev libncurses5-dev \
    libpixman-1-dev libssl-dev libstdc++6:i386 libtool libz1:i386 make \
    mtools netcat python python-crypto python3-crypto python-pyelftools \
    python3-pycryptodome python3-pyelftools python3-serial vim-common \
    rsync unzip uuid-dev xdg-utils xterm xz-utils zlib1g-dev \
    git python3-pip wget cpio \
    texlive texinfo \
sudo pip3 install pycryptodomex
```

3.2.2 Build OPTEE v3.9.0

Configure git

```
git config --global user.name "dummy"
git config --global user.email "dummy@gmail.com"
git config --global color.ui false
mkdir ~/bin
curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo && \
chmod a+x ~/bin/repo
```

3.2.2.1 Download Toolchains

```
export TOOLCHAIN_DIR=${HOME}/toolchains
sudo apt-get install -y wget xz-utils
mkdir -p ${TOOLCHAIN_DIR}/aarch64 ${TOOLCHAIN_DIR}/aarch32
wget http://192.168.100.100:2000/gcc-arm-8.3-2019.03-x86_64-arm-linux-gnueabi.tar.xz -O /dev/null -O
aarch32.tar.xz && \
tar xf aarch32.tar.xz --strip-components=1 -C ${TOOLCHAIN_DIR}/aarch32
wget http://192.168.100.100:2000/gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu.tar.xz -O /dev/null -O
aarch64.tar.xz && \
tar xf aarch64.tar.xz --strip-components=1 -C ${TOOLCHAIN_DIR}/aarch64
export PATH=${TOOLCHAIN_DIR}/aarch64/bin:${TOOLCHAIN_DIR}/aarch32/bin:${PATH}
```

3.2.2.2 Clone and Build OPTEE v3.9.0 for QEMU

Clone optee version 3.9.0 for QEMU

```
mkdir optee.3.9.0.qemu
cd optee.3.9.0.qemu
~/bin/repo init -u https://github.com/knknkn1162/manifest.git -m qemu.v8.xml -b 3.9.0
~/bin/repo sync -j4 --no-clone-bundle
ln -s ~/toolchains toolchains
cd build
make
```

If build is successful, the rootfs can be found as follows

```
ls -l ../out-br/images/rootfs.cpio.gz
```

3.2.2.3 Clone and Build OPTEE v3.9.0 for RPI3

Copy the following lines into "optee-rpi3.sh" script

```
#!/bin/bash -u
export OPTEE_VER=$1
export OPTEE_DIR=${PWD}/optee.${OPTEE_VER}-rpi3
mkdir ${OPTEE_DIR} || true
cd ${OPTEE_DIR}
~/bin/repo init -u https://github.com/knknkn1162/manifest.git -m rpi3.xml -b ${OPTEE_VER}
~/bin/repo sync -j4 --no-clone-bundle
ln -s ~/toolchains ${OPTEE_DIR}/. || true
echo 'CONFIG_CMDLINE="console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 root=/dev/mmcblk0p2 rootfstype=ext4
noinitrd rw rootwait init=/lib/systemd/systemd"' > build/defconfig-cmdline.txt
cd build
```

```
make OPTEE_CLIENT_BIN_ARCH_EXCLUDE=/boot
LINUX_DEFCONFIG_COMMON_FILES="${OPTEE_DIR}/linux/arch/arm64/configs/bcmrpi3_defconfig
${OPTEE_DIR}/build/kconfigs/rpi3.conf ${OPTEE_DIR}/build/defconfig-cmdline.txt"
BR2_PACKAGE_OPTEE_OS_EXT=n BR2_PACKAGE_OPTEE_TEST_EXT=n BR2_PACKAGE_OPTEE_EXAMPLES_EXT=n
BR2_TOOLCHAIN_EXTERNAL_GCC_8=y BR2_TOOLCHAIN_EXTERNAL_HEADERS_4_19=y BR2_HOST_GCC_AT_LEAST_8=y
BR2_TOOLCHAIN_HEADERS_AT_LEAST="4.19" -j`nproc`
```

Run the script as follows

```
chmod +x optee-rpi3.sh
./optee-rpi3.sh 3.9.0
```

If build is successful, the rootfs can be found as follows

```
ls -l ../out-br/images/rootfs.cpio.gz
```

3.2.3 Run OPTEE Examples

3.2.3.1 Launching QEMU Console

Run following commands from OPTEE build directory

```
cd $OPTEE_DIR/build
make run
```

Once above command is successful, QEMU is ready

```
* QEMU is now waiting to start the execution
* Start execution with either a 'c' followed by <enter> in the QEMU console or
* attach a debugger and continue from there.
*
* To run OP-TEE tests, use the xtest command in the 'Normal World' terminal
* Enter 'xtest -h' for help.
cd /TEE/demo/rpi3/optee.3.9.0.qemu/build/./out/bin &&
/TEE/demo/rpi3/optee.3.9.0.qemu/build/./qemu/aarch64-softmmu/qemu-system-aarch64 \
-nographic \
-serial tcp:localhost:54320 -serial tcp:localhost:54321 \
-smp 2 \
-s -S -machine virt,secure=on -cpu cortex-a57 \
-d unimp -semihosting-config enable,target=native \
-m 1057 \
-bios bl1.bin \
-initrd rootfs.cpio.gz \
-kernel Image -no-acpi \
-append 'console=ttyAMA0,38400 keep_bootcon root=/dev/vda2' \
-object rng-random,filename=/dev/urandom,id=rng0 -device
virtio-rng-pci,rng=rng0,max-bytes=1024,period=1000 -netdev user,id=vmnic -device
virtio-net-device,netdev=vmnic
QEMU 3.0.93 monitor - type 'help' for more information
(qemu) c
Now Optee started to boot from another tab on the Terminal
```

3.2.3.2 Run hello world example

Once boot completed it displays following message, then enter "root" to login to the shell

```
Welcome to Buildroot, type root or test to login
buildroot login: root
$
$ optee.example.hello.world
Invoking TA to increment 42
TA incremented value to 43
```

Poweroff the console in case, if you want to exit.

```
$ poweroff
```

3.3 SGX (Intel NUC)

Intel(R) Software Guard Extensions (Intel(R) SGX) is an Intel technology for application developers who is seeking to protect selected code and data from disclosure or modification. For more details check,

- <https://github.com/intel/linux-sgx/blob/master/README.md>

3.3.1 List of machines which are confirmed to work

1. Intel NUC7PJYH - Intel(R) Celeron(R) J4005 CPU @ 2.00GHz
2. Intel NUC7PJYH - Intel(R) Pentium(R) Silver J5005 CPU @ 1.50GHz
3. Intel NUC9VXQNX - Intel(R) Xeon(R) E-2286M CPU @ 2.40GHz (Partially working)

3.3.2 BIOS Versions which are failed or succeeded in IAS Test

1. BIOS Version JYGLKCPX.86A.0050.2019.0418.1441 - IAS Test was Failed
2. BIOS Version JYGLKCPX.86A.0053.2019.1015.1510 - IAS Test was Failed
3. BIOS Version JYGLKCPX.86A.0057.2020.1020.1637 - IAS Test was Success
4. BIOS Version QNCFLX70.0034.2019.1125.1424 - IAS Test was Failed
5. BIOS Version QNCFLX70.0059.2020.1130.2122 - IAS Test was Success

Update BIOS from:

- <https://downloadcenter.intel.com/download/29987/BIOS-Update-JYGLKCPX->
- <https://downloadcenter.intel.com/download/30069/BIOS-Update-QNCFLX70->

3.3.3 BIOS Settings

1. Make sure you are running with latest version BIOS
2. Make sure you enabled SGX support in BIOS
3. Make sure `Secure Boot` disabled in BIOS

Refer: <https://github.com/intel/sgx-software-enable/blob/master/README.md>

3.3.4 Required Packages

Intall following packages on Ubuntu 18.04

```
sudo apt-get install build-essential ocaml ocamlbuild automake autoconf libtool wget python libssl-dev git
cmake perl libssl-dev libcurl4-openssl-dev protobuf-compiler libprotobuf-dev debhelper cmake reprepro
expect unzip sshpass
```

3.3.5 Build SGX

There are 3 components which need to be build for SGX

1. linux-sgx
2. linux-sgx-driver
3. sgx-ra-sample

3.3.5.1 SGX SDK

Clone and build

```
git clone https://github.com/intel/linux-sgx.git -b sgx.2.10
cd linux-sgx
git checkout sgx.2.10
./download_prebuilt.sh
sudo cp external/toolset/ubuntu18.04/{as,ld,ld.gold,objdump} /usr/local/bin/
make -j`nproc` sdk_install.pkg DEBUG=1
```

Install SGX SDK

```
sudo ./linux/installer/bin//sgx_linux_x64_sdk.${version}.bin
```

where `${version}` is a string something similar to 2.10.100.2.

Answer the question with `no` and input the install dir as `/opt/intel`

Build and Install SGX PSW packages

See here: <https://github.com/intel/linux-sgx#install-the-intelr-sgx-psw>

```
source /opt/intel/sgxsdk/environment
make deb_psw_pkg DEBUG=1
rm ./linux/installer/deb/*/*sgx-dcap-pccs*.deb
sudo dpkg -i ./linux/installer/deb/*/*.deb
```

Install SGX PSW packages from Intel Repository

See here: <https://github.com/intel/linux-sgx#install-the-intelr-sgx-psw-1>

Using the local repo is recommended, since the system will resolve the dependencies automatically.

Check at page no.7, https://download.01.org/intel-sgx/sgx-linux/2.9/docs/Intel_SGX_Installation_Guide_Linux_2.9_Open_Source.pdf

```
sudo apt install libsgx-enclave-common libsgx-epid libsgx-launch libsgx-urts libsgx-uae-service
libsgx-quote-ex
```

If you see below error,

```
Errors were encountered while processing:
/tmp/apt-dpkg-install-pCB0cR/04-libsgx-headers.2.12.100.3-bionic1.amd64.deb
```

Here is the fix

```
sudo apt -o Dpkg::Options::="--force-overwrite" --fix-broken install
```

3.3.5.2 Build and Install SGX Driver

See [linux-sgx-driver](#).

Caveat: Whenever updating kernel, don't forget rebuilding this driver with new version of the kernel header. (There are a few linux-sgx-driver-dkms repo, though I've experienced troubles with them.)

Clone and build

```
$ git clone https://github.com/intel/linux-sgx-driver.git
$ cd linux-sgx-driver
$ make
```

Install SGX driver

```
$ sudo mkdir -p "/lib/modules/$(uname -r)"/kernel/drivers/intel/sgx"
$ sudo cp isgx.ko "/lib/modules/$(uname -r)"/kernel/drivers/intel/sgx"
$ sudo sh -c "cat /etc/modules | grep -Fxq isgx || echo isgx >> /etc/modules"
$ sudo /sbin/depmod
$ sudo /sbin/modprobe isgx
```

When modprobe fails with "Operation is not permitted", disable secure boot in BIOS. So that the unsigned kernel driver can be installed. If it is success, reboot your machine and verify `sudo lsmod | grep isgx` if it shows `isgx.ko`

3.3.6 Run sgx-ra-sample

3.3.6.1 Build sgx-ra-sample Clone and build OpenSSL 1.1.c

```
wget https://www.openssl.org/source/openssl-1.1.1c.tar.gz
tar xf openssl-1.1.1c.tar.gz
cd openssl-1.1.1c/
./config --prefix=/opt/openssl/1.1.1c --openssldir=/opt/openssl/1.1.1c
make
sudo make install
cd ..
```

Clone and build sgx-ra-sample

```
git clone https://github.com/intel/sgx-ra-sample.git
cd sgx-ra-sample/
./bootstrap
./configure --with-openssldir=/opt/openssl/1.1.1c
make
```

3.3.6.2 Prepare for IAS Test

1. Obtain a subscription key for the Intel SGX Attestation Service Utilizing Enhanced Privacy ID (EPID). See here: <https://api.portal.trustedservices.intel.com/EPID-attestation>
2. Download Intel.SGX.Attestation.RootCA.pem from above portal.
3. Edit settings file and update the file with your own values obtained from portal.

```
@@ -15,14 +15,14 @@ QUERY_IAS_PRODUCTION=0
# Your Service Provider ID. This should be a 32-character hex string.
# [REQUIRED]

-SPID=0123456789ABCDEF0123456789ABCDEF
+SPID=EF9AE4A8635825B88751C8698CB370B4

# Set to a non-zero value if this SPID is associated with linkable
# quotes. If you change this, you'll need to change SPID,
# IAS_PRIMARY_SUBSCRIPTION_KEY and IAS_SECONDARY_SUBSCRIPTION_KEY too.

-LINKABLE=0
+LINKABLE=1

#=====
@@ -50,18 +50,18 @@ USE_PLATFORM_SERVICES=0
# More Info: https://api.portal.trustedservices.intel.com/EPID-attestation
# Associated SPID above is required

-IAS_PRIMARY_SUBSCRIPTION_KEY=
+IAS_PRIMARY_SUBSCRIPTION_KEY=b6da4c9c41464924a14954ad8c03e8cf

# Intel Attestation Service Secondary Subscription Key
# This will be used in case the primary subscription key does not work

-IAS_SECONDARY_SUBSCRIPTION_KEY=
+IAS_SECONDARY_SUBSCRIPTION_KEY=188d91f86c064deb97e7472175ae1e79

# The Intel IAS SGX Report Signing CA file. You are sent this certificate
# when you apply for access to SGX Developer Services at
# http://software.intel.com/sgx [REQUIRED]

-IAS_REPORT_SIGNING_CA_FILE=
+IAS_REPORT_SIGNING_CA_FILE=./Intel.SGX.Attestation.RootCA.pem

# Debugging options
@@ -82,7 +82,7 @@ IAS_REPORT_SIGNING_CA_FILE=

# Set to non-zero for verbose output

-VERBOSE=0
+VERBOSE=1
```

3.3.6.3 Run IAS Test

Run "run-server"

```
./run-server
Listening for connections on port 7777
Waiting for a client to connect...
Connection from 127.0.0.1
Waiting for msg0||msg1
---- Copy/Paste Msg2 Below to Client -----
44f2e22125f052e0118018febceefd469f1f4b73de22b34ebdf27003605963946bb862980be691a2f532d8e66abb2ce4eaad
f778593blce7a14b2759f133becef9ae4a8635825b88751c8698cb370b401000100fbb75d7ce414edc45510aa87a7547ffb
a65415fca10e83c0936729c2abfb6c2687ee6dde5c6540a835fae6032bbfaf9f8eaeafbcafcf606e94ece103d82ecc82a457
al6692d4023dad4e4fee38fb20d00000000
-----
Waiting for msg3
+++ POST data written to /tmp/wgetpostwnMRxU
---- Copy/Paste Msg4 Below to Client -----
0300000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000
-----
Waiting for a client to connect...
```

Open another terminal and run "run-client"

[illegible]

3.3.6.4 Possible wget Error

Server may invoke wget command to get some files from intel servers. If the server side fails with following error

```
Connecting to api.trustedservices.intel.com (api.trustedservices.intel.com)|40.87.90.88|:443... connected.
ERROR: cannot verify api.trustedservices.intel.com's certificate, issued by 'CN=COMODO RSA Organization
Validation Secure Server CA,O=COMODO CA Limited,L=Salford,ST=Greater Manchester,C=GB':
Unable to locally verify the issuer's authority.
To connect to api.trustedservices.intel.com insecurely, use '--no-check-certificate'.
```

```
then add a line
ca-certificate = /etc/ssl/certs/ca-certificates.crt
```

to /etc/wgetrc file as super user, then test again.

3.3.6.5 BIOS Updating

If BIOS version is outdated, IAS may not succeed. So when you are done with BIOS update, the sgx driver would be required to make and install again.

Update BIOS from:

- <https://downloadcenter.intel.com/download/29987/BIOS-Update-JYGLKCPX->
- <https://downloadcenter.intel.com/download/30069/BIOS-Update-QNCFLX70->

3.3.6.6 Run LocalAttestation

Running SDK code samples in simulation mode

```
source /opt/intel/sgxsdk/environment
cd linux-sgx/SampleCode/LocalAttestation
make SGX_MODE=SIM
cd bin
./app
succeed to load enclaves.
succeed to establish secure channel.
Succeed to exchange secure message...
Succeed to close Session...
```

Running in hardware mode (It works when you have latest BIOS and SGX support is enabled in BIOS)

```
source /opt/intel/sgxsdk/environment
cd linux-sgx/SampleCode/LocalAttestation
make SGX_MODE=HW
cd bin
./app
succeed to load enclaves.
succeed to establish secure channel.
Succeed to exchange secure message...
Succeed to close Session...
```

3.4 Investigation of MbedTLS Configuration file

Mbed TLS should build out-of-the box on a large variety of platforms. However, we may need to adjust a few platform-specific settings or we can customize the set of features that will be built. All this operation can be performed in a single configuration file.

3.4.1 mbedtls configuration file (config.h)

The mbedtls configuration file `config.h` has the following sections

- System Support - Select options depending on platform
- Mbed TLS feature support - Select which features you want to enable for corresponding modules
- Mbed TLS modules - Select modules to build in
- Module configuration options - Set specific options for each module

3.4.1.1 Optee mbedtls config file

Location of the config file in optee environment

optee/mbedtls/include/mbedtls/config.h

Have a look at the source which uses config.h file for reference.

Example source:

optee/mbedtls/include/mbedtls/library/ssl-ciphersuites.c

Some sample configurations can be found in `configs/` directory. In Optee, the contents of configs directory is listed below.

```
$ ls -l optee/mbedtls/configs
total 24
-rw-r--r-- 1 akirat akirat 2852 Feb 17 2021 config-ccm-psk-tls1.2.h
-rw-r--r-- 1 akirat akirat 2102 Feb 17 2021 config-mini-tls1.1.h
-rw-r--r-- 1 akirat akirat 2628 Feb 17 2021 config-no-entropy.h
-rw-r--r-- 1 akirat akirat 3573 Feb 17 2021 config-suite-b.h
-rw-r--r-- 1 akirat akirat 2680 Feb 17 2021 config-thread.h
-rw-r--r-- 1 akirat akirat 1050 Feb 17 2021 README.txt
```

3.4.1.2 ta-ref mbedtls config file

Location of the config file in ta-ref environment

ta-ref/teep-device/libteep/mbedtls/include/mbedtls/config.h

Have a look at the source which uses config.h file for reference.

Example source:

ta-ref/teep-device/libteep/mbedtls/include/mbedtls/library/ssl-
ciphersuites.c

Some sample configurations can be found in `configs/` directory. In ta-ref, the contents of configs directory is listed below.

```
$ ls -l ta-ref/teep-device/libteep/mbedtls/configs
total 24
-rw-r--r-- 1 akirat akirat 2852 Feb 18 2021 config-ccm-psk-tls1.2.h
-rw-r--r-- 1 akirat akirat 2102 Feb 18 2021 config-mini-tls1.1.h
-rw-r--r-- 1 akirat akirat 2628 Feb 18 2021 config-no-entropy.h
-rw-r--r-- 1 akirat akirat 3573 Feb 18 2021 config-suite-b.h
-rw-r--r-- 1 akirat akirat 2680 Feb 18 2021 config-thread.h
-rw-r--r-- 1 akirat akirat 1050 Feb 18 2021 README.txt
```

3.4.1.3 teep-device mbedtls config file

Location of the config file in teep-device environment

teep-device/libteep/mbedtls/include/mbedtls/config.h

Have a look at the source which uses config.h file for reference.

Example source:

teep-device/libteep/mbedtls/include/mbedtls/library/ssl_ciphersuites.c

Some sample configurations can be found in `configs/` directory. In teep-device, the contents of configs directory is listed below.

```
$ ls -l teep-device/libteep/mbedtls/configs
total 24
-rw-r--r-- 1 akirat akirat 2852 Feb 18 2021 config-ccm-psk-tls1.2.h
-rw-r--r-- 1 akirat akirat 2102 Feb 18 2021 config-mini-tls1.1.h
-rw-r--r-- 1 akirat akirat 2628 Feb 18 2021 config-no-entropy.h
-rw-r--r-- 1 akirat akirat 3573 Feb 18 2021 config-suite-b.h
-rw-r--r-- 1 akirat akirat 2680 Feb 18 2021 config-thread.h
-rw-r--r-- 1 akirat akirat 1050 Feb 18 2021 README.txt
```

3.4.2 Supplement Investigation information

It is necessary to edit the following file to select the cryptographic algorithm when using mbedtls in optee.
optee/optee_os/lib/libmbedtls/include/mbedtls_config_kernel.h

In Optee, selection of algorithms can be made in the below file. GCM doesn't seem to be included by default.
optee/optee_os/lib/libmbedtls/include/mbedtls_config_uta.h

In ta-ref, selection of algorithms can be made in the below file. GCM is included by default.

4 Building

4.1 Install Doxygen-1.9.2

This PDF was generated using Doxygen version 1.9.2. To install doxygen-1.9.2 following procedure is necessary.

4.2 Install Required Packages

Install following packages on Ubuntu 18.04

```
sudo apt install doxygen-latex graphviz texlive-full texlive-latex-base latex-cjk-all
```

Above packages required to generate PDF using doxygen.

4.3 Build and Install

```
git clone https://github.com/doxygen/doxygen.git
cd doxygen
mkdir build
cd build
cmake -G "Unix Makefiles" ..
make
sudo make install
```

4.4 ta-ref with Keystone

Make sure Keystone and other dependant sources have been built

4.4.1 Cloning source and building

Install required packages

```
sudo apt-get update
sudo apt-get install -y clang-tools-6.0 libclang-6.0-dev cmake ocaml expect screen sshpass
```

Setup Env

```
export KEYSTONE_DIR=<path to your keystone directory>
export PATH=$PATH:$KEYSTONE_DIR/riscv/bin
```

Clone and Build KEYEDGE

```
GIT_SSL_NO_VERIFY=1 git clone --recursive https://192.168.100.100/rinkai/keyedge.git
cd keyedge
git checkout f9406aba2117147cc54462ede4766e26f028ced9
make
```

Clone and Build KEEDGER8R

```
GIT_SSL_NO_VERIFY=1 git clone --recursive https://192.168.100.100/rinkai/keedger8r.git
cd keedger8r
make
sed -i 's/MAX_EDGE_CALL 10$/MAX_EDGE_CALL 1000/' ${KEYSTONE_DIR}/sdk/lib/edge/include/edge.common.h
make -C ${KEYSTONE_DIR}/sdk/lib clean all
```

Clone the source

```
git clone https://192.168.100.100/rinkai/ta-ref.git
cd ta-ref
git checkout teep-device-tb-slim
git submodule sync --recursive
git submodule update --init --recursive
```

Build

```
export KEYSTONE_DIR=<path to keystone directory>
export KEYSTONE_SDK_DIR=$KEYSTONE_DIR/sdk
export KEYEDGE_DIR=<path to keyedge directory>
export KEEDGER8R_DIR=<path to keedger8r directory>
source env/keystone.sh
make build test-bin MACHINE=HIFIVE TEST_DIR=test.hello
make build test-bin MACHINE=HIFIVE TEST_DIR=test_gp
```

4.4.2 Check ta-ref by running test_gp, test.hello, on QEMU

Copy the test.hello and test_gp programs to QEMU.

4.4.2.1 Launch QEMU Console

```
cd $KEYSTONE_DIR
./scripts/run-qemu.sh
Welcome to Buildroot
```


4.4.2.2 test_hello

Run test_hello

```
cp test_hello/keystone/Enclave/Enclave.eapp.riscv $KEYSTONE_DIR/buildroot_overlay/root/test_hello/
cp test_hello/keystone/Enclave/App.client $KEYSTONE_DIR/buildroot_overlay/root/test_hello/
cp $KEYSTONE_SDK_DIR/rts/eyrie/eyrie-rt $KEYSTONE_DIR/buildroot_overlay/root/test_hello/
insmod keystone-driver.ko
./App.client Enclave.eapp.riscv eyrie-rt
hello world!
```

4.4.2.3 test_gp

Run test_gp

```
cp test_gp/keystone/Enclave/Enclave.eapp.riscv $KEYSTONE_DIR/buildroot_overlay/root/test_gp/
cp test_gp/keystone/Enclave/App.client $KEYSTONE_DIR/buildroot_overlay/root/test_gp/
cp $KEYSTONE_SDK_DIR/rts/eyrie/eyrie-rt $KEYSTONE_DIR/buildroot_overlay/root/test_gp/
insmod keystone-driver.ko
./App.client Enclave.eapp.riscv eyrie-rt
main start
TEE.GenerateRandom(0x000000003FFFFEE0, 16): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@random: 5ea8741bd8a3b298cf53d214eca693fb
TEE.GetREETime(): start
@[SE] gettimeofday 77 sec 865873 usec -> 0
@GP REE time 77 sec 865 millis
TEE.GetSystemTime(): start
@GP System time 100063195 sec 609 millis
TEE.CreatePersistentObject(): start
@[SE] open file FileOne flags 241 -> 3 (0)
TEE.WriteObjectData(): start
@[SE] write desc 3 buf 480d0 len 256-> 256
TEE.CloseObject(): start
@[SE] close desc 3 -> 0
TEE.OpenPersistentObject(): start
@[SE] open file FileOne flags 0 -> 3 (0)
TEE.ReadObjectData(): start
@[SE] read desc 3 buf fff41664 len 256-> 256
TEE.CloseObject(): start
@[SE] close desc 3 -> 0
256 bytes read: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20212223242526272829
2a2b2c2d2e2f303132333435363738393a3b3c3d3f
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
hash: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(0x000000003FFFFD88, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AllocateOperation(): start
TEE.GenerateRandom(0x000000003FFFFED0, 16): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
@cipher: e94431cd22a6029185d0dbb1a17b5d62843bfeef25591583d2d668ec6fed1c692f88ce4754d690c346c8d9f2726
630e0386abf4e45699a2ca2b34b344eaa454bc489c
TEE.AllocateOperation(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f
verify ok
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(0x000000003FFFC68, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AllocateOperation(): start
TEE.GenerateRandom(0x000000003FFFE8, 16): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AEInit(): start
TEE.AEEncryptFinal(): start
TEE.FreeOperation(): start
```

```

@cipher: c23e9ce04589e80a66debe23a788ae5393bdcd8e875e87e1bcf2b2d998f6418ccc6ee4ab112fdbfc5175868691e
fb40781a318ff439d30b49cc9f726886ad42d5be15
@tag: a551f999317b3fbd1eea7b622ce2caee
TEE.AllocateOperation(): start
TEE.AEInit(): start
TEE.AEDecryptFinal(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
@digest: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE.AllocateOperation(): start
TEE.AllocateTransientObject(): start
TEE.InitValueAttribute(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(0x000000003FFFE28, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AsymmetricSignDigest(): start
TEE.FreeOperation(): start
@signature: d6e6b6e54db8b6a62fc1927886938bead27f4813f19ce77182e3016b5426bcad067ca98cd75f9dfddafe9eb0
655c48df992d3ad674db69d831f26ae63caf1405
TEE.AllocateOperation(): start
TEE.AsymmetricVerifyDigest(): start
TEE.FreeOperation(): start
@@TEE.FreeOperation:
TEE.FreeTransientObject(): start
verify ok
main end

```

4.5 ta-ref with OPTEE

Make sure optee.3.9.0.rpi3 has been built already.

4.5.1 Cloning source and building

Clone the source

```

git clone https://192.168.100.100/rinkai/ta-ref.git
cd ta-ref
git checkout teep-device-tb-slim
git submodule sync --recursive
git submodule update --init --recursive

```

Build

```

export OPTEE_DIR=<path to optee.3.9.0.rpi3>
source env/optee.rpi3.sh
make build test-bin MACHINE=RPI3 TEST_DIR=test_hello
make build test-bin MACHINE=RPI3 TEST_DIR=test_gp

```

4.5.2 Check ta-ref by running test_gp, test_hello, on QEMU

Copy the test_hello and test_gp programs to QEMU buildroot directory

```

mkdir -p optee.3.9.0.qemu/out-br/target/home/gitlab/out/{test_hello,test_gp}
cp ta-ref/test_hello/optee/App/optee_ref.ta optee.3.9.0.qemu/out-br/target/home/gitlab/out/test_hello/
cp ta-ref/test_hello/optee/Enclave/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
  optee.3.9.0.qemu/out-br/target/home/gitlab/out/test_hello/
cp ta-ref/test_gp/optee/App/optee_ref.ta optee.3.9.0.qemu/out-br/target/home/gitlab/out/test_gp/
cp ta-ref/test_gp/optee/Enclave/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
  optee.3.9.0.qemu/out-br/target/home/gitlab/out/test_gp/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
cp ./test_gp/optee/Enclave/Enclave.nm /TEE/demo/rpi3/optee.3.9.0.qemu/out-br/target/home/gitlab/out/test_gp/

```

4.5.2.1 test_hello

Run test_hello

```
cp /home/gitlab/out/test_hello/
cp a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta /home/gitlab/out/
ln -s /home/gitlab/out/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
    /lib64/optee.armtz/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
./optee_ref.ta
--- enclave log start---
ecall.ta.main() start
hello world!
ecall.ta.main() end
--- enclave log end---
```

If executed successfully, you see above messages

4.5.2.2 test_gp

Run test_gp

```
cd /home/gitlab/out/test_gp/
cp a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta /home/gitlab/out/
ln -s /home/gitlab/out/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
    /lib64/optee.armtz/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
./optee_ref.ta
start TEEC.InvokeCommand
--- enclave log start---
ecall.ta.main() start
@random: fe0c7d3ee9b9bd5e63b8a0cce29af7eb
@GP REE time 1612156259 sec 390 millis
@GP System time 249187 sec 954 millis
256 bytes read: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20212223242526272829
2a2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b
5c5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d
8e8f909192939495969798999a9b9c9d9e9f9a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbef
c0c1c2c3c4c5c6c7c8c9cacbcccdcecf0d1d2d3d4d5d6d7d8d9dadbdcddeedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1
f2f3f4f5f6f7f8f9fafbfcfdfeff
verify ok
hash: 40aff2e9d2d8922e47afd4648e6967497158785fbd1da870e7110266bf944880
@cipher: 30a558176172c53be4a2ac320776de105da79c29726879fe67d06b629f065731285f8a90f8a521ce34ecee51e1
5e928d157ea10d149bb687dd78be79469c28696506283edcda527fcd86f6a47e852bbc3488df3fc651b46b034faf4ab5f12f
51a285478ea01e58d40e8177d415be243df93b23cdf889feb91fa3be8906fe190d836fe1168aed0473406be1054dd88a381
ef25381d920ea3780ba74fb1cfe1434cbd168de8386dce2e2b92ee0fc432f3c0514f462cbeaf96753b174a4a673f323e671
61272fe932ead4bc95770fcc130dd5877b521d6a79f961eeadd1680042f69257ccf9368927aa170176af8ac211dd22161997
7224837232dad970220f4
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9f9a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbefc0
c1c2c3c4c5c6c7c8c9cacbcccdcecf0d1d2d3d4d5d6d7d8d9dadbdcddeedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2
f3f4f5f6f7f8f9fafbfcfdfeff
verify ok
@cipher: ff409d8fe203bf0d81de36832b86c702f07edd343f408d3a2fb5ab347b4f72b10031efff0c17b7e0bc56c3f2f95
f53c0d731ed87eb3e1187b6714a25cfc65082284682b44450941654e7edc99af0f7b037c3ba9ea731036070aa9496e34cfeb
db6845e8aa9955416ba227970d3dd1f8207b5743e1490a7f5fd78d81fce0a24576de06a2f528d49c5b11e79a5cab015806ba
d73f118e205a3645a95b2b330ff9da12e00c693e7ee8cfd04eb0f08c3c657c4fa0ae384ed2d5ab1e15ffcc835c3e4cc116cd
1049611f896cf445ab36dc8b393a6fe75d20d45b2273a5d8c2d3b935e3f22bc82b24c952812d66a902155d288d5f26ac6722
fe72498bd72ea523c914c
@tag: 9b357baf76d2632fa7d16231640d6324
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9f9a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbefc0
c1c2c3c4c5c6c7c8c9cacbcccdcecf0d1d2d3d4d5d6d7d8d9dadbdcddeedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2
f3f4f5f6f7f8f9fafbfcfdfeff
verify ok
@digest: 40aff2e9d2d8922e47afd4648e6967497158785fbd1da870e7110266bf944880
@signature: 719fa9898f3423b754675b835268f9b2368b77a429eeabf7369d60d135dee08158c3902fd2ed3c1bf17cb34e
76f2ba25da915fa3970c757962f7533c8d8bad7d
@@TEELFreeOperation:
verify ok
ecall.ta.main() end
--- enclave log end---
res = TEEC.SUCCESS; TEEC.InvokeCommand succeeded!
```

If executed successfully, you see above messages

4.6 ta-ref with SGX

Build ta-ref for Intel SGX platforms

4.6.1 Cloning source and building

Clone the source

```
git clone https://192.168.100.100/rinkai/ta-ref.git
cd ta-ref
git checkout teep-device-tb-slim
git submodule sync --recursive
git submodule update --init --recursive
```

Build

```
source /opt/intel/sgxsdk/environment
source env/sgx.x64.sh
make build test-bin MACHINE=NUC TEST_DIR=test.hello
make build test-bin MACHINE=NUC TEST_DIR=test.gp
```

4.6.2 Check ta-ref by running test_gp, test.hello, simulation mode on any pc

Copy the ta-ref's test.hello & test_gp executables to test directory

4.6.2.1 test.hello

Run test.hello

```
cp test.hello/sgx/Enclave/enclave.signed.so <test directory>
cp test.hello/sgx/App/sgx.app <test directory>
<test directory>/sgx.app
hello world!
Info: Enclave successfully returned.
```

4.6.2.2 test_gp

Run test_gp

```
cp test_gp/sgx/Enclave/enclave.signed.so <test directory>
cp test_gp/sgx/App/sgx.app <test directory>
<test directory>/sgx.app
main start
TEE.GenerateRandom(): start
@random: f35c1d1e4bbf6641c5511c9dc5aaf638
TEE.GetREETime(): start
request to get unix time 1612257364, 199
@GP REE time 1612257364 sec 199 millis
TEE.GetSystemTime(): start
@GP System time 727941859 sec 984 millis
TEE.CreatePersistentObject(): start
request to open FileOne flags 241 -> 3
TEE.WriteObjectData(): start
request to write 256 bytes to descriptor 3
TEE.CloseObject(): start
request to close descriptor 3
TEE.OpenPersistentObject(): start
request to open FileOne flags 0 -> 3
TEE.ReadObjectData(): start
request to read 256 bytes from descriptor 3
TEE.CloseObject(): start
request to close descriptor 3
256 bytes read: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20212223242526272829
2a2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b
5c5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d
8e8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbef
c0c1c2c3c4c5c6c7c8c9cacbcccdcecf0d1d2d3d4d5d6d7d8d9dadbdcddeedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1
f2f3f4f5f6f7f8f9fafbfcfdfeff
```

```

verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
hash: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(): start
TEE.AllocateOperation(): start
TEE.GenerateRandom(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
@cipher: 7427bff21e729a824a239e25332ebd455d18fa6aec1ec6618b77c252f768e0a9345608b0135727568867ce5b0fa
c872f6647787861b88220840281f3944eea456a2769081e6598079b52edc541e2201ffd2e96a6c3e485be25a0ce4f5c07544
aa0c67b3e34bd069b293843daf66db51b751b3c09f2a9c6912c22a0602c8ecbd0effd4698081660e218f6f0c1249e3691a33
e91836953953513040eb29ce709efe50f96e67f07d6a1b00f08beacebc5950f9744b0049cb76ec5ba17a49d7270b60034c47
23bb79dc61d465062b0394e8d93f98c2391ee2b02b7b537b375e0e1cc5eeb8eb2e62df839048db0f1fdbdd1b7f5c6ef2faa1
a5b305ef045936c9146f8
TEE.AllocateOperation(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbbfbfc0
c1c2c3c4c5c6c7c8c9cacbcccdcecfcd0d1d2d3d4d5d6d7d8d9dadbdcddeedfe0e1e2e3e4e5e6e7e8e9eaebeceedeefeff0f1f2
f3f4f5f6f7f8f9fafbfcfdfefff
verify ok
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(): start
TEE.AllocateOperation(): start
TEE.GenerateRandom(): start
TEE.AEInit(): start
TEE.AEEncryptFinal(): start
TEE.FreeOperation(): start
@cipher: e33f34122c80b9a10002725e4e21542256da7c7cd3f6dd1b62b71cf8308f9e4a0daa50b29880a8f76707c4ed432
549c4da9e68e7930189d2127fdd7aa2379106090814b5deed9a9e161ef0886da03a2a94c3fb9e0faadfd1ce8bb09fb5388bb
23a042944fbb269d486aa4f21a91a41968184122520dfc308850059efce660a52adb17361bd52f570bfba05cccad32ffa9ea
c94914725ded073355f28eb3dc30d60f00cfd2de76c3a05df8bef32f302bb4d14b493a3a90b1dee4eba64e625695c4d58ec4
febff8436d62e4cac82fcb00e60c8138af7176995a742b08a572f64e539e9f9850a9f6f33907a829108ca6540332aab53f3f
6a4fd2c3de35c5556a427
@tag: 4c920ce2aef079e468ab24e25730d9d2
TEE.AllocateOperation(): start
TEE.AEInit(): start
TEE.AEDecryptFinal(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbbfbfc0
c1c2c3c4c5c6c7c8c9cacbcccdcecfcd0d1d2d3d4d5d6d7d8d9dadbdcddeedfe0e1e2e3e4e5e6e7e8e9eaebeceedeefeff0f1f2
f3f4f5f6f7f8f9fafbfcfdfefff
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
@digest: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE.AllocateOperation(): start
TEE.AllocateTransientObject(): start
TEE.InitValueAttribute(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(): start
TEE.AsymmetricSignDigest(): start
TEE.FreeOperation(): start
@signature: 100b392ce043e9b8dc703088f505dd3083ec47bfcb8d59d968a66b54e80464d684d56dc9c44336f08fd96309
79863a2d8fbb7cd672a819ef609357e9ac6a3d80e
TEE.AllocateOperation(): start
TEE.AsymmetricVerifyDigest(): start
TEE.FreeOperation(): start
@@TEE.FreeOperation:
TEE.FreeTransientObject(): start
verify ok
main end
Info: Enclave successfully returned.

```

5 Running on Dev Boards

5.1 Keystone, Unleased

Make sure Keystone and other dependant sources have been built

5.1.1 Preparation of rootfs on SD Card

Build a modified gdisk which can handle the sifive specific partition types.

Prerequisites: libncursesw5-dev, libpopt-dev

```
$ cd ..
$ sudo apt install libncursesw5-dev lib64ncurses5-dev uuid-dev libpopt-dev build-essential
$ git clone https://192.168.100.100/rinkai/gptfdisk.git
$ cd gptfdisk
$ git checkout -b risc-v-sd 3d6a15873f582803aa8ad3288b3e32d3daff9fde
$ make
```

5.1.1.1 Create SD-card partition manually

```
sudo ./gdisk /dev/mmcblk0
GPT fdisk (gdisk) version 1.0.4
Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present
Found valid GPT with protective MBR; using GPT.
Command (? for help): n
Partition number (1-128, default 1): 1
First sector (34-15523806, default = 2048) or {+-}size{KMGT}:
Last sector (2048-15523806, default = 15523806) or {+-}size{KMGT}: 67583
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300): 5202
Changed type of partition to 'SiFive bare-metal (or stage 2 loader)'
Command (? for help): n
Partition number (2-128, default 2): 4
First sector (34-15523806, default = 67584) or {+-}size{KMGT}:
Last sector (67584-15523806, default = 15523806) or {+-}size{KMGT}: 67839
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300): 5201
Changed type of partition to 'SiFive FSBL (first-stage bootloader)'
Command (? for help): n
Partition number (2-128, default 2):
First sector (34-15523806, default = 69632) or {+-}size{KMGT}: 264192
Last sector (264192-15523806, default = 15523806) or {+-}size{KMGT}:
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300): 8300
Changed type of partition to 'Linux filesystem'
Command (? for help): p
Disk /dev/mmcblk0: 15523840 sectors, 7.4 GiB
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 11A0F8F6-D5DE-4993-8C0D-D543DFBA17AD
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 15523806
Partitions will be aligned on 2048-sector boundaries
Total free space is 198366 sectors (96.9 MiB)
Number  Start (sector)    End (sector)  Size      Code  Name
   1            2048             67583     32.0 MiB   5202   SiFive bare-metal (...)
   2         264192        15523806     7.3 GiB   8300   Linux filesystem
   4         67584         67839      128.0 KiB  5201   SiFive FSBL (first-...
Command (? for help): i
Partition number (1-4): 4
Partition GUID code: 5B193300-FC78-40CD-8002-E86C45580B47 (SiFive FSBL (first-stage bootloader))
Partition unique GUID: FC1FBC7C-EC94-4B0A-9DAF-0ED85452B885
First sector: 67584 (at 33.0 MiB)
Last sector: 67839 (at 33.1 MiB)
Partition size: 256 sectors (128.0 KiB)
Attribute flags: 0000000000000000
Partition name: 'SiFive FSBL (first-stage bootloader)'
Command (? for help): i
```

```

Partition number (1-4): 1
Partition GUID code: 2E54B353-1271-4842-806F-E436D6AF6985 (SiFive bare-metal (or stage 2 loader))
Partition unique GUID: 2FFF07EF-E44A-4278-A16D-C29697C6653D
First sector: 2048 (at 1024.0 KiB)
Last sector: 67583 (at 33.0 MiB)
Partition size: 65536 sectors (32.0 MiB)
Attribute flags: 0000000000000000
Partition name: 'SiFive bare-metal (or stage 2 loader)'
Command (? for help): wq
Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!
Do you want to proceed? (Y/N): y
OK; writing new GUID partition table (GPT) to /dev/mmcblk1.
Warning: The kernel is still using the old partition table.
The new table will be used at the next reboot or after you
run partprobe(8) or kpartx(8)
The operation has completed successfully.

```

5.1.1.2 Write boot and rootfs files into SD-card

Build FSBL for hifive-Unleashed board

```

$ git clone https://github.com/keystone-enclave/freedom-u540-c000-bootloader.git
$ cd freedom-u540-c000-bootloader
$ git checkout -b dev-unleashed bbfcc288fb438312af51adef420aa444a0833452
$# Make sure riscv64 compiler set to PATH (export PATH=$KEYSTONE_DIR/riscv/bin:$PATH)
$ make

```

Writing fsbl.bin and bbl.bin

```

sudo dd if=freedom-u540-c000-bootloader/fsbl.bin of=/dev/mmcblk0p4 bs=4096 conv=fsync
sudo dd if=$KEYSTONE_DIR/hifive-work/bbl.bin of=/dev/mmcblk0p1 bs=4096 conv=fsync

```

Once files written, insert the SD-card into unleashed

5.1.2 Copying binaries of test_hello and test_gp

```

sudo mount /dev/mmcblk0p1 /media/rootfs/
sudo mkdir /media/rootfs/root/{test_hello,test_gp}
Copy test_hello
sudo cp ta-ref/test_hello/keystone/Enclave/Enclave.eapp.riscv /media/rootfs/root/test_hello/
sudo cp ta-ref/test_hello/keystone/Enclave/App.client /media/rootfs/root/test_hello/
sudo cp $KEYSTONE.SDK_DIR/rts/eyrie/eyrie-rt /media/rootfs/root/test_hello/
Copy test_gp
sudo cp ta-ref/test_gp/keystone/Enclave/Enclave.eapp.riscv /media/rootfs/root/test_gp/
sudo cp ta-ref/test_gp/keystone/Enclave/App.client /media/rootfs/root/test_gp/
sudo cp $KEYSTONE.SDK_DIR/rts/eyrie/eyrie-rt /media/rootfs/root/test_gp/

```

Now, we are ready to test on unleashed board.

5.1.3 Check test_hello and test_gp on Unleashed

1. Insert SD-card into unleashed board
2. Boot Hifive-Unleashed board
3. Connect Unleashed board with your development machine over USB-Serial cable (/dev/ttyUSB1)
4. Checking on Unleashed

```

Login to serial console with user=root, passwd=sifive
buildroot login: root
Password:
$

```

test_hello:

```

insmod keystone-driver.ko
./App.client Enclave.eapp.riscv eyrie-rt
hello world!

```

test_gp:

```

insmod keystone-driver.ko
./App.client Enclave.eapp.riscv eyrie-rt

```

```

main start
TEE.GenerateRandom(0x000000003FFFFEE0, 16): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@random: 5ea8741bd8a3b298cf53d214eca693fb
TEE.GetREETime(): start
@[SE] gettimeofday 77 sec 865873 usec -> 0
@GP REE time 77 sec 865 millis
TEE.GetSystemTime(): start
@GP System time 100063195 sec 609 millis
TEE.CreatePersistentObject(): start
@[SE] open file FileOne flags 241 -> 3 (0)
TEE.WriteObjectData(): start
@[SE] write desc 3 buf 480d0 len 256-> 256
TEE.CloseObject(): start
@[SE] close desc 3 -> 0
TEE.OpenPersistentObject(): start
@[SE] open file FileOne flags 0 -> 3 (0)
TEE.ReadObjectData(): start
@[SE] read desc 3 buf fff41664 len 256-> 256
TEE.CloseObject(): start
@[SE] close desc 3 -> 0
256 bytes read: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20212223242526272829
2a2b2c2d2e2f303132333435363738393a3b3c3d3f
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
hash: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(0x000000003FFFFD88, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AllocateOperation(): start
TEE.GenerateRandom(0x000000003FFFFED0, 16): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
@cipher: e94431cd22a6029185d0dbb1a17b5d62843bfeef25591583d2d668ec6fed1c692f88ce4754d690c346c8d9f2726
630e0386abf4e45699a2ca2b34b344eaa454bc489c
TEE.AllocateOperation(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f
verify ok
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(0x000000003FFFFC68, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AllocateOperation(): start
TEE.GenerateRandom(0x000000003FFFFEC8, 16): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AEInit(): start
TEE.AEEncryptFinal(): start
TEE.FreeOperation(): start
@cipher: c23e9ce04589e80a66debe23a788ae5393bdcd8e875e87e1bcf2b2d998f6418ccc6ee4ab112fdbfc5175868691e
fb40781a318ff439d30b49cc9f726886ad42d5be15
@tag: a551f999317b3fbd1eea7b622ce2caee
TEE.AllocateOperation(): start
TEE.AEInit(): start
TEE.AEDecryptFinal(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
@digest: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE.AllocateOperation(): start
TEE.AllocateTransientObject(): start
TEE.InitValueAttribute(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(0x000000003FFFFE28, 32): start
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
@[SE] getrandom buf fff41844 len 16 flags 0 -> 16
TEE.AsymmetricSignDigest(): start
TEE.FreeOperation(): start
@signature: d6e6b6e54db8b6a2fc1927886938bead27f4813f19ce77182e3016b5426bcad067ca98cd75f9dfddafe9eb0

```



```
655c48df992d3ad674db69d831f26ae63caf1405
TEE.AllocateOperation(): start
TEE.AsymmetricVerifyDigest(): start
TEE.FreeOperation(): start
@@TEE.FreeOperation:
TEE.FreeTransientObject(): start
verify ok
main end
```

Test is successful.

5.2 OPTEE, RPI3

Make sure OPTEE v3.9.0 and other dependant sources have been built

5.2.1 Preparation of rootfs on SD Card

Use following examples to create partitions of boot and roots on SD-card

```
make img-help
$ fdisk /dev/sdx      # where sdx is the name of your sd-card
  > p                 # prints partition table
  > d                 # repeat until all partitions are deleted
  > n                 # create a new partition
  > p                 # create primary
  > 1                 # make it the first partition
  > <enter>            # use the default sector
  > +32M              # create a boot partition with 32MB of space
  > n                 # create rootfs partition
  > p
  > 2
  > <enter>
  > <enter>           # fill the remaining disk, adjust size to fit your needs
  > t                 # change partition type
  > 1                 # select first partition
  > e                 # use type 'e' (FAT16)
  > a                 # make partition bootable
  > 1                 # select first partition
  > p                 # double check everything looks right
  > w                 # write partition table to disk.
```

Usually your SD-card detected as /dev/mmcblk0. After partition it looks like below BOOT partition = /dev/mmcblk0p1 rootfs partition = /dev/mmcblk0p2

Write boot file

```
$ mkfs.vfat -F16 -n BOOT /dev/mmcblk0p1
$ mkdir -p /media/boot
$ sudo mount /dev/mmcblk0p1 /media/boot
$ cd /media
$ gunzip -cd optee-3.9.0-rpi3/out-br/images/rootfs.cpio.gz | sudo cpio -idmv "boot/*"
$ umount boot
```

Write rootfs

```
$ mkfs.ext4 -L rootfs /dev/mmcblk0p2
$ mkdir -p /media/rootfs
$ sudo mount /dev/mmcblk0p2 /media/rootfs
$ cd rootfs
$ gunzip -cd <your-base-dir>/optee-3.9.0-rpi3/build/./out-br/images/rootfs.cpio.gz | sudo cpio -idmv
$ rm -rf /media/rootfs/boot/*
$ cd .. && sudo umount rootfs
```

If you use CI from AIST, download rpi3.sdimage as follows

```
$ wget http://192.168.100.100:2000/optee-rpi3.sdimage.tar.xz
$ tar xf optee-rpi3.sdimage.tar.xz
$ dd if=rpi3.sdimage.bin of=/dev/mmcblk0p2 conv=fsync bs=4096
```

Now SD-card is ready to boot RPI3.

5.2.2 Copying binaries of test_hello and test_gp to rootfs partition

Copying test_hello & test_gp

```
$ sudo mount /dev/mmcblk0p2 /media/rootfs
$ sudo mkdir -p /media/rootfs/home/gitlab/out/{test_hello,test_gp}
$ sudo cp ta-ref/test_hello/optee/App/optee_ref.ta /media/rootfs/home/gitlab/out/test_hello/
$ sudo cp ta-ref/test_hello/optee/Enclave/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
/media/rootfs/home/gitlab/out/test_hello/
$ sudo cp ta-ref/test_gp/optee/App/optee_ref.ta /media/rootfs/home/gitlab/out/test_gp/
$ sudo cp ta-ref/test_gp/optee/Enclave/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
/media/rootfs/home/gitlab/out/test_gp/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
$ sudo cp ta-ref/test_gp/optee/Enclave/Enclave.nm /media/rootfs/home/gitlab/out/test_gp/
```

5.2.3 Check test_hello and test_gp

1. Insert SD-card into RPI3 board, then power-on
2. Connect RPI3 board Serial console to your laptop (/dev/ttyUSB0 over minicom)
3. Checking on RPI3

Login to Serial console and enter "root" as username

```
buildroot login: root
Password:
$
```

test_hello:

```
cp /home/gitlab/out/test_hello/
cp a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta /home/gitlab/out/
ln -s /home/gitlab/out/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
/lib64/optee.armtz/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
./optee_ref.ta
--- enclave log start---
ecall.ta_main() start
hello world!
ecall.ta_main() end
--- enclave log end---
```

If executed successfully, you see above messages

test_gp:

```
cd /home/gitlab/out/test_gp/
cp a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta /home/gitlab/out/
ln -s /home/gitlab/out/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
/lib64/optee.armtz/a6f77c1e-96fe-4a0e-9e74-262582a4c8f1.ta
./optee_ref.ta
start TEEC.InvokeCommand
--- enclave log start---
ecall.ta_main() start
@random: fe0c7d3eefb9bd5e63b8a0cce29af7eb
@GP REE time 1612156259 sec 390 millis
@GP System time 249187 sec 954 millis
256 bytes read: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20212223242526272829
2a2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b
5c5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d
8e8f909192939495969798999a9b9c9d9e9f9a9b9c9d9e9f9a9b9c9d9e9f9a9b9c9d9e9f9a9b9c9d9e9f9a9b9c9d9e9f9a
c0c1c2c3c4c5c6c7c8c9cacbcccdcecfcd0d1d2d3d4d5d6d7d8d9daddbdcdddedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f
f2f3f4f5f6f7f8f9fafbfcfdfefff
verify ok
hash: 40aff2e9d2d8922e47afd4648e6967497158785fbd1da870e7110266bf944880
@cipher: 30a558176172c53be4a2ac320776de105da79c29726879fe67d06b629f065731285f8a90f8a521ce34ecee51e1
5e928d157ea10d149bb687dd78be79469c28696506283edcda527fcd86f6a47e852bbc3488df3fc651b46b034faf4ab5f12f
51a285478ea01e58d40e8177d415be243df93b23cdf889feb91fa3be8906fe190d836fe61168aed0473406be1054dd88a381
ef25381d920ea3780ba74fb1cfe1434cbd168de8386dcc2e2b92eee0fc432f3c0514f462cbeaf96753b174a4a673f323e671
61272fe932ead4bc95770fcc130dd5877b521d6a79f961eeadd1680042f69257ccf9368927aa170176af8ac211dd22161997
7224837232dad970220f4
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9f9a9b9c9d9e9f9a9b9c9d9e9f9a9b9c9d9e9f9a9b9c9d9e9f9a9b9c9d9e9f9a
c1c2c3c4c5c6c7c8c9cacbcccdcecfcd0d1d2d3d4d5d6d7d8d9daddbdcdddedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2
f3f4f5f6f7f8f9fafbfcfdfefff
```

```

verify ok
@cipher: ff409d8fe203bf0d81de36832b86c702f07edd343f408d3a2fb5ab347b4f72b10031efff0c17b7e0bc56c3f2f95
f53c0d731ed87eb3e1187b6714a25cfc65082284682b44450941654e7edc99af0f7b037c3ba9ea731036070aa9496e34cfeb
db6845e8aa9955416ba227970d3dd1f8207b5743e1490a7f5fd78d81fce0a24576de06a2f528d49c5b11e79a5cab015806ba
d73f118e205a3645a95b2b330ffd9da12e00c693e7ee8cfd04eb0f08c3c657c4fa0ae384ed2d5ab1e15ffc835c3e4cc116cd
1049611f896cf445ab36dc8b393a6fe75d20d45b2273a5d8c2d3b935e3f22bc82b24c952812d66a902155d288d5f26ac6722
fe72498bd72ea523c914c
@tag: 9b357baf76d2632fa7d16231640d6324
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbbfc0
c1c2c3c4c5c6c7c8c9cacbcccdcccfcd0d1d2d3d4d5d6d7d8d9dadbdcdddedfe0e1e2e3e4e5e6e7e8e9eaebeceeeeff0f1f2
f3f4f5f6f7f8f9fafbfcfdfeff
verify ok
@digest: 40aff2e9d2d8922e47afd4648e6967497158785fbd1da870e7110266bf944880
@signature: 719fa9898f3423b754675b835268f9b2368b77a429eeabf7369d60d135dee08158c3902fd2ed3c1bf17cb34e
76f2ba25da915fa3970c757962f7533c8d8bad7d
@@TEE.FreeOperation:
verify ok
ecall.ta_main() end
--- enclave log end---
res = TEEC_SUCCESS; TEEC_InvokeCommand succeeded!

```

If executed successfully, you see above messages

5.3 SGX, NUC

Make sure SGX SDK, sgx driver and other dependant sources have been built and installed on NUC machine

5.3.1 Copying binaries of test_hello and test_gp to NUC machine

Login to NUC machine over SSH (Assuming that SSH enabled on NIC machine). Assuming that ta-ref was natively built on NUC machine at ~/ta-ref

```

ssh <ssh-user>@<IP-Address> 'mkdir -p ~/test_hello,test_gp'
scp ta-ref/test_hello/sgx/Enclave/enclave.signed.so <ssh-user>@<IP-Address>:~/test_hello
scp ta-ref/test_hello/sgx/App/sgx.app <ssh-user>@<IP-Address>:~/test_hello
scp ta-ref/test_gp/sgx/Enclave/enclave.signed.so <ssh-user>@<IP-Address>:~/test_gp
scp ta-ref/test_gp/sgx/App/sgx.app <ssh-user>@<IP-Address>:~/test_gp

```

Now can login to NUC machine for further testing.

5.3.2 Check test_hello and test_gp

Checking test_hello

```

cd ~/test_hello
./sgx.app
hello world!
Info: Enclave successfully returned.

```

Checking test_gp

```

cd ~/test_gp
./sgx.app
main start
TEE.GenerateRandom(): start
@random: f35c1d1e4bbf6641c5511c9dc5aaf638
TEE.GetREETime(): start
request to get unix time 1612257364, 199
@GP REE time 1612257364 sec 199 millis
TEE.GetSystemTime(): start
@GP System time 727941859 sec 984 millis
TEE.CreatePersistentObject(): start
request to open FileOne flags 241 -> 3
TEE.WriteObjectData(): start
request to write 256 bytes to descriptor 3
TEE.CloseObject(): start
request to close descriptor 3
TEE.OpenPersistentObject(): start
request to open FileOne flags 0 -> 3
TEE.ReadObjectData(): start

```

```

request to read 256 bytes from descriptor 3
TEE.CloseObject(): start
request to close descriptor 3
256 bytes read: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20212223242526272829
2a2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b
5c5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d
8e8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbefc
c0c1c2c3c4c5c6c7c8c9cacbcccdcecfcd0d1d2d3d4d5d6d7d8d9dadbdcddeedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f
f2f3f4f5f6f7f8f9fafbfcfdfefff
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
hash: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(): start
TEE.AllocateOperation(): start
TEE.GenerateRandom(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
@cipher: 7427bfff21e729a824a239e25332ebd455d18fa6aec1ec6618b77c252f768e0a9345608b0135727568867ce5b0fa
c872f6647787861b88220840281f3944eea456a2769081e6598079b52edc541e2201ffd2e96a6c3e485be25a0ce4f5c07544
aa0c67b3e34bd069b293843daf66db51b751b3c09f2a9c6912c22a0602c8ecbd0effd4698081660e218f6f0c1249e3691a33
e91836953953513040eb29ce709efe50f96e67f07d6a1b00f08beacebc5950f9744b0049cb76ec5ba17a49d7270b60034c47
23bb79dc61d465062b0394e8d93f98c2391ee2b02b7b537b375e0e1cc5eeb8eb2e62df839048db0f1fdbdd1b7f5c6ef2faa1
a5b305ef045936c9146f8
TEE.AllocateOperation(): start
TEE.CipherInit(): start
TEE.CipherUpdate(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbefc0
c1c2c3c4c5c6c7c8c9cacbcccdcecfcd0d1d2d3d4d5d6d7d8d9dadbdcddeedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2
f3f4f5f6f7f8f9fafbfcfdfefff
verify ok
TEE.AllocateTransientObject(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(): start
TEE.AllocateOperation(): start
TEE.GenerateRandom(): start
TEE.AEInit(): start
TEE.AEEncryptFinal(): start
TEE.FreeOperation(): start
@cipher: e33f34122c80b9a10002725e4e21542256da7c7cd3f6dd1b62b71cf8308f9e4a0daa50b29880a8f76707c4ed432
549c4da9e68e7930189d2127fdd7aa2379106090814b5deed9a9e161ef0886da03a2a94c3fb9e0faadfd1ce8bb09fb5388bb
23a042944fbb269d486aa4f21a91a41968184122520dfc308850059efce660a52adb17361bd52f570bfba05cccad32ffa9ea
c94914725ded073355f28eb3dc30d60f00cfd2de76c3a05df8bef32f302bb4d14b493a3a90b1dee4eba64e625695c4d58ec4
feb8436d62e4cac82fcb00e60c8138af7176995a742b08a572f64e539e9f9850a9f6f33907a829108ca6540332aab53f3f
6a4fd2c3de35c5556a427
@tag: 4c920ce2aef079e468ab24e25730d9d2
TEE.AllocateOperation(): start
TEE.AEInit(): start
TEE.AEDecryptFinal(): start
TEE.FreeOperation(): start
TEE.FreeTransientObject(): start
decrypted to: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a
2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c
5d5e5f606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f808182838485868788898a8b8c8d8e
8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbefc0
c1c2c3c4c5c6c7c8c9cacbcccdcecfcd0d1d2d3d4d5d6d7d8d9dadbdcddeedfe0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2
f3f4f5f6f7f8f9fafbfcfdfefff
verify ok
TEE.AllocateOperation(): start
TEE.FreeOperation(): start
TEE.DigestDoFinal(): start
TEE.FreeOperation(): start
@digest: 9b04c091da96b997afb8f2585d608aeb9c4a904f7d52c8f28c7e4d2dd9fba5f
TEE.AllocateOperation(): start
TEE.AllocateTransientObject(): start
TEE.InitValueAttribute(): start
TEE.GenerateKey(): start
TEE.GenerateRandom(): start
TEE.AsymmetricSignDigest(): start
TEE.FreeOperation(): start
@signature: 100b392ce043e9b8dc703088f505dd3083ec47bfcb8d59d968a66b54e80464d684d56dc9c44336f08fd96309
79863a2d8fbb7cd672a819ef609357e9ac6a3d80e
TEE.AllocateOperation(): start
TEE.AsymmetricVerifyDigest(): start
TEE.FreeOperation(): start
@@TEE.FreeOperation:
TEE.FreeTransientObject(): start

```

```
verify ok  
main end  
Info: Enclave successfully returned.
```

6 API Compare With Full-Set of GP API

6.1 GP API

API Functions by Category

APIs supported by both GP and AIST-GP are in Blue

API list from TEE Internal Core API Specification documentation, GlobalPlatform Technology

Asymmetric	TEE_FreeOperation
TEE_AsymmetricDecrypt	TEE_GetOperationInfo
TEE_AsymmetricEncrypt	TEE_GetOperationInfoMultiple
TEE_AsymmetricSignDigest	TEE_IsAlgorithmSupported
TEE_AsymmetricVerifyDigest	TEE_ResetOperation
Authenticated Encryption	TEE_SetOperationKey
TEE_AEDecryptFinal	TEE_SetOperationKey2
TEE_AEEncryptFinal	Initialization
TEE_AEInit	TEE_BigIntInit
TEE_AEUpdate	TEE_BigIntInitFMM
TEE_AEUpdateAAD	TEE_BigIntInitFMMContext
Basic Arithmetic	Internal Client API
TEE_BigIntAdd	TEE_CloseTASession
TEE_BigIntDiv	TEE_InvokeTACommand
TEE_BigIntMul	TEE_OpenTASession
TEE_BigIntNeg	Key Derivation
TEE_BigIntSquare	TEE_DeriveKey
TEE_BigIntSub	Logical Operation
Cancellation	TEE_BigIntCmp
TEE_GetCancellationFlag	TEE_BigIntCmpS32
TEE_MaskCancellation	TEE_BigIntGetBit
TEE_UnmaskCancellation	TEE_BigIntGetBitCount
Converter	TEE_BigIntShiftRight
TEE_BigIntConvertFromOctetString	MAC
TEE_BigIntConvertFromS32	TEE_MACCompareFinal
TEE_BigIntConvertToOctetString	TEE_MACComputeFinal
TEE_BigIntConvertToS32	TEE_MACInit
Data Stream Access	TEE_MACUpdate
TEE_ReadObjectData	Memory Allocation and Size of Objects
TEE_SeekObjectData	TEE_BigIntFMMContextSizeInU32
TEE_TruncateObjectData	TEE_BigIntFMMSizeInU32
TEE_WriteObjectData	TEE_BigIntSizeInU32 (macro)
Deprecated	Memory Management
TEE_CloseAndDeletePersistentObject	TEE_CheckMemoryAccessRights
TEE_CopyObjectAttributes	TEE_Free
TEE_GetObjectInfo	TEE_GetInstanceData
TEE_RestrictObjectUsage	TEE_Malloc
Fast Modular Multiplication	TEE_MemCompare
TEE_BigIntComputeFMM	TEE_MemFill
TEE_BigIntConvertFromFMM	TEE_MemMove
TEE_BigIntConvertToFMM	TEE_Realloc
Generic Object	TEE_SetInstanceData
TEE_CloseObject	Message Digest
TEE_GetObjectBufferAttribute	TEE_DigestDoFinal
TEE_GetObjectInfo (deprecated)	TEE_DigestUpdate
TEE_GetObjectInfo1	Modular Arithmetic
TEE_GetObjectValueAttribute	TEE_BigIntAddMod
TEE_RestrictObjectUsage (deprecated)	TEE_BigIntInvMod
TEE_RestrictObjectUsage1	TEE_BigIntMod
Generic Operation	TEE_BigIntMulMod
TEE_AllocateOperation	TEE_BigIntSquareMod
TEE_CopyOperation	TEE_BigIntSubMod

Other Arithmetic

- TEE_BigIntComputeExtendedGcd
- TEE_BigIntIsProbablePrime
- TEE_BigIntRelativePrime

Panic Function

- TEE_Panic

Persistent Object

- TEE_CloseAndDeletePersistentObject
(deprecated)
- TEE_CloseAndDeletePersistentObject1
- TEE_CreatePersistentObject
- TEE_OpenPersistentObject
- TEE_RenamePersistentObject

Persistent Object Enumeration *

- TEE_AllocatePersistentObjectEnumerator
- TEE_FreePersistentObjectEnumerator
- TEE_GetNextPersistentObject
- TEE_ResetPersistentObjectEnumerator
- TEE_StartPersistentObjectEnumerator

Property Access

- TEE_AllocatePropertyEnumerator
- TEE_FreePropertyEnumerator
- TEE_GetNextProperty
- TEE_GetPropertyAsBinaryBlock
- TEE_GetPropertyAsBool
- TEE_GetPropertyAsIdentity
- TEE_GetPropertyAsString
- TEE_GetPropertyAsU32
- TEE_GetPropertyAsU64
- TEE_GetPropertyAsUUID
- TEE_GetPropertyName

- TEE_ResetPropertyEnumerator

- TEE_StartPropertyEnumerator

Random Data Generation

- TEE_GenerateRandom

Symmetric Cipher

- TEE_CipherDoFinal
- TEE_CipherInit
- TEE_CipherUpdate

TA Interface

- TA_CloseSessionEntryPoint
- TA_CreateEntryPoint
- TA_DestroyEntryPoint
- TA_InvokeCommandEntryPoint
- TA_OpenSessionEntryPoint

Time

- TEE_GetREETime
- TEE_GetSystemTime
- TEE_GetTAPersistentTime
- TEE_SetTAPersistentTime
- TEE_Wait

Transient Object

- TEE_AllocateTransientObject
- TEE_CopyObjectAttributes (deprecated)
- TEE_CopyObjectAttributes1
- TEE_FreeTransientObject
- TEE_GenerateKey
- TEE_InitRefAttribute
- TEE_InitValueAttribute
- TEE_PopulateTransientObject
- TEE_ResetTransientObject

7 Class Index

7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

_TEE_ObjectHandle	36
_TEE_OperationHandle	38
addrinfo	39
enclave_report	40
pollfd	41
report	42
sm_report	43
TEE_Attribute	43
TEE_Identity	45
TEE_ObjectInfo	46
TEE_OperationInfo	47
TEE_OperationInfoKey	48
TEE_OperationInfoMultiple	49
TEE_Param	50
TEE_SEAID	51
TEE_SEReaderProperties	52
TEE_Time	53
TEE_UUID	53
TEEC_Context	54
TEEC_Operation	55
TEEC_Parameter	56
TEEC_RegisteredMemoryReference	57
TEEC_Session	59
TEEC_SharedMemory	59
TEEC_TempMemoryReference	61
TEEC_UUID	62
TEEC_Value	63

8 File Index

8.1 File List

Here is a list of all files with brief descriptions:

ta-ref/api/tee-internal-api-cryptlib.c	165
ta-ref/api/include/compiler.h	64
ta-ref/api/include/report.h	67
ta-ref/api/include/tee-common.h	
Common type and definitions of RISC-V TEE	68
ta-ref/api/include/tee-ta-internal.h	
Candidate API list for Global Platform like RISC-V TEE	69
ta-ref/api/include/tee_api_defines.h	96
ta-ref/api/include/tee_api_defines_extensions.h	102
ta-ref/api/include/tee_api_types.h	104
ta-ref/api/include/tee_client_api.h	110
ta-ref/api/include/tee_internal_api.h	119
ta-ref/api/include/tee_internal_api_extensions.h	119
ta-ref/api/include/tee_ta_api.h	122
ta-ref/api/include/test_dev_key.h	126
ta-ref/api/include/trace.h	127
ta-ref/api/include/trace_levels.h	132
ta-ref/api/keystone/tee-internal-api-machine.c	133
ta-ref/api/keystone/tee-internal-api.c	135
ta-ref/api/keystone/tee_api_tee_types.h	154
ta-ref/api/keystone/teec_stub.c	160
ta-ref/api/keystone/trace.c	163
ta-ref/api/optee/tee_api_tee_types.h	157
ta-ref/api/sgx/tee-internal-api.c	145
ta-ref/api/sgx/tee_api_tee_types.h	157

9 Class Documentation

9.1 __TEE_ObjectHandle Struct Reference

```
#include <tee_api_tee_types.h>
```

Public Attributes

- unsigned int [type](#)
- int [flags](#)
- int [desc](#)
- mbedtls_aes_context [persist_ctx](#)
- unsigned char [persist_iv](#) [TEE_OBJECT_NONCE_SIZE]
- unsigned char [public_key](#) [TEE_OBJECT_KEY_SIZE]
- unsigned char [private_key](#) [TEE_OBJECT_SKEY_SIZE]

9.1.1 Member Data Documentation

9.1.1.1 desc int __TEE_ObjectHandle::desc

9.1.1.2 flags int __TEE_ObjectHandle::flags

9.1.1.3 persist_ctx mbedtls_aes_context __TEE_ObjectHandle::persist_ctx

9.1.1.4 persist_iv unsigned char __TEE_ObjectHandle::persist_iv

9.1.1.5 private_key unsigned char __TEE_ObjectHandle::private_key

9.1.1.6 public_key unsigned char __TEE_ObjectHandle::public_key

9.1.1.7 type unsigned int __TEE_ObjectHandle::type

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

- ta-ref/api/keystone/[tee_api_tee_types.h](#)
- ta-ref/api/sgx/[tee_api_tee_types.h](#)

9.2 __TEE_OperationHandle Struct Reference

```
#include <tee_api_tee_types.h>
```

Public Attributes

- int [mode](#)
- int [flags](#)
- int [alg](#)
- sha3_ctx_t [ctx](#)
- mbedtls_aes_context [aectx](#)
- mbedtls_gcm_context [aegcmctx](#)
- int [aegcm_state](#)
- unsigned char [aeiv](#) [TEE_OBJECT_NONCE_SIZE]
- unsigned char [aekey](#) [32]
- unsigned char [pubkey](#) [TEE_OBJECT_KEY_SIZE]
- unsigned char [prikey](#) [TEE_OBJECT_SKEY_SIZE]

9.2.1 Member Data Documentation

9.2.1.1 aectx mbedtls_aes_context __TEE_OperationHandle::aectx

9.2.1.2 aegcm_state int __TEE_OperationHandle::aegcm_state

9.2.1.3 aegcmctx mbedtls_gcm_context __TEE_OperationHandle::aegcmctx

9.2.1.4 aeiv unsigned char __TEE_OperationHandle::aeiv

9.2.1.5 aekey unsigned char __TEE_OperationHandle::aekey

9.2.1.6 alg int __TEE_OperationHandle::alg

9.2.1.7 ctx `sha3_ctx_t __TEE_OperationHandle::ctx`

9.2.1.8 flags `int __TEE_OperationHandle::flags`

9.2.1.9 mode `int __TEE_OperationHandle::mode`

9.2.1.10 prikey `unsigned char __TEE_OperationHandle::prikey`

9.2.1.11 pubkey `unsigned char __TEE_OperationHandle::pubkey`

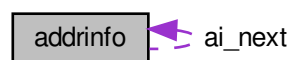
The documentation for this struct was generated from the following files:

- [ta-ref/api/keystone/tee_api_tee_types.h](#)
- [ta-ref/api/sgx/tee_api_tee_types.h](#)

9.3 addrinfo Struct Reference

```
#include <tee-api-types.h>
```

Collaboration diagram for addrinfo:



Public Attributes

- `int ai_flags`
- `int ai_family`
- `int ai_socktype`
- `int ai_protocol`
- `socklen_t ai_addrlen`
- `struct sockaddr * ai_addr`
- `char * ai_canonname`
- `struct addrinfo * ai_next`

9.3.1 Member Data Documentation

9.3.1.1 ai_addr `struct sockaddr* addrinfo::ai_addr`

9.3.1.2 ai_addrlen `socklen_t addrinfo::ai_addrlen`

9.3.1.3 ai_canonname `char* addrinfo::ai_canonname`

9.3.1.4 ai_family `int addrinfo::ai_family`

9.3.1.5 ai_flags `int addrinfo::ai_flags`

9.3.1.6 ai_next `struct addrinfo* addrinfo::ai_next`

9.3.1.7 ai_protocol `int addrinfo::ai_protocol`

9.3.1.8 ai_socktype `int addrinfo::ai_socktype`

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

- [ta-ref/api/include/tee_api_types.h](#)

9.4 enclave_report Struct Reference

```
#include <report.h>
```

Public Attributes

- uint8_t [hash](#) [MDSIZE]
- uint64_t [data_len](#)
- uint8_t [data](#) [ATTEST_DATA_MAXLEN]
- uint8_t [signature](#) [SIGNATURE_SIZE]

9.4.1 Member Data Documentation

9.4.1.1 data uint8_t enclave_report::data[ATTEST_DATA_MAXLEN]

9.4.1.2 data_len uint64_t enclave_report::data_len

9.4.1.3 hash uint8_t enclave_report::hash[MDSIZE]

9.4.1.4 signature uint8_t enclave_report::signature[SIGNATURE_SIZE]

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

- ta-ref/api/include/[report.h](#)

9.5 pollfd Struct Reference

```
#include <tee_api_types.h>
```

Public Attributes

- int [fd](#)
- short int [events](#)
- short int [revents](#)

9.5.1 Member Data Documentation

9.5.1.1 events `short int pollfd::events`

9.5.1.2 fd `int pollfd::fd`

9.5.1.3 revents `short int pollfd::revents`

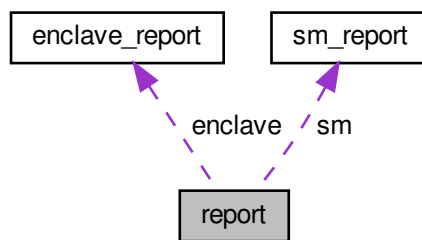
The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_api_types.h](#)

9.6 report Struct Reference

```
#include <report.h>
```

Collaboration diagram for report:



Public Attributes

- struct [enclave_report](#) `enclave`
- struct [sm_report](#) `sm`
- `uint8_t` [dev_public_key](#) `[PUBLIC_KEY_SIZE]`

9.6.1 Member Data Documentation

9.6.1.1 dev_public_key `uint8_t report::dev_public_key[PUBLIC_KEY_SIZE]`

9.6.1.2 enclave struct `enclave_report` `report::enclave`

9.6.1.3 sm struct `sm_report` `report::sm`

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

- [ta-ref/api/include/report.h](#)

9.7 sm_report Struct Reference

```
#include <report.h>
```

Public Attributes

- `uint8_t` `hash` [MDSIZE]
- `uint8_t` `public_key` [PUBLIC_KEY_SIZE]
- `uint8_t` `signature` [SIGNATURE_SIZE]

9.7.1 Member Data Documentation

9.7.1.1 hash `uint8_t` `sm_report::hash`[MDSIZE]

9.7.1.2 public_key `uint8_t` `sm_report::public_key`[PUBLIC_KEY_SIZE]

9.7.1.3 signature `uint8_t` `sm_report::signature`[SIGNATURE_SIZE]

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

- [ta-ref/api/include/report.h](#)

9.8 TEE Attribute Struct Reference

```
#include <tee_api_types.h>
```


Public Attributes

- uint32_t attributeID
- union {
 - struct {
 - void * buffer
 - uint32_t length
 - } ref
 - struct {
 - uint32_t a
 - uint32_t b
 - } value
- } content

9.8.1 Member Data Documentation

9.8.1.1 a uint32_t TEE_Attribute::a

9.8.1.2 attributeID uint32_t TEE_Attribute::attributeID

9.8.1.3 b uint32_t TEE_Attribute::b

9.8.1.4 buffer void* TEE_Attribute::buffer

9.8.1.5 union { ... } TEE_Attribute::content

9.8.1.6 length uint32_t TEE_Attribute::length

9.8.1.7 struct { ... } TEE_Attribute::ref

9.8.1.8 `struct { ... } TEE_Attribute::value`

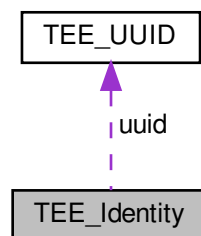
The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_api_types.h](#)

9.9 TEE_Identity Struct Reference

```
#include <tee_api_types.h>
```

Collaboration diagram for TEE_Identity:



Public Attributes

- `uint32_t login`
- `TEE_UUID uuid`

9.9.1 Member Data Documentation

9.9.1.1 `login` `uint32_t TEE_Identity::login`

9.9.1.2 `uuid` `TEE_UUID TEE_Identity::uuid`

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

- [ta-ref/api/include/tee_api_types.h](#)

9.10 TEE_ObjectInfo Struct Reference

```
#include <tee_api_types.h>
```

Public Attributes

- uint32_t [objectType](#)
- union {
 - uint32_t [keySize](#)
 - uint32_t [objectSize](#)
- };
- union {
 - uint32_t [maxKeySize](#)
 - uint32_t [maxObjectSize](#)
- };
- uint32_t [objectUsage](#)
- uint32_t [dataSize](#)
- uint32_t [dataPosition](#)
- uint32_t [handleFlags](#)

9.10.1 Member Data Documentation

9.10.1.1 `__extension__ union { ... } TEE_ObjectInfo::@3`

9.10.1.2 `__extension__ union { ... } TEE_ObjectInfo::@5`

9.10.1.3 **dataPosition** `uint32_t TEE_ObjectInfo::dataPosition`

9.10.1.4 **dataSize** `uint32_t TEE_ObjectInfo::dataSize`

9.10.1.5 **handleFlags** `uint32_t TEE_ObjectInfo::handleFlags`

9.10.1.6 keySize uint32_t TEE_ObjectInfo::keySize

9.10.1.7 maxKeySize uint32_t TEE_ObjectInfo::maxKeySize

9.10.1.8 maxObjectSize uint32_t TEE_ObjectInfo::maxObjectSize

9.10.1.9 objectSize uint32_t TEE_ObjectInfo::objectSize

9.10.1.10 objectType uint32_t TEE_ObjectInfo::objectType

9.10.1.11 objectUsage uint32_t TEE_ObjectInfo::objectUsage

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

- ta-ref/api/include/[tee_api_types.h](#)

9.11 TEE_OperationInfo Struct Reference

```
#include <tee_api_types.h>
```

Public Attributes

- uint32_t [algorithm](#)
- uint32_t [operationClass](#)
- uint32_t [mode](#)
- uint32_t [digestLength](#)
- uint32_t [maxKeySize](#)
- uint32_t [keySize](#)
- uint32_t [requiredKeyUsage](#)
- uint32_t [handleState](#)

9.11.1 Member Data Documentation

9.11.1.1 algorithm uint32_t TEE_OperationInfo::algorithm

9.11.1.2 digestLength uint32_t TEE_OperationInfo::digestLength

9.11.1.3 handleState uint32_t TEE_OperationInfo::handleState

9.11.1.4 keySize uint32_t TEE_OperationInfo::keySize

9.11.1.5 maxKeySize uint32_t TEE_OperationInfo::maxKeySize

9.11.1.6 mode uint32_t TEE_OperationInfo::mode

9.11.1.7 operationClass uint32_t TEE_OperationInfo::operationClass

9.11.1.8 requiredKeyUsage uint32_t TEE_OperationInfo::requiredKeyUsage

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

- ta-ref/api/include/[tee_api_types.h](#)

9.12 TEE_OperationInfoKey Struct Reference

```
#include <tee_api_types.h>
```

Public Attributes

- uint32_t [keySize](#)
- uint32_t [requiredKeyUsage](#)

9.12.1 Member Data Documentation

9.12.1.1 keySize `uint32_t TEE_OperationInfoKey::keySize`

9.12.1.2 requiredKeyUsage `uint32_t TEE_OperationInfoKey::requiredKeyUsage`

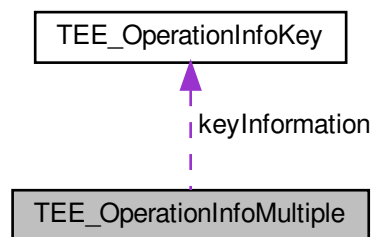
The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_api_types.h](#)

9.13 TEE_OperationInfoMultiple Struct Reference

```
#include <tee_api_types.h>
```

Collaboration diagram for TEE_OperationInfoMultiple:



Public Attributes

- `uint32_t` [algorithm](#)
- `uint32_t` [operationClass](#)
- `uint32_t` [mode](#)
- `uint32_t` [digestLength](#)
- `uint32_t` [maxKeySize](#)
- `uint32_t` [handleState](#)
- `uint32_t` [operationState](#)
- `uint32_t` [numberOfKeys](#)
- [TEE_OperationInfoKey](#) [keyInformation](#) []

9.13.1 Member Data Documentation

9.13.1.1 algorithm `uint32_t TEE_OperationInfoMultiple::algorithm`

9.13.1.2 digestLength `uint32_t TEE_OperationInfoMultiple::digestLength`

9.13.1.3 handleState `uint32_t TEE_OperationInfoMultiple::handleState`

9.13.1.4 keyInformation `TEE_OperationInfoKey TEE_OperationInfoMultiple::keyInformation[]`

9.13.1.5 maxKeySize `uint32_t TEE_OperationInfoMultiple::maxKeySize`

9.13.1.6 mode `uint32_t TEE_OperationInfoMultiple::mode`

9.13.1.7 numberOfKeys `uint32_t TEE_OperationInfoMultiple::numberOfKeys`

9.13.1.8 operationClass `uint32_t TEE_OperationInfoMultiple::operationClass`

9.13.1.9 operationState `uint32_t TEE_OperationInfoMultiple::operationState`

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

- [ta-ref/api/include/tee_api_types.h](#)

9.14 TEE_Param Union Reference

```
#include <tee_api_types.h>
```

Public Attributes

- struct {
 void * [buffer](#)
 uint32_t [size](#)
} [memref](#)
- struct {
 uint32_t [a](#)
 uint32_t [b](#)
} [value](#)

9.14.1 Member Data Documentation

9.14.1.1 a uint32_t TEE_Param::a

9.14.1.2 b uint32_t TEE_Param::b

9.14.1.3 buffer void* TEE_Param::buffer

9.14.1.4 struct { ... } TEE_Param::memref

9.14.1.5 size uint32_t TEE_Param::size

9.14.1.6 struct { ... } TEE_Param::value

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

- [ta-ref/api/include/tee_api_types.h](#)

9.15 TEE_SEAID Struct Reference

```
#include <tee_api_types.h>
```


Public Attributes

- `uint8_t * buffer`
- `size_t bufferLen`

9.15.1 Member Data Documentation

9.15.1.1 `buffer` `uint8_t* TEE_SEAID::buffer`

9.15.1.2 `bufferLen` `size_t TEE_SEAID::bufferLen`

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

- `ta-ref/api/include/tee_api_types.h`

9.16 TEE_SERedReaderProperties Struct Reference

```
#include <tee_api_types.h>
```

Public Attributes

- `bool sePresent`
- `bool teeOnly`
- `bool selectResponseEnable`

9.16.1 Member Data Documentation

9.16.1.1 `selectResponseEnable` `bool TEE_SERedReaderProperties::selectResponseEnable`

9.16.1.2 `sePresent` `bool TEE_SERedReaderProperties::sePresent`

9.16.1.3 `teeOnly` `bool TEE_SERedReaderProperties::teeOnly`

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

- `ta-ref/api/include/tee_api_types.h`

9.17 TEE_Time Struct Reference

```
#include <tee_api_types.h>
```

Public Attributes

- uint32_t [seconds](#)
- uint32_t [millis](#)

9.17.1 Member Data Documentation

9.17.1.1 millis uint32_t TEE_Time::millis

9.17.1.2 seconds uint32_t TEE_Time::seconds

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

- ta-ref/api/include/[tee_api_types.h](#)

9.18 TEE_UUID Struct Reference

```
#include <tee_api_types.h>
```

Public Attributes

- uint32_t [timeLow](#)
- uint16_t [timeMid](#)
- uint16_t [timeHiAndVersion](#)
- uint8_t [clockSeqAndNode](#) [8]

9.18.1 Member Data Documentation

9.18.1.1 clockSeqAndNode uint8_t TEE_UUID::clockSeqAndNode[8]

9.18.1.2 timeHiAndVersion `uint16_t TEE_UUID::timeHiAndVersion`

9.18.1.3 timeLow `uint32_t TEE_UUID::timeLow`

9.18.1.4 timeMid `uint16_t TEE_UUID::timeMid`

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

- [ta-ref/api/include/tee_api_types.h](#)

9.19 TEEC_Context Struct Reference

```
#include <tee_client_api.h>
```

Public Attributes

- `int` [fd](#)
- `bool` [reg_mem](#)

9.19.1 Detailed Description

struct [TEEC_Context](#) - Represents a connection between a client application and a TEE.

9.19.2 Member Data Documentation

9.19.2.1 fd `int TEEC_Context::fd`

9.19.2.2 reg_mem `bool TEEC_Context::reg_mem`

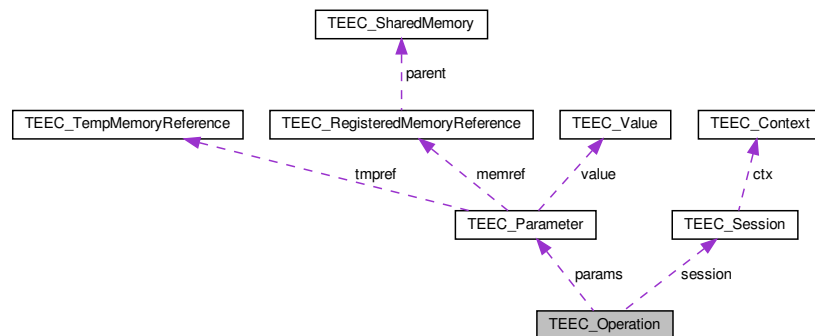
The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_client_api.h](#)

9.20 TEEC_Operation Struct Reference

```
#include <tee_client_api.h>
```

Collaboration diagram for TEEC_Operation:



Public Attributes

- `uint32_t started`
- `uint32_t paramTypes`
- `TEEC_Parameter params` [TEEC_CONFIG_PAYLOAD_REF_COUNT]
- `TEEC_Session * session`

9.20.1 Detailed Description

struct `TEEC_Operation` - Holds information and memory references used in `TEEC_InvokeCommand()`.

Parameters

<i>started</i>	Client must initialize to zero if it needs to cancel an operation about to be performed.
<i>paramTypes</i>	Type of data passed. Use <code>TEEC_PARAMS_TYPE</code> macro to create the correct flags. 0 means <code>TEEC_NONE</code> is passed for all params.
<i>params</i>	Array of parameters of type <code>TEEC_Parameter</code> .
<i>session</i>	Internal pointer to the last session used by <code>TEEC_InvokeCommand</code> with this operation.

9.20.2 Member Data Documentation

9.20.2.1 params `TEEC_Parameter` `TEEC_Operation::params` [TEEC_CONFIG_PAYLOAD_REF_COUNT]

9.20.2.2 paramTypes `uint32_t TEEC_Operation::paramTypes`

9.20.2.3 session `TEEC_Session* TEEC_Operation::session`

9.20.2.4 started `uint32_t TEEC_Operation::started`

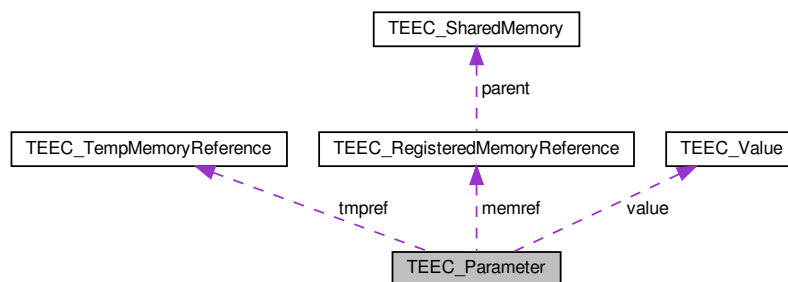
The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_client_api.h](#)

9.21 TEEC_Parameter Union Reference

```
#include <tee_client_api.h>
```

Collaboration diagram for TEEC_Parameter:



Public Attributes

- [TEEC_TempMemoryReference tmpref](#)
- [TEEC_RegisteredMemoryReference memref](#)
- [TEEC_Value value](#)

9.21.1 Detailed Description

union [TEEC_Parameter](#) - Memory container to be used when passing data between client application and trusted code.

Either the client uses a shared memory reference, parts of it or a small raw data container.

Parameters

<i>tmpref</i>	A temporary memory reference only valid for the duration of the operation.
<i>memref</i>	The entire shared memory or parts of it.
<i>value</i>	The small raw data container to use

9.21.2 Member Data Documentation

9.21.2.1 memref [TEEC_RegisteredMemoryReference](#) `TEEC_Parameter::memref`

9.21.2.2 tmpref [TEEC_TempMemoryReference](#) `TEEC_Parameter::tmpref`

9.21.2.3 value [TEEC_Value](#) `TEEC_Parameter::value`

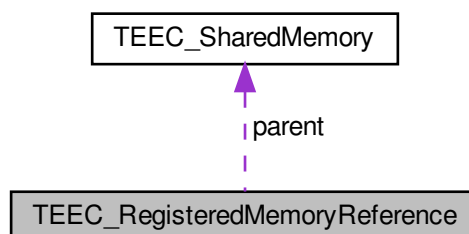
The documentation for this union was generated from the following file:

- [ta-ref/api/include/tee_client_api.h](#)

9.22 TEEC_RegisteredMemoryReference Struct Reference

```
#include <tee_client_api.h>
```

Collaboration diagram for TEEC_RegisteredMemoryReference:



Public Attributes

- [TEEC_SharedMemory](#) * [parent](#)
- [size_t](#) [size](#)
- [size_t](#) [offset](#)

9.22.1 Detailed Description

struct [TEEC_RegisteredMemoryReference](#) - use a pre-registered or pre-allocated shared memory block of memory to transfer data between a client application and trusted code.

Parameters

<i>parent</i>	Points to a shared memory structure. The memory reference may utilize the whole shared memory or only a part of it. Must not be NULL
<i>size</i>	The size, in bytes, of the memory buffer.
<i>offset</i>	The offset, in bytes, of the referenced memory region from the start of the shared memory block.

9.22.2 Member Data Documentation

9.22.2.1 offset `size_t TEEC_RegisteredMemoryReference::offset`

9.22.2.2 parent `TEEC_SharedMemory* TEEC_RegisteredMemoryReference::parent`

9.22.2.3 size `size_t TEEC_RegisteredMemoryReference::size`

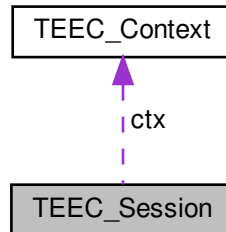
The documentation for this struct was generated from the following file:

- `ta-ref/api/include/tee_client_api.h`

9.23 TEEC_Session Struct Reference

```
#include <tee_client_api.h>
```

Collaboration diagram for TEEC_Session:



Public Attributes

- `TEEC_Context * ctx`
- `uint32_t session_id`

9.23.1 Detailed Description

struct `TEEC_Session` - Represents a connection between a client application and a trusted application.

9.23.2 Member Data Documentation

9.23.2.1 `ctx` `TEEC_Context*` `TEEC_Session::ctx`

9.23.2.2 `session_id` `uint32_t` `TEEC_Session::session_id`

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

- `ta-ref/api/include/tee_client_api.h`

9.24 TEEC_SharedMemory Struct Reference

```
#include <tee_client_api.h>
```


Public Attributes

- void * [buffer](#)
- size_t [size](#)
- uint32_t [flags](#)
- int [id](#)
- size_t [allocated_size](#)
- void * [shadow_buffer](#)
- int [registered_fd](#)
- bool [buffer_allocated](#)

9.24.1 Detailed Description

struct [TEEC_SharedMemory](#) - Memory to transfer data between a client application and trusted code.

Parameters

<i>buffer</i>	The memory buffer which is to be, or has been, shared with the TEE.
<i>size</i>	The size, in bytes, of the memory buffer.
<i>flags</i>	Bit-vector which holds properties of buffer. The bit-vector can contain either or both of the TEEC_MEM_INPUT and TEEC_MEM_OUTPUT flags.

A shared memory block is a region of memory allocated in the context of the client application memory space that can be used to transfer data between that client application and a trusted application. The user of this struct is responsible to populate the buffer pointer.

9.24.2 Member Data Documentation

9.24.2.1 [allocated_size](#) `size_t TEEC_SharedMemory::allocated_size`

9.24.2.2 [buffer](#) `void* TEEC_SharedMemory::buffer`

9.24.2.3 [buffer_allocated](#) `bool TEEC_SharedMemory::buffer_allocated`

9.24.2.4 [flags](#) `uint32_t TEEC_SharedMemory::flags`

9.24.2.5 id `int TEEC_SharedMemory::id`

9.24.2.6 registered_fd `int TEEC_SharedMemory::registered_fd`

9.24.2.7 shadow_buffer `void* TEEC_SharedMemory::shadow_buffer`

9.24.2.8 size `size_t TEEC_SharedMemory::size`

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

- [ta-ref/api/include/tee_client_api.h](#)

9.25 TEEC_TempMemoryReference Struct Reference

```
#include <tee_client_api.h>
```

Public Attributes

- `void *` [buffer](#)
- `size_t` [size](#)

9.25.1 Detailed Description

struct [TEEC_TempMemoryReference](#) - Temporary memory to transfer data between a client application and trusted code, only used for the duration of the operation.

Parameters

<i>buffer</i>	The memory buffer which is to be, or has been shared with the TEE.
<i>size</i>	The size, in bytes, of the memory buffer.

A memory buffer that is registered temporarily for the duration of the operation to be called.

9.25.2 Member Data Documentation

9.25.2.1 buffer `void* TEEC_TempMemoryReference::buffer`

9.25.2.2 size `size_t TEEC_TempMemoryReference::size`

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

- [ta-ref/api/include/tee_client_api.h](#)

9.26 TEEC_UUID Struct Reference

```
#include <tee_client_api.h>
```

Public Attributes

- `uint32_t timeLow`
- `uint16_t timeMid`
- `uint16_t timeHiAndVersion`
- `uint8_t clockSeqAndNode [8]`

9.26.1 Detailed Description

This type contains a Universally Unique Resource Identifier (UUID) type as defined in RFC4122. These UUID values are used to identify Trusted Applications.

9.26.2 Member Data Documentation

9.26.2.1 clockSeqAndNode `uint8_t TEEC_UUID::clockSeqAndNode[8]`

9.26.2.2 timeHiAndVersion `uint16_t TEEC_UUID::timeHiAndVersion`

9.26.2.3 timeLow `uint32_t TEEC_UUID::timeLow`

9.26.2.4 timeMid `uint16_t TEEC_UUID::timeMid`

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

- [ta-ref/api/include/tee_client_api.h](#)

9.27 TEEC_Value Struct Reference

```
#include <tee-client_api.h>
```

Public Attributes

- `uint32_t a`
- `uint32_t b`

9.27.1 Detailed Description

struct [TEEC_Value](#) - Small raw data container

Instead of allocating a shared memory buffer this structure can be used to pass small raw data between a client application and trusted code.

Parameters

<i>a</i>	The first integer value.
<i>b</i>	The second second value.

9.27.2 Member Data Documentation

9.27.2.1 **a** `uint32_t TEEC_Value::a`

9.27.2.2 **b** `uint32_t TEEC_Value::b`

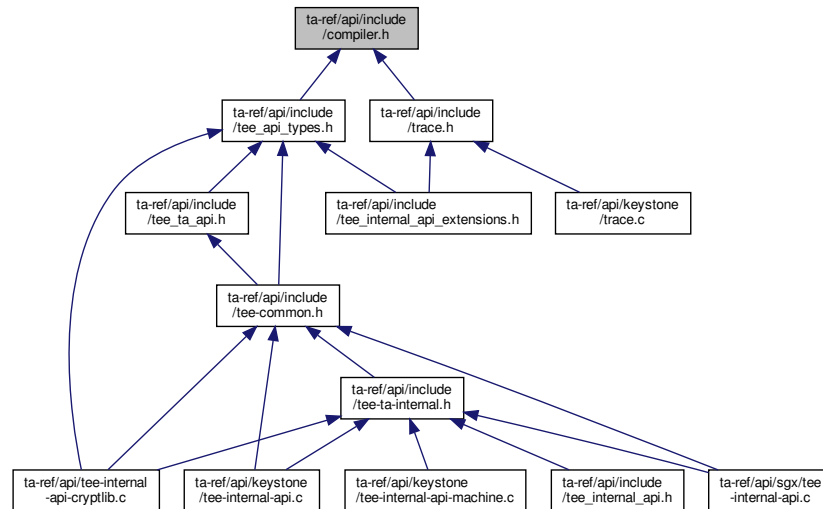
The documentation for this struct was generated from the following file:

- [ta-ref/api/include/tee_client_api.h](#)

10 File Documentation

10.1 ta-ref/api/include/compiler.h File Reference

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



10.2 compiler.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25 * POSSIBILITY OF SUCH DAMAGE.
26 */
27
28 #ifndef COMPILER_H
29 #define COMPILER_H
30
31 #ifndef DOXYGEN_SHOULD_SKIP_THIS
32 /*
33  * Macros that should be used instead of using __attribute__ directly to
34  * ease portability and make the code easier to read.
35  */
36
37 #define __deprecated __attribute__((deprecated))

```

```

38 #define __packed      __attribute__((packed))
39 #define __weak         __attribute__((weak))
40 #define __noreturn     __attribute__((noreturn))
41 #define __pure         __attribute__((pure))
42 #define __aligned(x)   __attribute__((aligned(x)))
43 #define __printf(a, b) __attribute__((format(printf, a, b)))
44 #define __noinline     __attribute__((noinline))
45 #define __attr_const    __attribute__((__const__))
46 #define __unused       __attribute__((unused))
47 #define __maybe_unused __attribute__((unused))
48 #define __used         __attribute__((__used__))
49 #define __must_check    __attribute__((warn_unused_result))
50 #define __cold         __attribute__((__cold__))
51 #define __section(x)    __attribute__((section(x)))
52 #define __data         __section("__data")
53 #define __bss          __section("__bss")
54 #define __rodata        __section("__rodata")
55 #define __rodata_unpaged __section("__rodata.unpaged")
56 #define __early_ta      __section("__rodata.early_ta")
57 #define __noprof        __attribute__((no_instrument_function))
58
59 #define __compiler_bswap64(x)  __builtin_bswap64(x)
60 #define __compiler_bswap32(x)  __builtin_bswap32(x)
61 #define __compiler_bswap16(x)  __builtin_bswap16(x)
62
63 #define __GCC_VERSION (__GNUC__ * 10000 + __GNUC_MINOR__ * 100 + \
64                      __GNUC_PATCHLEVEL__)
65
66 #if __GCC_VERSION >= 50100 && !defined(__CHECKER__)
67 #define __HAVE_BUILTIN_OVERFLOW 1
68 #endif
69
70 #ifdef __HAVE_BUILTIN_OVERFLOW
71 #define __compiler_add_overflow(a, b, res) \
72     __builtin_add_overflow(a, (b), (res))
73
74 #define __compiler_sub_overflow(a, b, res) \
75     __builtin_sub_overflow(a, (b), (res))
76
77 #define __compiler_mul_overflow(a, b, res) \
78     __builtin_mul_overflow(a, (b), (res))
79 #else
80 /*
81  * Copied/inspired from https://www.fefe.de/intof.html
82  */
83 #define __INTOF_HALF_MAX_SIGNED(type) ((type)1 << (sizeof(type)*8-2))
84 #define __INTOF_MAX_SIGNED(type) (__INTOF_HALF_MAX_SIGNED(type) - 1 + \
85                                  __INTOF_HALF_MAX_SIGNED(type))
86 #define __INTOF_MIN_SIGNED(type) (-1 - __INTOF_MAX_SIGNED(type))
87
88 #define __INTOF_MIN(type) ((type)-1 < 1?__INTOF_MIN_SIGNED(type):(type)0)
89 #define __INTOF_MAX(type) ((type)~__INTOF_MIN(type))
90
91 #define __INTOF_ASSIGN(dest, src) (__extension__({ \
92     typeof(src) __intof_x = (src); \
93     typeof(dest) __intof_y = __intof_x; \
94     (((uintmax_t)__intof_x == (uintmax_t)__intof_y) && \
95      ((__intof_x < 1) == (__intof_y < 1)) ? \
96      (void)((dest) = __intof_y, 0 : 1); \
97     }))
98
99 #define __INTOF_ADD(c, a, b) (__extension__({ \
100     typeof(a) __intofa_a = (a); \
101     typeof(b) __intofa_b = (b); \
102     \
103     __intofa_b < 1 ? \
104     ((__INTOF_MIN(typeof(c)) - __intofa_b <= __intofa_a) ? \
105      __INTOF_ASSIGN((c), __intofa_a + __intofa_b) : 1) : \
106     ((__INTOF_MAX(typeof(c)) - __intofa_b >= __intofa_a) ? \
107      __INTOF_ASSIGN((c), __intofa_a + __intofa_b) : 1); \
108     }))
109
110 #define __INTOF_SUB(c, a, b) (__extension__({ \
111     typeof(a) __intofs_a = a; \
112     typeof(b) __intofs_b = b; \
113     \
114     __intofs_b < 1 ? \
115     ((__INTOF_MAX(typeof(c)) + __intofs_b >= __intofs_a) ? \
116      __INTOF_ASSIGN((c), __intofs_a - __intofs_b) : 1) : \
117     ((__INTOF_MIN(typeof(c)) + __intofs_b <= __intofs_a) ? \
118      __INTOF_ASSIGN((c), __intofs_a - __intofs_b) : 1); \
119     }))
120
121 /*
122  * Dealing with detecting overflow in multiplication of integers.
123  * First step is to remove two corner cases with the minum signed integer

```

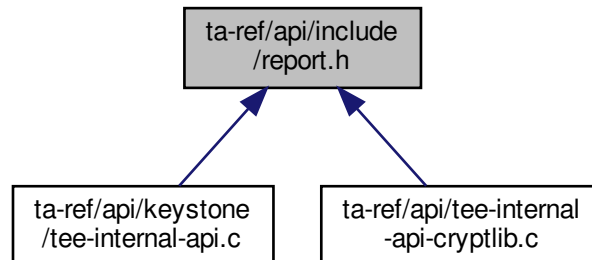
```

126 * which can't be represented as a positive integer + sign.
127 * Multiply with 0 or 1 can't overflow, no checking needed of the operation,
128 * only if it can be assigned to the result.
129 *
130 * After the corner cases are eliminated we convert the two factors to
131 * positive unsigned values, keeping track of the original in another
132 * variable which is used at the end to determine the sign of the product.
133 *
134 * The two terms (a and b) are divided into upper and lower half (x1 upper
135 * and x0 lower), so the product is:
136 * ((a1 << hshift) + a0) * ((b1 << hshift) + b0)
137 * which also is:
138 * ((a1 * b1) << (hshift * 2)) +          (T1)
139 * ((a1 * b0 + a0 * b1) << hshift) +      (T2)
140 * (a0 * b0)                             (T3)
141 *
142 * From this we can tell and (a1 * b1) has to be 0 or we'll overflow, that
143 * is, at least one of a1 or b1 has to be 0. Once this has been checked the
144 * addition: ((a1 * b0) << hshift) + ((a0 * b1) << hshift)
145 * isn't an addition as one of the terms will be 0.
146 *
147 * Since each factor in: (a0 * b0)
148 * only uses half the capacity of the underlying type it can't overflow
149 *
150 * The addition of T2 and T3 can overflow so we use __INTOF_ADD() to
151 * perform that addition. If the addition succeeds without overflow the
152 * result is assigned the required sign and checked for overflow again.
153 */
154
155 #define __intof_mul_negate ((__intof_oa < 1) != (__intof_ob < 1))
156 #define __intof_mul_hshift (sizeof(uintmax_t) * 8 / 2)
157 #define __intof_mul_hmask (UINTMAX_MAX >> __intof_mul_hshift)
158 #define __intof_mul_a0 ((uintmax_t)(__intof_a) >> __intof_mul_hshift)
159 #define __intof_mul_b0 ((uintmax_t)(__intof_b) >> __intof_mul_hshift)
160 #define __intof_mul_a1 ((uintmax_t)(__intof_a) & __intof_mul_hmask)
161 #define __intof_mul_b1 ((uintmax_t)(__intof_b) & __intof_mul_hmask)
162 #define __intof_mul_t (__intof_mul_a1 * __intof_mul_b0 + \
163     __intof_mul_a0 * __intof_mul_b1)
164
165 #define __INTOF_MUL(c, a, b) (__extension__({ \
166     typeof(a) __intof_oa = (a); \
167     typeof(a) __intof_a = __intof_oa < 1 ? -__intof_oa : __intof_oa; \
168     typeof(b) __intof_ob = (b); \
169     typeof(b) __intof_b = __intof_ob < 1 ? -__intof_ob : __intof_ob; \
170     typeof(c) __intof_c; \
171     \
172     __intof_oa == 0 || __intof_ob == 0 || \
173     __intof_oa == 1 || __intof_ob == 1 ? \
174     __INTOF_ASSIGN((c), __intof_oa * __intof_ob) : \
175     (__intof_mul_a0 && __intof_mul_b0) || \
176     __intof_mul_t > __intof_mul_hmask ? 1 : \
177     __INTOF_ADD((__intof_c), __intof_mul_t << __intof_mul_hshift, \
178         __intof_mul_a1 * __intof_mul_b1) ? 1 : \
179     __intof_mul_negate ? __INTOF_ASSIGN((c), -__intof_c) : \
180     __INTOF_ASSIGN((c), __intof_c); \
181 })))
182
183 #define __compiler_add_overflow(a, b, res) __INTOF_ADD(*(res), (a), (b))
184 #define __compiler_sub_overflow(a, b, res) __INTOF_SUB(*(res), (a), (b))
185 #define __compiler_mul_overflow(a, b, res) __INTOF_MUL(*(res), (a), (b))
186
187 #endif
188 #define __compiler_compare_and_swap(p, oval, nval) \
189     __atomic_compare_exchange_n((p), (oval), (nval), true, \
190         __ATOMIC_ACQUIRE, __ATOMIC_RELAXED) \
191
192 #define __compiler_atomic_load(p) __atomic_load_n((p), __ATOMIC_RELAXED)
193 #define __compiler_atomic_store(p, val) \
194     __atomic_store_n((p), (val), __ATOMIC_RELAXED)
195
196 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
197 #endif /*COMPILER_H*/

```

10.3 ta-ref/api/include/report.h File Reference

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



Classes

- struct [enclave_report](#)
- struct [sm_report](#)
- struct [report](#)

10.4 report.h

[Go to the documentation of this file.](#)

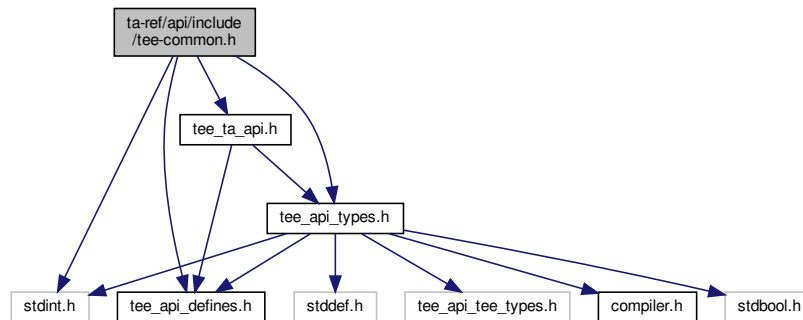
```

1
2 #ifndef _REPORT_H
3 #define _REPORT_H
4
5 #ifndef DOXYGEN_SHOULD_SKIP_THIS
6 #define MDSIZE 64
7 #define SIGNATURE_SIZE 64
8 #define PUBLIC_KEY_SIZE 32
9 #define ATTEST_DATA_MAXLEN 1024
10 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
11
12 /* attestation reports */
13 struct enclave_report
14 {
15     uint8_t hash[MDSIZE];
16     uint64_t data_len;
17     uint8_t data[ATTEST_DATA_MAXLEN];
18     uint8_t signature[SIGNATURE_SIZE];
19 };
20
21 struct sm_report
22 {
23     uint8_t hash[MDSIZE];
24     uint8_t public_key[PUBLIC_KEY_SIZE];
25     uint8_t signature[SIGNATURE_SIZE];
26 };
27
28 struct report
29 {
30     struct enclave_report enclave;
31     struct sm_report sm;
32     uint8_t dev_public_key[PUBLIC_KEY_SIZE];
33 };
34
35 #endif // _REPORT_H
  
```

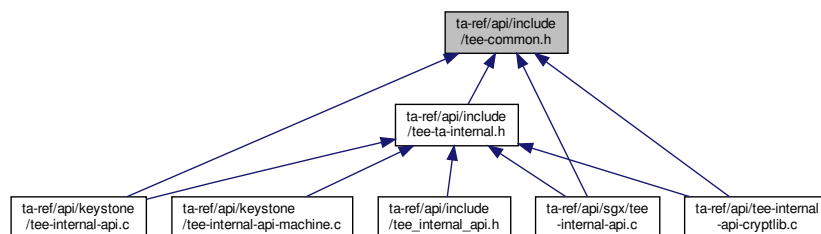

10.5 ta-ref/api/include/tee-common.h File Reference

Common type and definitions of RISC-V TEE.

```
#include <stdint.h>
#include <tee_api_defines.h>
#include <tee_api_types.h>
#include <tee_ta_api.h>
Include dependency graph for tee-common.h:
```



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



10.5.1 Detailed Description

Common type and definitions of RISC-V TEE.

draft RISC-V Internal TEE API

Author

Akira Tsukamoto, AIST

Date

2019/09/25

10.6 tee-common.h

[Go to the documentation of this file.](#)

```

1 /*
2  * SPDX-License-Identifier: BSD-2-Clause
3  *
4  * Copyright (C) 2019 National Institute of Advanced Industrial Science
5  *                               and Technology (AIST)
6  * All rights reserved.
7  *
8  * Redistribution and use in source and binary forms, with or without
9  * modification, are permitted provided that the following conditions are met:
10 *
11 * 1. Redistributions of source code must retain the above copyright notice,
12 * this list of conditions and the following disclaimer.
13 *
14 * 2. Redistributions in binary form must reproduce the above copyright notice,
15 * this list of conditions and the following disclaimer in the documentation
16 * and/or other materials provided with the distribution.
17 *
18 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
22 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
28 * POSSIBILITY OF SUCH DAMAGE.
29 */
30
31 #ifndef TEE_COMMON_H
32 #define TEE_COMMON_H
33
34 #include <stdint.h>
35
36 #ifdef __cplusplus
37 extern "C" {
38 #endif
39
40 #ifndef DOXYGEN_SHOULD_SKIP_THIS
41 #ifdef DEBUG
42 #define pr_deb(...) do { printf(__VA_ARGS__); } while (0)
43 #else
44 #define pr_deb(...) do { } while (0)
45 #endif /* DEBUG */
46 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
47
48 // #include <tee_api.h>
49 #include <tee_api_defines.h>
50 #include <tee_api_types.h>
51 #include <tee_ta_api.h>
52
53 // typedef uint32_t TEE_Result;
54
55 #ifdef __cplusplus
56 }
57 #endif
58 #endif /* TEE_COMMON_H */

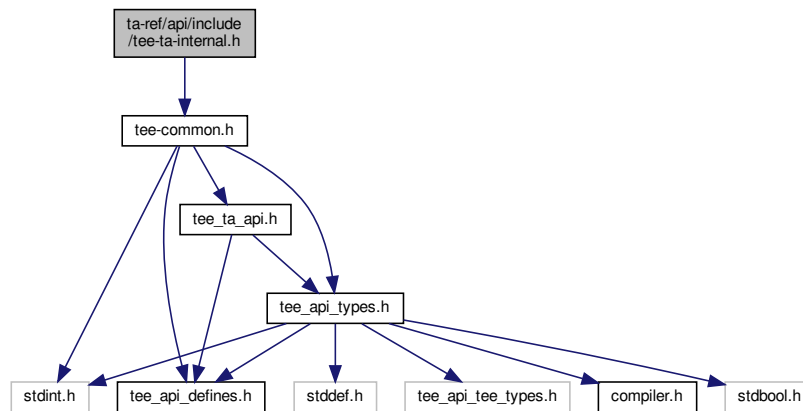
```

10.7 ta-ref/api/include/tee-ta-internal.h File Reference

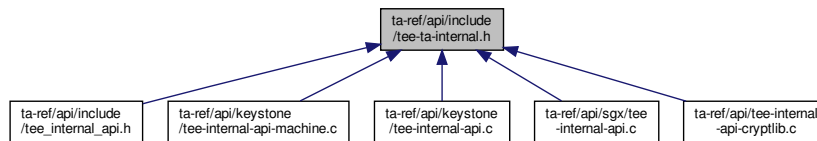
Candidate API list for Global Platform like RISC-V TEE.

```
#include "tee-common.h"
```

Include dependency graph for tee-ta-internal.h:



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



Functions

- void `__attribute__((noreturn)) TEE_Panic(unsigned long code)`
Core Functions, Time Functions.
- void `TEE_GetREETime (TEE_Time *time)`
Core Functions, Time Functions.
- void `TEE_GetSystemTime (TEE_Time *time)`
Core Functions, Time Functions.
- `TEE_Result GetRelTimeStart (uint64_t start)`
Core Functions, Time Functions.
- `TEE_Result GetRelTimeEnd (uint64_t end)`
Core Functions, Time Functions.
- `TEE_Result TEE_CreatePersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle attributes, const void *initialData, uint32_t initialDataLen, TEE_ObjectHandle *object)`
Core Functions, Secure Storage Functions (data is isolated for each TA)
- `TEE_Result TEE_OpenPersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle *object)`
Core Functions, Secure Storage Functions (data is isolated for each TA)
- `TEE_Result TEE_GetObjectInfo1 (TEE_ObjectHandle object, TEE_ObjectInfo *objectInfo)`
Core Functions, Secure Storage Functions (data is isolated for each TA)

- **TEE_Result TEE_WriteObjectData** (**TEE_ObjectHandle** object, const void *buffer, uint32_t size)
Core Functions, Secure Storage Functions (data is isolated for each TA)
- **TEE_Result TEE_ReadObjectData** (**TEE_ObjectHandle** object, void *buffer, uint32_t size, uint32_t *count)
Core Functions, Secure Storage Functions (data is isolated for each TA)
- void **TEE_CloseObject** (**TEE_ObjectHandle** object)
Core Functions, Secure Storage Functions (data is isolated for each TA)
- void **TEE_GenerateRandom** (void *randomBuffer, uint32_t randomBufferLen)
Crypto, common.
- **TEE_Result TEE_AllocateOperation** (**TEE_OperationHandle** *operation, uint32_t algorithm, uint32_t mode, uint32_t maxKeySize)
Crypto, for all Crypto Functions.
- void **TEE_FreeOperation** (**TEE_OperationHandle** operation)
Crypto, for all Crypto Functions.
- void **TEE_DigestUpdate** (**TEE_OperationHandle** operation, const void *chunk, uint32_t chunkSize)
Crypto, Message Digest Functions.
- **TEE_Result TEE_DigestDoFinal** (**TEE_OperationHandle** operation, const void *chunk, uint32_t chunkLen, void *hash, uint32_t *hashLen)
- **TEE_Result TEE_SetOperationKey** (**TEE_OperationHandle** operation, **TEE_ObjectHandle** key)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- **TEE_Result TEE_AEInit** (**TEE_OperationHandle** operation, const void *nonce, uint32_t nonceLen, uint32_t tagLen, uint32_t AADLen, uint32_t payloadLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- **TEE_Result TEE_AEUpdate** (**TEE_OperationHandle** operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- void **TEE_AEUpdateAAD** (**TEE_OperationHandle** operation, const void *AADdata, uint32_t AADdataLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- **TEE_Result TEE_AEEncryptFinal** (**TEE_OperationHandle** operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen, void *tag, uint32_t *tagLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- **TEE_Result TEE_AEDecryptFinal** (**TEE_OperationHandle** operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen, void *tag, uint32_t tagLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- void **TEE_CipherInit** (**TEE_OperationHandle** operation, const void *nonce, uint32_t nonceLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- **TEE_Result TEE_CipherUpdate** (**TEE_OperationHandle** operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- **TEE_Result TEE_GenerateKey** (**TEE_ObjectHandle** object, uint32_t keySize, const **TEE_Attribute** *params, uint32_t paramCount)
Crypto, Asymmetric key Verification Functions.
- **TEE_Result TEE_AllocateTransientObject** (**TEE_ObjectType** objectType, uint32_t maxKeySize, **TEE_ObjectHandle** *object)
Crypto, Asymmetric key Verification Functions.
- void **TEE_InitRefAttribute** (**TEE_Attribute** *attr, uint32_t attributeID, const void *buffer, uint32_t length)
Crypto, Asymmetric key Verification Functions.
- void **TEE_InitValueAttribute** (**TEE_Attribute** *attr, uint32_t attributeID, uint32_t a, uint32_t b)
Crypto, Asymmetric key Verification Functions.
- void **TEE_FreeTransientObject** (**TEE_ObjectHandle** object)
Crypto, Asymmetric key Verification Functions.
- **TEE_Result TEE_AsymmetricSignDigest** (**TEE_OperationHandle** operation, const **TEE_Attribute** *params, uint32_t paramCount, const void *digest, uint32_t digestLen, void *signature, uint32_t *signatureLen)

Crypto, Asymmetric key Verification Functions.

- `TEE_Result TEE_AsymmetricVerifyDigest (TEE_OperationHandle operation, const TEE_Attribute *params, uint32_t paramCount, const void *digest, uint32_t digestLen, const void *signature, uint32_t signatureLen)`

Crypto, Asymmetric key Verification Functions.

10.7.1 Detailed Description

Candidate API list for Global Platform like RISC-V TEE.

draft RISC-V Internal TEE API

Author

Akira Tsukamoto, AIST

Date

2019/09/25

10.7.2 Function Documentation

10.7.2.1 `__attribute__((noret)) void __attribute__((noret))`

TEE.Panic() - Raises a panic in the Trusted Application instance.

When a Trusted Application calls the TEE.Panic function, the current instance shall be destroyed and all the resources opened by the instance shall be reclaimed. All sessions opened from the panicking instance on another TA shall be gracefully closed and all cryptographic objects and operations shall be closed properly.

Parameters

<i>code</i>	An informative panic code defined by the TA.
-------------	--

Returns

panic code will be returned.

TEE.Panic() - Raises a Panic in the Trusted Application instance

When a Trusted Application calls the TEE.Panic function, the current instance shall be destroyed and all the resources opened by the instance shall be reclaimed.

Parameters

<i>ec</i>	An informative panic code defined by the TA. May be displayed in traces if traces are available.
-----------	--

10.7.2.2 GetRelTimeEnd()

```
TEE_Result GetRelTimeEnd (
    uint64_t end )
```

Core Functions, Time Functions.

Return the elapsed.

[GetRelTimeEnd\(\)](#) - finds the real time of the end timing.

This function prints the ending time.

Parameters

<i>end</i>	End timing
------------	------------

Returns

0 If success

[GetRelTimeStart\(\)](#) - find the real time of the end timing.

This function prints the End timing.

Parameters

<i>end</i>	End timing
------------	------------

Returns

0 if success else error occurred

10.7.2.3 GetRelTimeStart()

```
TEE_Result GetRelTimeStart (
    uint64_t start )
```

Core Functions, Time Functions.

Fast relative Time function which guarantees no hart switch or context switch between Trusted and Untrusted sides.

Most of the time ending up writing similar functions when only measuring the relative time in usec resolution which do not require the quality of the time itself but the distance of the two points.
For the usage above, the function does not have to return wall clock time.
Not prepared in both Keystone and GP.

[GetRelTimeStart\(\)](#) - Gets the real time of the start timing.

This function prints the starting time.

Parameters

<i>start</i>	Start timing
--------------	--------------

Returns

0 on success

[GetRelTimeStart\(\)](#) - Gets the real time of the start timing.

This function prints the start timing.

Parameters

<i>start</i>	start timing
--------------	--------------

Returns

0 if success else error occurred.

10.7.2.4 TEE_AEDecryptFinal() `TEE_Result TEE_AEDecryptFinal (`
 `TEE_OperationHandle operation,`
 `const void * srcData,`
 `uint32_t srcLen,`
 `void * destData,`
 `uint32_t * destLen,`
 `void * tag,`
 `uint32_t tagLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CCM, TEE_ALG_AES_GCM.

[TEE_AEDecryptFinal\(\)](#) - Processes data that has not been processed by previous calls to TEE_AEUpdate as well as data supplied in srcData.

This function completes the AE operation and compares the computed tag with the tag supplied in the parameter tag. The operation handle can be reused or newly initialized. The buffers srcData and destData shall be either completely disjoint or equal in their starting positions. The operation may be in either initial or active state and enters initial state afterwards.

Parameters

<i>operation</i>	Handle of a running AE operation
<i>srcData</i>	Reference to final chunk of input data to be encrypted
<i>srcLen</i>	length of the input data
<i>destData</i>	Output buffer. Can be omitted if the output is to be discarded.
<i>destLen</i>	length of the buffer.
<i>tag</i>	Output buffer filled with the computed tag
<i>tagLen</i>	length of the tag.

Returns

0 on success.

TEE_ERROR_SHORT_BUFFER If the output buffer is not large enough to contain the output

TEE_ERROR_MAC_INVALID If the computed tag does not match the supplied tag

10.7.2.5 TEE_AEEncryptFinal() `TEE_Result TEE_AEEncryptFinal (`
`TEE_OperationHandle operation,`
`const void * srcData,`
`uint32_t srcLen,`
`void * destData,`
`uint32_t * destLen,`
`void * tag,`
`uint32_t * tagLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CCM, TEE_ALG_AES_GCM.

[TEE_AEEncryptFinal\(\)](#) - processes data that has not been processed by previous calls to TEE_AEUpdate as well as data supplied in srcData .

TEE_AEEncryptFinal completes the AE operation and computes the tag. The operation handle can be reused or newly initialized. The buffers srcData and destData SHALL be either completely disjoint or equal in their starting positions. The operation may be in either initial or active state and enters initial state afterwards.

Parameters

<i>operation</i>	Handle of a running AE operation
<i>srcData</i>	Reference to final chunk of input data to be encrypted
<i>srcLen</i>	length of the input data
<i>destData</i>	Output buffer. Can be omitted if the output is to be discarded.
<i>destLen</i>	length of the buffer.
<i>tag</i>	Output buffer filled with the computed tag
<i>tagLen</i>	length of the tag.

Returns

0 on success.

TEE_ERROR_SHORT_BUFFER If the output or tag buffer is not large enough to contain the output.

10.7.2.6 TEE_AEInit() `TEE_Result TEE_AEInit (`
 `TEE_OperationHandle operation,`
 `const void * nonce,`
 `uint32_t nonceLen,`
 `uint32_t tagLen,`
 `uint32_t AADLen,`
 `uint32_t payloadLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CCM, TEE_ALG_AES_GCM.

[TEE_AEInit\(\)](#) - Initializes an Authentication Encryption operation.

The operation must be in initial state and remains in the initial state afterwards.

Parameters

<i>operation</i>	A handle on the operation.
<i>nonce</i>	The operation nonce or IV
<i>nonceLen</i>	length of nonce
<i>tagLen</i>	Size in bits of the tag
<i>AADLen</i>	Length in bytes of the AAD
<i>payloadLen</i>	Length in bytes of the payload.

Returns

0 on success.

TEE_ERROR_NOT_SUPPORTED If the tag length is not supported by the algorithm.

10.7.2.7 TEE_AEUpdate() `TEE_Result TEE_AEUpdate (`
 `TEE_OperationHandle operation,`
 `const void * srcData,`
 `uint32_t srcLen,`
 `void * destData,`
 `uint32_t * destLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CCM, TEE_ALG_AES_GCM.

[TEE_AEUpdate\(\)](#) - Accumulates data for an Authentication Encryption operation

This function describes Input data does not have to be a multiple of block size. Subsequent calls to this function are possible. Unless one or more calls of this function have supplied sufficient input data, no output is generated. when using this routine to decrypt the returned data may be corrupt since the integrity check is not performed until all the data has been processed. If this is a concern then only use the TEE_AEDecryptFinal routine.

Parameters

<i>operation</i>	Handle of a running AE operation.
<i>srcData</i>	Input data buffer to be encrypted or decrypted
<i>srcLen</i>	length of the input buffer.
<i>destData</i>	Output buffer
<i>destLen</i>	length of the out put buffer.

Returns

0 on success.

TEE_ERROR_SHORT_BUFFER if the output buffer is not large enough to contain the output.

10.7.2.8 TEE_AEUpdateAAD() `void TEE_AEUpdateAAD (`
`TEE_OperationHandle operation,`
`const void * AADdata,`
`uint32_t AADdataLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CCM, TEE_ALG_AES_GCM.

[TEE_AEUpdateAAD\(\)](#) - Feeds a new chunk of Additional Authentication Data (AAD) to the AE operation. Subsequent calls to this function are possible.

The TEE_AEUpdateAAD function feeds a new chunk of Additional Authentication Data (AAD) to the AE operation. Subsequent calls to this function are possible. The buffers *srcData* and *destData* shall be either completely disjoint or equal in their starting positions. The operation SHALL be in initial state and remains in initial state afterwards.

Parameters

<i>operation</i>	Handle on the AE operation
<i>AADdata</i>	Input buffer containing the chunk of AAD
<i>AADdataLen</i>	length of the chunk of AAD.

10.7.2.9 TEE_AllocateOperation() `TEE_Result TEE_AllocateOperation (`
`TEE_OperationHandle * operation,`
`uint32_t algorithm,`
`uint32_t mode,`
`uint32_t maxKeySize)`

Crypto, for all Crypto Functions.

All Crypto Functions use TEE_OperationHandle* operation instances.
 Create Crypto instance.

[TEE_AllocateOperation\(\)](#) - Allocates a handle for a new cryptographic operation and sets the mode and algorithm type.

If this function does not return with TEE_SUCCESS then there is no valid handle value. Once a cryptographic operation has been created, the implementation shall guarantee that all resources necessary for the operation are allocated and that any operation with a key of at most maxKeySize bits can be performed. For algorithms that take multiple keys, for example the AES XTS algorithm, the maxKeySize parameter specifies the size of the largest key. It is up to the implementation to properly allocate space for multiple keys if the algorithm so requires.

Parameters

<i>operation</i>	reference to generated operation handle.
<i>algorithm</i>	One of the cipher algorithms.
<i>mode</i>	The operation mode.
<i>maxKeySize</i>	Maximum key size in bits for the operation.

Returns

0 in case of success

TEE_ERROR_OUT_OF_MEMORY If there are not enough resources to allocate the operation.

TEE_ERROR_NOT_SUPPORTED If the mode is not compatible with the algorithm or key size or if the algorithm is not one of the listed algorithms or if maxKeySize is not appropriate for the algorithm.

10.7.2.10 TEE_AllocateTransientObject() `TEE_Result TEE_AllocateTransientObject (`
`TEE_ObjectType objectType,`
`uint32_t maxKeySize,`
`TEE_ObjectHandle * object)`

Crypto, Asymmetric key Verification Functions.

Create object storing asymmetric key.

[TEE_AllocateTransientObject\(\)](#) - Allocates an uninitialized transient object. Transient objects are used to hold a cryptographic object (key or key-pair).

The value TEE.KEYSIZE.NO_KEY should be used for maxObjectSize for object types that do not require a key so that all the container resources can be pre-allocated. As allocated, the container is uninitialized. It can be initialized by subsequently importing the object material, generating an object, deriving an object, or loading an object from the Trusted Storage.

Parameters

<i>objectType</i>	Type of uninitialized object container to be created
<i>maxKeySize</i>	Key Size of the object.
<i>object</i>	Filled with a handle on the newly created key container.

Returns

0 on success

TEE_ERROR_OUT_OF_MEMORY If not enough resources are available to allocate the object handle.

TEE_ERROR_NOT_SUPPORTED If the key size is not supported or the object type is not supported.

10.7.2.11 TEE_AsymmetricSignDigest() `TEE_Result TEE_AsymmetricSignDigest (`
`TEE.OperationHandle operation,`
`const TEE.Attribute * params,`
`uint32_t paramCount,`
`const void * digest,`
`uint32_t digestLen,`
`void * signature,`
`uint32_t * signatureLen)`

Crypto, Asymmetric key Verification Functions.

Sign a message digest within an asymmetric key operation.

Keystone has `ed25519_sign()`.

Equivalent in openssl is `EVP_DigestSign()`.

[TEE_AsymmetricSignDigest\(\)](#) - Signs a message digest within an asymmetric operation.

Parameters

<i>operation</i>	Handle on the operation, which SHALL have been suitably set up with an operation key.
<i>params</i>	Optional operation parameters
<i>paramCount</i>	size of param
<i>digest</i>	Input buffer containing the input message digest
<i>digestLen</i>	length of input buffer.
<i>signature</i>	Output buffer written with the signature of the digest
<i>signatureLen</i>	length of output buffer.

Returns

0 on success

TEE_ERROR_SHORT_BUFFER If the signature buffer is not large enough to hold the result

10.7.2.12 TEE_AsymmetricVerifyDigest() `TEE_Result TEE_AsymmetricVerifyDigest (`
`TEE.OperationHandle operation,`
`const TEE.Attribute * params,`
`uint32_t paramCount,`
`const void * digest,`
`uint32_t digestLen,`

```
const void * signature,
uint32_t signatureLen )
```

Crypto, Asymmetric key Verification Functions.

Verifies a message digest signature within an asymmetric key operation.

Keystone has `ed25519_verify()`.

Equivalent in openssl is `EVP_DigestVerify()`.

[TEE_AsymmetricVerifyDigest\(\)](#) - verifies a message digest signature within an asymmetric operation.

This function describes the message digest signature verify by calling `ed25519_verify()`.

Parameters

<i>operation</i>	Handle on the operation, which SHALL have been suitably set up with an operation key.
<i>params</i>	Optional operation parameters
<i>paramCount</i>	size of param.
<i>digest</i>	Input buffer containing the input message digest
<i>digestLen</i>	length of input buffer.
<i>signature</i>	Output buffer written with the signature of the digest
<i>signatureLen</i>	length of output buffer.

Returns

TEE_SUCCESS on success

TEE_ERROR_SIGNATURE_INVALID if the signature is invalid.

10.7.2.13 TEE_CipherInit() `void TEE_CipherInit (`
`TEE_OperationHandle operation,`
`const void * nonce,`
`uint32_t nonceLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CBC.

[TEE_CipherInit\(\)](#) - starts the symmetric cipher operation.

The operation shall have been associated with a key. If the operation is in active state, it is reset and then initialized. If the operation is in initial state, it is moved to active state.

Parameters

<i>operation</i>	A handle on an opened cipher operation setup with a key
<i>nonce</i>	Buffer containing the operation Initialization Vector as appropriate.
<i>nonceLen</i>	length of the buffer

10.7.2.14 TEE_CipherUpdate() `TEE.Result` TEE_CipherUpdate (
 `TEE.OperationHandle` operation,
 const void * srcData,
 uint32_t srcLen,
 void * destData,
 uint32_t * destLen)

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Supports TEE_ALG_AES_CBC.

`TEE_CipherUpdate()` - encrypts or decrypts input data.

Input data does not have to be a multiple of block size. Subsequent calls to this function are possible. Unless one or more calls of this function have supplied sufficient input data, no output is generated. The cipher operation is finalized with a call to `TEE_CipherDoFinal`. The buffers `srcData` and `destData` SHALL be either completely disjoint or equal in their starting positions. The operation SHALL be in active state.

Parameters

<i>operation</i>	Handle of a running Cipher operation
<i>srcData</i>	Input data buffer to be encrypted or decrypted
<i>srcLen</i>	length of input buffer
<i>destData</i>	output buffer
<i>destLen</i>	output buffer length.

Returns

0 on success else

TEE_ERROR_SHORT_BUFFER If the output buffer is not large enough to contain the output. In this case, the input is not fed into the algorithm.

10.7.2.15 TEE_CloseObject() void TEE_CloseObject (
 `TEE.ObjectHandle` object)

Core Functions, Secure Storage Functions (data is isolated for each TA)

Destroy object (key, key-pair or Data).

`TEE_CloseObject()` - Closes an opened object handle.

The object can be persistent or transient. For transient objects, `TEE_CloseObject` is equivalent to `TEE_FreeTransientObject`.

Parameters

<i>object</i>	Handle of the object.
---------------	-----------------------

Returns

TEE_SUCCESS if success else error occurred.

[TEE_CloseObject\(\)](#) - Function closes an opened object handle.

The object can be persistent or transient. For transient objects, TEE_CloseObject is equivalent to TEE_Free↔ TransientObject.

Parameters

<i>object</i>	Handle of the object
---------------	----------------------

Returns

TEE_SUCCESS if success else error occurred.

10.7.2.16 TEE_CreatePersistentObject() `TEE_Result TEE_CreatePersistentObject (`
 uint32_t *storageID*,
 const void * *objectID*,
 uint32_t *objectIDLen*,
 uint32_t *flags*,
 [TEE_ObjectHandle](#) *attributes*,
 const void * *initialData*,
 uint32_t *initialDataLen*,
 [TEE_ObjectHandle](#) * *object*)

Core Functions, Secure Storage Functions (data is isolated for each TA)

Create persistent object (key, key-pair or Data).

For the people who have not written code on GP then probably do not need to care the meaning of what is Persistent Object is, since the following are enough to use secure storage feature.

[TEE_CreatePersistentObject\(\)](#) - Creates a persistent object with initial attributes.

In this function an initial data stream content returns either a handle on the created object or TEE_HANDLE_NULL upon failure.

Parameters

<i>storageID</i>	The storage to use.
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
Paramter list continued on next page	

<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>attributes</i>	A handle on a persistent object or an initialized transient object from which to take the persistent object attributes
<i>initialData</i>	The initial data content of the persistent object
<i>initialDataLen</i>	The initial data content of the persistent object
<i>object</i>	A pointer to the handle which contains the opened handle upon successful completion

Returns

0 if success else error occurred.

[TEE_CreatePersistentObject\(\)](#) - Creates a persistent object with initial attributes.

An initial data stream content, and optionally returns either a handle on the created object, or TEE_HANDLE_NULL upon failure.

Parameters

<i>storageID</i>	The storage to use.
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>attributes</i>	A handle on a persistent object or an initialized transient object from which to take the persistent object attributes
<i>initialData</i>	The initial data content of the persistent object
<i>initialDataLen</i>	The initial data content of the persistent object
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion

Returns

0 if success, else error occurred.

10.7.2.17 TEE_DigestDoFinal() `TEE_Result TEE_DigestDoFinal (`
`TEE_OperationHandle operation,`
`const void * chunk,`
`uint32_t chunkLen,`
`void * hash,`
`uint32_t * hashLen)`

Function accumulates message data for hashing.

[TEE_DigestDoFinal\(\)](#) - Finalizes the message digest operation and produces the message hash.

This function finalizes the message digest operation and produces the message hash. Afterwards the Message Digest operation is reset to initial state and can be reused.

Parameters

<i>operation</i>	Handle of a running Message Digest operation.
<i>chunk</i>	Chunk of data to be hashed.
<i>chunkLen</i>	size of the chunk.
<i>hash</i>	Output buffer filled with the message hash.
<i>hashLen</i>	length of the message hash.

Returns

0 on success

TEE_ERROR_SHORT_BUFFER If the output buffer is too small. In this case, the operation is not finalized.

10.7.2.18 TEE.DigestUpdate() void TEE.DigestUpdate (
 TEE.OperationHandle operation,
 const void * chunk,
 uint32_t chunkSize)

Crypto, Message Digest Functions.

Function accumulates message data for hashing.

[TEE.DigestUpdate\(\)](#) - Accumulates message data for hashing.

This function describes the message does not have to be block aligned. Subsequent calls to this function are possible. The operation may be in either initial or active state and becomes active.

Parameters

<i>operation</i>	Handle of a running Message Digest operation.
<i>chunk</i>	Chunk of data to be hashed
<i>chunkSize</i>	size of the chunk.

10.7.2.19 TEE.FreeOperation() void TEE.FreeOperation (
 TEE.OperationHandle operation)

Crypto, for all Crypto Functions.

All Crypto Functions use TEE.OperationHandle* operation instances.

Destroy Crypto instance.

[TEE.FreeOperation\(\)](#) - Deallocates all resources associated with an operation handle.

This function deallocates all resources associated with an operation handle. After this function is called, the operation handle is no longer valid. All cryptographic material in the operation is destroyed. The function does nothing if operation is TEE_HANDLE_NULL.

Parameters

<i>operation</i>	Reference to operation handle.
------------------	--------------------------------

Returns

nothing after the operation free.

10.7.2.20 TEE_FreeTransientObject() `void TEE_FreeTransientObject (TEE_ObjectHandle object)`

Crypto, Asymmetric key Verification Functions.

Destroy object storing asymmetric key.

[TEE_FreeTransientObject\(\)](#) - Deallocates a transient object previously allocated with [TEE_AllocateTransientObject](#) .

this function describes the object handle is no longer valid and all resources associated with the transient object shall have been reclaimed after the [TEE_AllocateTransientObject\(\)](#) call.

Parameters

<i>object</i>	Handle on the object to free.
---------------	-------------------------------

10.7.2.21 TEE_GenerateKey() `TEE_Result TEE_GenerateKey (TEE_ObjectHandle object, uint32_t keySize, const TEE_Attribute * params, uint32_t paramCount)`

Crypto, Asymmetric key Verification Functions.

Generate asymmetric keypair.

[TEE_GenerateKey \(\)](#) - Generates a random key or a key-pair and populates a transient key object with the generated key material.

The size of the desired key is passed in the keySize parameter and shall be less than or equal to the maximum key size specified when the transient object was created.

Parameters

<i>object</i>	Handle on an uninitialized transient key to populate with the generated key.
Paramter list continued on next page	

<i>keySize</i>	Requested key size shall be less than or equal to the maximum key size specified when the object container was created
<i>params</i>	Parameters for the key generation.
<i>paramCount</i>	The values of all parameters are copied into the object so that the params array and all the memory buffers it points to may be freed after this routine returns without affecting the object.

Returns

0 on success

TEE_ERROR_BAD_PARAMETERS If an incorrect or inconsistent attribute is detected. The checks that are performed depend on the implementation.

10.7.2.22 TEE_GenerateRandom() void TEE_GenerateRandom (
 void * *randomBuffer*,
 uint32_t *randomBufferLen*)

Crypto, common.

Random Data Generation Function. The quality of the random is implementation dependent.

I am not sure this should be in Keystone or not, but it is very handy.

Good to have adding a way to check the quality of the random implementation.

[TEE_GenerateRandom\(\)](#) - Generates random data.

This function generates random data of random buffer length and is stored in to random Buffer by calling wc.RNG↔
_GenerateBlock(). If ret is not equal
to 0 then TEE_Panic is called.

Parameters

<i>randomBuffer</i>	Reference to generated random data
<i>randomBufferLen</i>	Byte length of requested random data

Returns

random data random data will be returned.

[TEE_GenerateRandom\(\)](#) - Generates random data.

This function generates random data of random bufferlength and is stored in to randomBuffer by calling sgx.read↔
_rand().

Parameters

<i>randomBuffer</i>	Reference to generated random data
<i>randomBufferLen</i>	Byte length of requested random data

10.7.2.23 TEE_GetObjectInfo1() `TEE_Result TEE_GetObjectInfo1 (`
`TEE_ObjectHandle object,`
`TEE_ObjectInfo * objectInfo)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

Get length of object required before reading the object.

[TEE_GetObjectInfo1\(\)](#) - Returns the characteristics of an object.

This function returns a handle which can be used to access the object's attributes and data stream.

Parameters

<i>objectInfo</i>	Pointer to a structure filled with the object information
<i>object</i>	Handle of the object

Returns

0 if success else error occurred.

[TEE_GetObjectInfo1\(\)](#) - Function returns the characteristics of an object.

It returns a handle that can be used to access the object's attributes and data stream.

Parameters

<i>objectInfo</i>	Pointer to a structure filled with the object information
<i>object</i>	Handle of the object

Returns

0 if success else error occurred.

10.7.2.24 TEE_GetREETime() `void TEE_GetREETime (`
`TEE_Time * time)`

Core Functions, Time Functions.

Wall clock time of host OS, expressed in the number of seconds since 1970-01-01 UTC.
This could be implemented on Keystone using ocall.

[TEE_GetREETime\(\)](#) - Retrieves the current REE system time.

This function retrieves the current time as seen from the point of view of the REE.

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds
-------------	--

[TEE_GetREETime\(\)](#) - Function retrieves the current REE system time.

This function retrieves the current time as seen from the point of view of the REE.

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds.
-------------	---

10.7.2.25 TEE_GetSystemTime() `void TEE_GetSystemTime (`
 `TEE_Time * time)`

Core Functions, Time Functions.

Time of TEE-controlled secure timer or Host OS time, implementation dependent.

[TEE_GetSystemTime\(\)](#) - Retrieves the current system time.

This function describes the system time has an arbitrary implementation defined origin that can vary across TA instances. The minimum guarantee is that the system time shall be monotonic for a given TA instance.

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds
-------------	--

[TEE_GetSystemTime\(\)](#) - Retrieves the current system time.

The system time has an arbitrary implementation-defined origin that can vary across TA instances

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds.
-------------	---

10.7.2.26 TEE_InitRefAttribute() `void TEE_InitRefAttribute (`
 `TEE_Attribute * attr,`
 `uint32_t attributeID,`

```
const void * buffer,
uint32_t length )
```

Crypto, Asymmetric key Verification Functions.

Storing asymmetric key.

[TEE_InitRefAttribute\(\)](#) - The helper function can be used to populate a single attribute either with a reference to a buffer or with integer values.

In `TEE_InitRefAttribute()` only the buffer pointer is copied, not the content of the buffer. This means that the attribute structure maintains a pointer back to the supplied buffer. It is the responsibility of the TA author to ensure that the contents of the buffer maintain their value until the attributes array is no longer in use.

Parameters

<i>attr</i>	attribute structure to initialize.
<i>attributeID</i>	Identifier of the attribute to populate.
<i>buffer</i>	input buffer that holds the content of the attribute.
<i>length</i>	buffer length.

10.7.2.27 TEE_InitValueAttribute() `void TEE_InitValueAttribute (`
`TEE_Attribute * attr,`
`uint32_t attributeID,`
`uint32_t a,`
`uint32_t b)`

Crypto, Asymmetric key Verification Functions.

Storing asymmetric key.

[TEE_InitValueAttribute\(\)](#) - The helper function can be used to populate a single attribute either with a reference to a buffer or with integer values.

Parameters

<i>attr</i>	attribute structure to initialize.
<i>attributeID</i>	Identifier of the attribute to populate.
<i>a</i>	unsigned integer value to assign to the a member of the attribute structure.
<i>b</i>	unsigned integer value to assign to the b member of the attribute structure

10.7.2.28 TEE_OpenPersistentObject() `TEE_Result TEE_OpenPersistentObject (`
`uint32_t storageID,`
`const void * objectID,`
`uint32_t objectIDLen,`

```
uint32_t flags,
TEE_ObjectHandle * object )
```

Core Functions, Secure Storage Functions (data is isolated for each TA)

Open persistent object.

[TEE_OpenPersistentObject\(\)](#) - Opens a handle on an existing persistent object.

This function returns a handle which can be used to access the object's attributes and data stream.

Parameters

<i>storageID</i>	The storage to use
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion

Returns

0 if success else error occurred.

[TEE_OpenPersistentObject\(\)](#) - Opens a handle on an existing persistent object.

This function returns a handle that can be used to access the object's attributes and data stream.

Parameters

<i>storageID</i>	The storage to use.
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion

Returns

0 if success, else error occurred.

10.7.2.29 TEE_ReadObjectData() [TEE_Result](#) TEE_ReadObjectData (
[TEE_ObjectHandle](#) object,
void * buffer,
uint32_t size,
uint32_t * count)

Core Functions, Secure Storage Functions (data is isolated for each TA)

Read object.

[TEE_ReadObjectData\(\)](#) - Attempts to read size bytes from the data stream associated with the object into the buffer pointed to by buffer.

The bytes are read starting at the position in the data stream currently stored in the object handle. The handle's position is incremented by the number of bytes actually read. On completion of TEE_ReadObjectData sets the number of bytes actually read in the "uint32_t" pointed to by count. The value written to *count may be less than size if the number of bytes until the end-of-stream is less than size. It is set to 0 if the position at the start of the read operation is at or beyond the end-of-stream. These are the only cases where *count may be less than size.

Parameters

<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write
<i>count</i>	size of the buffer.

Returns

TEE_SUCCESS if success else error occurred.

[TEE_ReadObjectData\(\)](#) - Attempts to read size bytes from the data stream associated with the object object into the buffer pointed to by buffer.

The bytes are read starting at the position in the data stream currently stored in the object handle. The handle's position is incremented by the number of bytes actually read. On completion TEE_ReadObjectData sets the number of bytes actually read in the uint32_t pointed to by count. The value written to *count may be less than size if the number of bytes until the end-of-stream is less than size. It is set to 0 if the position at the start of the read operation is at or beyond the end-of-stream. These are the only cases where *count may be less than size.

Parameters

<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write
<i>count</i>	size of the buffer.

Returns

TEE_SUCCESS if success, else error occurred.

10.7.2.30 TEE_SetOperationKey() [TEE_Result](#) TEE_SetOperationKey (
[TEE_OperationHandle](#) operation,
[TEE_ObjectHandle](#) key)

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

Set symmetric key used in operation.

[TEE.SetOperationKey\(\)](#) - Programs the key of an operation; that is, it associates an operation with a key.

The key material is copied from the key object handle into the operation. After the key has been set, there is no longer any link between the operation and the key object. The object handle can be closed or reset and this will not affect the operation. This copied material exists until the operation is freed using [TEE.FreeOperation](#) or another key is set into the operation.

Parameters

<i>operation</i>	Operation handle.
<i>key</i>	A handle on a key object.

Returns

0 on success return

[TEE_ERROR_CORRUPT_OBJECT](#) If the object is corrupt. The object handle is closed.

[TEE_ERROR_STORAGE_NOT_AVAILABLE](#) If the persistent object is stored in a storage area which is currently inaccessible.

10.7.2.31 TEE.WriteObjectData() [TEE.Result](#) [TEE.WriteObjectData](#) (
[TEE.ObjectHandle](#) *object*,
 const void * *buffer*,
 uint32_t *size*)

Core Functions, Secure Storage Functions (data is isolated for each TA)

Write object.

[TEE.WriteObjectData\(\)](#) - Writes the buffer data in to persistent objects.

In this function it checks if object is present or not, the encryption/ decryption buffer is taken by calling `mbedtls.aes↔_crypt.cbc()` then that buffer data is encrypted and mapped to object. On the base of object creation [TEE.SUCCESS](#) appears else [TEE.ERROR_GENERIC](#) appears.

Parameters

<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write

Returns

[TEE.SUCCESS](#) if success else error occurred.

[TEE.WriteObjectData\(\)](#) - writes size bytes from the buffer pointed to by buffer to the data stream associated with the open object handle object.

If the current data position points before the end-of-stream, then size bytes are written to the data stream, overwriting bytes starting at the current data position. If the current data position points beyond the stream's end, then the data stream is first extended with zero bytes until the length indicated by the data position indicator is reached, and then size bytes are written to the stream.

Parameters

<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write

Returns

TEE_SUCCESS if success else error occurred.

10.8 tee-ta-internal.h

[Go to the documentation of this file.](#)

```

1 /*
2  * SPDX-License-Identifier: BSD-2-Clause
3  *
4  * Copyright (C) 2019 National Institute of Advanced Industrial Science
5  *                               and Technology (AIST)
6  * All rights reserved.
7  *
8  * Redistribution and use in source and binary forms, with or without
9  * modification, are permitted provided that the following conditions are met:
10  *
11  * 1. Redistributions of source code must retain the above copyright notice,
12  * this list of conditions and the following disclaimer.
13  *
14  * 2. Redistributions in binary form must reproduce the above copyright notice,
15  * this list of conditions and the following disclaimer in the documentation
16  * and/or other materials provided with the distribution.
17  *
18  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
22  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
28  * POSSIBILITY OF SUCH DAMAGE.
29  */
30
31 #ifndef TA_INTERNAL_TEE_H
32 #define TA_INTERNAL_TEE_H
33
34 #include "tee-common.h"
35
36 #ifdef __cplusplus
37 extern "C" {
38 #endif
39
40 void __attribute__((noreturn)) TEE_Panic(unsigned long code);
41
42 void TEE_GetREETime(TEE_Time *time);
43
44 /* Wall clock time is important for verifying certificates. */
45 void TEE_GetSystemTime(TEE_Time *time);
46
47 /* Start timer */
48 TEE_Result GetRelTimeStart(uint64_t start);
49
50 TEE_Result GetRelTimeEnd(uint64_t end);
51
52 #endif

```

```

79
85 TEE_Result TEE_CreatePersistentObject(uint32_t storageID, const void *objectID,
86                                     uint32_t objectIDLen, uint32_t flags,
87                                     TEE_ObjectHandle attributes,
88                                     const void *initialData,
89                                     uint32_t initialDataLen,
90                                     TEE_ObjectHandle *object);
91
92
93 TEE_Result TEE_OpenPersistentObject(uint32_t storageID, const void *objectID,
94                                     uint32_t objectIDLen, uint32_t flags,
95                                     TEE_ObjectHandle *object);
96
97
98 TEE_Result TEE_GetObjectInfo1(TEE_ObjectHandle object, TEE_ObjectInfo *objectInfo);
99
100
101 TEE_Result TEE_WriteObjectData(TEE_ObjectHandle object, const void *buffer,
102                               uint32_t size);
103
104 TEE_Result TEE_ReadObjectData(TEE_ObjectHandle object, void *buffer,
105                               uint32_t size, uint32_t *count);
106
107
108 void TEE_CloseObject(TEE_ObjectHandle object);
109
110
111
112
113 void TEE_GenerateRandom(void *randomBuffer, uint32_t randomBufferLen);
114
115
116
117
118 TEE_Result TEE_AllocateOperation(TEE_OperationHandle *operation,
119                                  uint32_t algorithm, uint32_t mode,
120                                  uint32_t maxKeySize);
121
122
123 void TEE_FreeOperation(TEE_OperationHandle operation);
124
125
126
127
128 void TEE_DigestUpdate(TEE_OperationHandle operation,
129                      const void *chunk, uint32_t chunkSize);
130
131 TEE_Result TEE_DigestDoFinal(TEE_OperationHandle operation, const void *chunk,
132                              uint32_t chunkLen, void *hash, uint32_t *hashLen);
133
134
135
136
137 TEE_Result TEE_SetOperationKey(TEE_OperationHandle operation,
138                                TEE_ObjectHandle key);
139
140
141
142
143 TEE_Result TEE_AEInit(TEE_OperationHandle operation, const void *nonce,
144                      uint32_t nonceLen, uint32_t tagLen, uint32_t AADLen,
145                      uint32_t payloadLen);
146
147
148 TEE_Result TEE_AEUpdate(TEE_OperationHandle operation, const void *srcData,
149                        uint32_t srcLen, void *destData, uint32_t *destLen);
150
151 void TEE_AEUpdateAAD(TEE_OperationHandle operation, const void *AADdata,
152                    uint32_t AADdataLen);
153
154
155 TEE_Result TEE_AEEncryptFinal(TEE_OperationHandle operation,
156                               const void *srcData, uint32_t srcLen,
157                               void *destData, uint32_t *destLen, void *tag,
158                               uint32_t *tagLen);
159
160 TEE_Result TEE_AEDecryptFinal(TEE_OperationHandle operation,
161                               const void *srcData, uint32_t srcLen,
162                               void *destData, uint32_t *destLen, void *tag,
163                               uint32_t tagLen);
164
165
166
167
168 void TEE_CipherInit(TEE_OperationHandle operation, const void *nonce,
169                    uint32_t nonceLen);
170
171 TEE_Result TEE_CipherUpdate(TEE_OperationHandle operation, const void *srcData,
172                             uint32_t srcLen, void *destData, uint32_t *destLen);
173
174
175
176
177 TEE_Result TEE_GenerateKey(TEE_ObjectHandle object, uint32_t keySize,
178                           const TEE_Attribute *params, uint32_t paramCount);
179
180
181
182 TEE_Result TEE_AllocateTransientObject(TEE_ObjectType objectType,
183                                       uint32_t maxKeySize,
184                                       TEE_ObjectHandle *object);
185
186 void TEE_InitRefAttribute(TEE_Attribute *attr, uint32_t attributeID,
187                          const void *buffer, uint32_t length);
188
189 void TEE_InitValueAttribute(TEE_Attribute *attr, uint32_t attributeID,
190                            uint32_t a, uint32_t b);
191
192 void TEE_FreeTransientObject(TEE_ObjectHandle object);
193
194
195
196
197
198
199
200
201
202

```

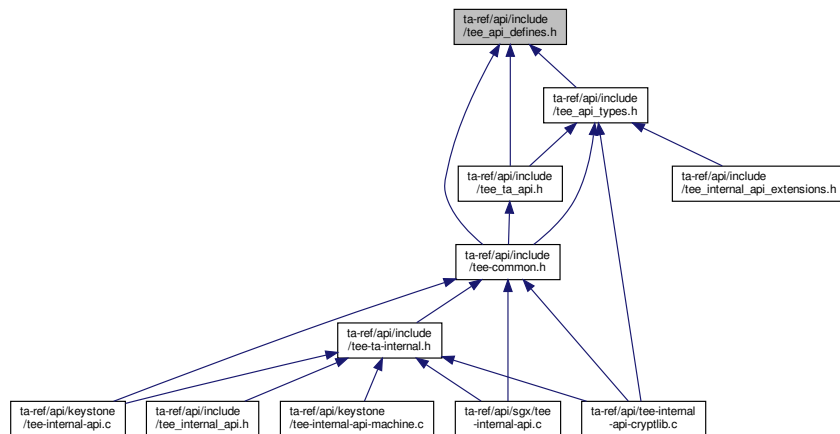
```

206 TEE_Result TEE_AsymmetricSignDigest(TEE_OperationHandle operation,
207                                     const TEE_Attribute *params,
208                                     uint32_t paramCount, const void *digest,
209                                     uint32_t digestLen, void *signature,
210                                     uint32_t *signatureLen);
212
216 TEE_Result TEE_AsymmetricVerifyDigest(TEE_OperationHandle operation,
217                                       const TEE_Attribute *params,
218                                       uint32_t paramCount, const void *digest,
219                                       uint32_t digestLen, const void *signature,
220                                       uint32_t signatureLen);
221
222 #ifdef __cplusplus
223 }
224 #endif
225
226 #endif /* TA_INTERNAL_TEE_H */

```

10.9 ta-ref/api/include/tee_api_defines.h File Reference

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



10.10 tee_api_defines.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25 * POSSIBILITY OF SUCH DAMAGE.

```

```

26  */
27
28 /* Based on GP TEE Internal Core API Specification Version 1.1 */
29
30 #ifndef TEE_API_DEFINES_H
31 #define TEE_API_DEFINES_H
32
33 #ifndef DOXYGEN_SHOULD_SKIP_THIS
34 #define TEE_INT_CORE_API_SPEC_VERSION 0x0000000A
35 #define TEE_HANDLE_NULL 0
36 #define TEE_TIMEOUT_INFINITE 0xFFFFFFFF
37
38 /* API Error Codes */
39 #define TEE_SUCCESS 0x00000000
40 #define TEE_ERROR_CORRUPT_OBJECT 0xF0100001
41 #define TEE_ERROR_CORRUPT_OBJECT_2 0xF0100002
42 #define TEE_ERROR_STORAGE_NOT_AVAILABLE 0xF0100003
43 #define TEE_ERROR_STORAGE_NOT_AVAILABLE_2 0xF0100004
44 #define TEE_ERROR_GENERIC 0xFFFF0000
45 #define TEE_ERROR_ACCESS_DENIED 0xFFFF0001
46 #define TEE_ERROR_CANCEL 0xFFFF0002
47 #define TEE_ERROR_ACCESS_CONFLICT 0xFFFF0003
48 #define TEE_ERROR_EXCESS_DATA 0xFFFF0004
49 #define TEE_ERROR_BAD_FORMAT 0xFFFF0005
50 #define TEE_ERROR_BAD_PARAMETERS 0xFFFF0006
51 #define TEE_ERROR_BAD_STATE 0xFFFF0007
52 #define TEE_ERROR_ITEM_NOT_FOUND 0xFFFF0008
53 #define TEE_ERROR_NOT_IMPLEMENTED 0xFFFF0009
54 #define TEE_ERROR_NOT_SUPPORTED 0xFFFF000A
55 #define TEE_ERROR_NO_DATA 0xFFFF000B
56 #define TEE_ERROR_OUT_OF_MEMORY 0xFFFF000C
57 #define TEE_ERROR_BUSY 0xFFFF000D
58 #define TEE_ERROR_COMMUNICATION 0xFFFF000E
59 #define TEE_ERROR_SECURITY 0xFFFF000F
60 #define TEE_ERROR_SHORT_BUFFER 0xFFFF0010
61 #define TEE_ERROR_EXTERNAL_CANCEL 0xFFFF0011
62 #define TEE_ERROR_OVERFLOW 0xFFFF300F
63 #define TEE_ERROR_TARGET_DEAD 0xFFFF3024
64 #define TEE_ERROR_STORAGE_NO_SPACE 0xFFFF3041
65 #define TEE_ERROR_MAC_INVALID 0xFFFF3071
66 #define TEE_ERROR_SIGNATURE_INVALID 0xFFFF3072
67 #define TEE_ERROR_TIME_NOT_SET 0xFFFF5000
68 #define TEE_ERROR_TIME_NEEDS_RESET 0xFFFF5001
69
70 /* Parameter Type Constants */
71 #define TEE_PARAM_TYPE_NONE 0
72 #define TEE_PARAM_TYPE_VALUE_INPUT 1
73 #define TEE_PARAM_TYPE_VALUE_OUTPUT 2
74 #define TEE_PARAM_TYPE_VALUE_INOUT 3
75 #define TEE_PARAM_TYPE_MEMREF_INPUT 5
76 #define TEE_PARAM_TYPE_MEMREF_OUTPUT 6
77 #define TEE_PARAM_TYPE_MEMREF_INOUT 7
78
79 /* Login Type Constants */
80 #define TEE_LOGIN_PUBLIC 0x00000000
81 #define TEE_LOGIN_USER 0x00000001
82 #define TEE_LOGIN_GROUP 0x00000002
83 #define TEE_LOGIN_APPLICATION 0x00000004
84 #define TEE_LOGIN_APPLICATION_USER 0x00000005
85 #define TEE_LOGIN_APPLICATION_GROUP 0x00000006
86 #define TEE_LOGIN_TRUSTED_APP 0xF0000000
87
88 /* Origin Code Constants */
89 #define TEE_ORIGIN_API 0x00000001
90 #define TEE_ORIGIN_COMMS 0x00000002
91 #define TEE_ORIGIN_TEE 0x00000003
92 #define TEE_ORIGIN_TRUSTED_APP 0x00000004
93
94 /* Property Sets pseudo handles */
95 #define TEE_PROPSET_TEE_IMPLEMENTATION (TEE_PropSetHandle) 0xFFFFFFFFD
96 #define TEE_PROPSET_CURRENT_CLIENT (TEE_PropSetHandle) 0xFFFFFFFFE
97 #define TEE_PROPSET_CURRENT_TA (TEE_PropSetHandle) 0xFFFFFFFFF
98
99 /* Memory Access Rights Constants */
100 #define TEE_MEMORY_ACCESS_READ 0x00000001
101 #define TEE_MEMORY_ACCESS_WRITE 0x00000002
102 #define TEE_MEMORY_ACCESS_ANY_OWNER 0x00000004
103
104 /* Memory Management Constant */
105 #define TEE_MALLOC_FILL_ZERO 0x00000000
106
107 /* Other constants */
108 #define TEE_STORAGE_PRIVATE 0x00000001
109
110 #define TEE_DATA_FLAG_ACCESS_READ 0x00000001
111 #define TEE_DATA_FLAG_ACCESS_WRITE 0x00000002
112 #define TEE_DATA_FLAG_ACCESS_WRITE_META 0x00000004

```

```

113 #define TEE_DATA_FLAG_SHARE_READ 0x00000010
114 #define TEE_DATA_FLAG_SHARE_WRITE 0x00000020
115 #define TEE_DATA_FLAG_OVERWRITE 0x00000400
116 #define TEE_DATA_MAX_POSITION 0xFFFFFFFF
117 #define TEE_OBJECT_ID_MAX_LEN 64
118 #define TEE_USAGE_EXTRACTABLE 0x00000001
119 #define TEE_USAGE_ENCRYPT 0x00000002
120 #define TEE_USAGE_DECRYPT 0x00000004
121 #define TEE_USAGE_MAC 0x00000008
122 #define TEE_USAGE_SIGN 0x00000010
123 #define TEE_USAGE_VERIFY 0x00000020
124 #define TEE_USAGE_DERIVE 0x00000040
125 #define TEE_HANDLE_FLAG_PERSISTENT 0x00010000
126 #define TEE_HANDLE_FLAG_INITIALIZED 0x00020000
127 #define TEE_HANDLE_FLAG_KEY_SET 0x00040000
128 #define TEE_HANDLE_FLAG_EXPECT_TWO_KEYS 0x00080000
129 #define TEE_OPERATION_CIPHER 1
130 #define TEE_OPERATION_MAC 3
131 #define TEE_OPERATION_AE 4
132 #define TEE_OPERATION_DIGEST 5
133 #define TEE_OPERATION_ASYMMETRIC_CIPHER 6
134 #define TEE_OPERATION_ASYMMETRIC_SIGNATURE 7
135 #define TEE_OPERATION_KEY_DERIVATION 8
136 #define TEE_OPERATION_STATE_INITIAL 0x00000000
137 #define TEE_OPERATION_STATE_ACTIVE 0x00000001
138
139 /* Algorithm Identifiers */
140 #define TEE_ALG_AES_ECB_NOPAD 0x10000010
141 #define TEE_ALG_AES_CBC_NOPAD 0x10000110
142 #define TEE_ALG_AES_CTR 0x10000210
143 #define TEE_ALG_AES_CTS 0x10000310
144 #define TEE_ALG_AES_XTS 0x10000410
145 #define TEE_ALG_AES_CBC_MAC_NOPAD 0x30000110
146 #define TEE_ALG_AES_CBC_MAC_PKCS5 0x30000510
147 #define TEE_ALG_AES_CMAC 0x30000610
148 #define TEE_ALG_AES_CCM 0x40000710
149 #define TEE_ALG_AES_GCM 0x40000810
150 #define TEE_ALG_DES_ECB_NOPAD 0x10000011
151 #define TEE_ALG_DES_CBC_NOPAD 0x10000111
152 #define TEE_ALG_DES_CBC_MAC_NOPAD 0x30000111
153 #define TEE_ALG_DES_CBC_MAC_PKCS5 0x30000511
154 #define TEE_ALG_DES3_ECB_NOPAD 0x10000013
155 #define TEE_ALG_DES3_CBC_NOPAD 0x10000113
156 #define TEE_ALG_DES3_CBC_MAC_NOPAD 0x30000113
157 #define TEE_ALG_DES3_CBC_MAC_PKCS5 0x30000513
158 #define TEE_ALG_RSASSA_PKCS1_V1_5_MD5 0x70001830
159 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA1 0x70002830
160 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA224 0x70003830
161 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA256 0x70004830
162 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA384 0x70005830
163 #define TEE_ALG_RSASSA_PKCS1_V1_5_SHA512 0x70006830
164 #define TEE_ALG_RSASSA_PKCS1_V1_5_MD5_SHA1 0x7000F830
165 #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1 0x70212930
166 #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA224 0x70313930
167 #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256 0x70414930
168 #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA384 0x70515930
169 #define TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA512 0x70616930
170 #define TEE_ALG_RSAES_PKCS1_V1_5 0x60000130
171 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1 0x60210230
172 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224 0x60310230
173 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256 0x60410230
174 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384 0x60510230
175 #define TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512 0x60610230
176 #define TEE_ALG_RSA_NOPAD 0x60000030
177 #define TEE_ALG_DSA_SHA1 0x70002131
178 #define TEE_ALG_DSA_SHA224 0x70003131
179 #define TEE_ALG_DSA_SHA256 0x70004131
180 #define TEE_ALG_DH_DERIVE_SHARED_SECRET 0x80000032
181 #define TEE_ALG_MD5 0x50000001
182 #define TEE_ALG_SHA1 0x50000002
183 #define TEE_ALG_SHA224 0x50000003
184 #define TEE_ALG_SHA256 0x50000004
185 #define TEE_ALG_SHA384 0x50000005
186 #define TEE_ALG_SHA512 0x50000006
187 #define TEE_ALG_MD5_SHA1 0x5000000F
188 #define TEE_ALG_HMAC_MD5 0x30000001
189 #define TEE_ALG_HMAC_SHA1 0x30000002
190 #define TEE_ALG_HMAC_SHA224 0x30000003
191 #define TEE_ALG_HMAC_SHA256 0x30000004
192 #define TEE_ALG_HMAC_SHA384 0x30000005
193 #define TEE_ALG_HMAC_SHA512 0x30000006
194 /*
195  * Fix GP Internal Core API v1.1
196  * "Table 6-12: Structure of Algorithm Identifier"
197  * indicates ECDSA have the algorithm "0x41" and ECDH "0x42"
198  * whereas
199  * "Table 6-11: List of Algorithm Identifiers" defines

```

```

200 *      TEE.ALG.ECDSA.P192 as 0x70001042
201 *
202 * We chose to define TEE.ALG.ECDSA.P192 as 0x70001041 (conform to table 6-12)
203 */
204 #define TEE.ALG.ECDSA.P192                0x70001041
205 #define TEE.ALG.ECDSA.P224                0x70002041
206 #define TEE.ALG.ECDSA.P256                0x70003041
207 #define TEE.ALG.ECDSA.P384                0x70004041
208 #define TEE.ALG.ECDSA.P521                0x70005041
209 #define TEE.ALG.ECDH.P192                 0x80001042
210 #define TEE.ALG.ECDH.P224                 0x80002042
211 #define TEE.ALG.ECDH.P256                 0x80003042
212 #define TEE.ALG.ECDH.P384                 0x80004042
213 #define TEE.ALG.ECDH.P521                 0x80005042
214
215 /* Object Types */
216
217 #define TEE.TYPE.AES                      0xA0000010
218 #define TEE.TYPE.DES                      0xA0000011
219 #define TEE.TYPE.DES3                     0xA0000013
220 #define TEE.TYPE.HMAC.MD5                 0xA0000001
221 #define TEE.TYPE.HMAC.SHA1                0xA0000002
222 #define TEE.TYPE.HMAC.SHA224              0xA0000003
223 #define TEE.TYPE.HMAC.SHA256              0xA0000004
224 #define TEE.TYPE.HMAC.SHA384              0xA0000005
225 #define TEE.TYPE.HMAC.SHA512              0xA0000006
226 #define TEE.TYPE.RSA.PUBLIC.KEY            0xA0000030
227 #define TEE.TYPE.RSA.KEYPAIR              0xA1000030
228 #define TEE.TYPE.DSA.PUBLIC.KEY            0xA0000031
229 #define TEE.TYPE.DSA.KEYPAIR              0xA1000031
230 #define TEE.TYPE.DH.KEYPAIR               0xA1000032
231 #define TEE.TYPE.ECDSA.PUBLIC.KEY          0xA0000041
232 #define TEE.TYPE.ECDSA.KEYPAIR            0xA1000041
233 #define TEE.TYPE.ECDH.PUBLIC.KEY           0xA0000042
234 #define TEE.TYPE.ECDH.KEYPAIR             0xA1000042
235 #define TEE.TYPE.GENERIC.SECRET           0xA0000000
236 #define TEE.TYPE.CORRUPTED.OBJECT         0xA00000BE
237 #define TEE.TYPE.DATA                     0xA00000BF
238
239 /* List of Object or Operation Attributes */
240
241 #define TEE.ATTR.SECRET.VALUE              0xC0000000
242 #define TEE.ATTR.RSA.MODULUS               0xD0000130
243 #define TEE.ATTR.RSA.PUBLIC.EXPONENT        0xD0000230
244 #define TEE.ATTR.RSA.PRIVATE.EXPONENT      0xC0000330
245 #define TEE.ATTR.RSA.PRIME1                0xC0000430
246 #define TEE.ATTR.RSA.PRIME2                0xC0000530
247 #define TEE.ATTR.RSA.EXPONENT1             0xC0000630
248 #define TEE.ATTR.RSA.EXPONENT2             0xC0000730
249 #define TEE.ATTR.RSA.COEFFICIENT           0xC0000830
250 #define TEE.ATTR.DSA.PRIME                 0xD0001031
251 #define TEE.ATTR.DSA.SUBPRIME              0xD0001131
252 #define TEE.ATTR.DSA.BASE                  0xD0001231
253 #define TEE.ATTR.DSA.PUBLIC.VALUE           0xD0000131
254 #define TEE.ATTR.DSA.PRIVATE.VALUE         0xC0000231
255 #define TEE.ATTR.DH.PRIME                  0xD0001032
256 #define TEE.ATTR.DH.SUBPRIME               0xD0001132
257 #define TEE.ATTR.DH.BASE                   0xD0001232
258 #define TEE.ATTR.DH.X.BITS                 0xF0001332
259 #define TEE.ATTR.DH.PUBLIC.VALUE            0xD0000132
260 #define TEE.ATTR.DH.PRIVATE.VALUE          0xC0000232
261 #define TEE.ATTR.RSA.OAEP.LABEL            0xD0000930
262 #define TEE.ATTR.RSA.PSS.SALT.LENGTH       0xF0000A30
263 #define TEE.ATTR.ECC.PUBLIC.VALUE.X         0xD0000141
264 #define TEE.ATTR.ECC.PUBLIC.VALUE.Y         0xD0000241
265 #define TEE.ATTR.ECC.PRIVATE.VALUE         0xC0000341
266 #define TEE.ATTR.ECC.CURVE                 0xF0000441
267
268 #define TEE.ATTR.BIT.PROTECTED              (1 << 28)
269 #define TEE.ATTR.BIT.VALUE                  (1 << 29)
270
271 /* List of Supported ECC Curves */
272 #define TEE.ECC.CURVE.NIST.P192            0x00000001
273 #define TEE.ECC.CURVE.NIST.P224            0x00000002
274 #define TEE.ECC.CURVE.NIST.P256            0x00000003
275 #define TEE.ECC.CURVE.NIST.P384            0x00000004
276 #define TEE.ECC.CURVE.NIST.P521            0x00000005
277
278
279 /* Panicked Functions Identification */
280 /* TA Interface */
281 #define TEE.PANIC.ID.TA.CLOSESESSIONENTRYPOINT 0x00000101
282 #define TEE.PANIC.ID.TA.CREATEENTRYPOINT      0x00000102
283 #define TEE.PANIC.ID.TA.DESTROYENTRYPOINT      0x00000103
284 #define TEE.PANIC.ID.TA.INVOKECOMMANDENTRYPOINT 0x00000104
285 #define TEE.PANIC.ID.TA.OPENSESSIONENTRYPOINT 0x00000105
286 /* Property Access */

```



```

287 #define TEE_PANIC_ID_TEE_ALLOCATEPROPERTYENUMERATOR 0x00000201
288 #define TEE_PANIC_ID_TEE_FREEPROPERTYENUMERATOR 0x00000202
289 #define TEE_PANIC_ID_TEE_GETNEXTPROPERTY 0x00000203
290 #define TEE_PANIC_ID_TEE_GETTPROPERTYASBINARYBLOCK 0x00000204
291 #define TEE_PANIC_ID_TEE_GETTPROPERTYASBOOL 0x00000205
292 #define TEE_PANIC_ID_TEE_GETTPROPERTYASIDENTITY 0x00000206
293 #define TEE_PANIC_ID_TEE_GETTPROPERTYASSTRING 0x00000207
294 #define TEE_PANIC_ID_TEE_GETTPROPERTYASU32 0x00000208
295 #define TEE_PANIC_ID_TEE_GETTPROPERTYASUUID 0x00000209
296 #define TEE_PANIC_ID_TEE_GETTPROPERTYNAME 0x0000020A
297 #define TEE_PANIC_ID_TEE_RESETPROPERTYENUMERATOR 0x0000020B
298 #define TEE_PANIC_ID_TEE_STARTTPROPERTYENUMERATOR 0x0000020C
299 /* Panic Function */
300 #define TEE_PANIC_ID_TEE_PANIC 0x00000301
301 /* Internal Client API */
302 #define TEE_PANIC_ID_TEE_CLOSETASESSION 0x00000401
303 #define TEE_PANIC_ID_TEE_INVOKETACOMMAND 0x00000402
304 #define TEE_PANIC_ID_TEE_OPENTASESSION 0x00000403
305 /* Cancellation */
306 #define TEE_PANIC_ID_TEE_GETCANCELLATIONFLAG 0x00000501
307 #define TEE_PANIC_ID_TEE_MASKCANCELLATION 0x00000502
308 #define TEE_PANIC_ID_TEE_UNMASKCANCELLATION 0x00000503
309 /* Memory Management */
310 #define TEE_PANIC_ID_TEE_CHECKMEMORYACCESSRIGHTS 0x00000601
311 #define TEE_PANIC_ID_TEE_FREE 0x00000602
312 #define TEE_PANIC_ID_TEE_GETINSTANCEDATA 0x00000603
313 #define TEE_PANIC_ID_TEE_MALLOC 0x00000604
314 #define TEE_PANIC_ID_TEE_MEMCOMPARE 0x00000605
315 #define TEE_PANIC_ID_TEE_MEMFILL 0x00000606
316 #define TEE_PANIC_ID_TEE_MEMMOVE 0x00000607
317 #define TEE_PANIC_ID_TEE_REALLOC 0x00000608
318 #define TEE_PANIC_ID_TEE_SETINSTANCEDATA 0x00000609
319 /* Generic Object */
320 #define TEE_PANIC_ID_TEE_CLOSEOBJECT 0x00000701
321 #define TEE_PANIC_ID_TEE_GETOBJECTBUFFERATTRIBUTE 0x00000702
322 /* deprecated */
323 #define TEE_PANIC_ID_TEE_GETOBJECTINFO 0x00000703
324 #define TEE_PANIC_ID_TEE_GETOBJECTVALUEATTRIBUTE 0x00000704
325 /* deprecated */
326 #define TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE 0x00000705
327 #define TEE_PANIC_ID_TEE_GETOBJECTINFO1 0x00000706
328 #define TEE_PANIC_ID_TEE_RESTRICTOBJECTUSAGE1 0x00000707
329 /* Transient Object */
330 #define TEE_PANIC_ID_TEE_ALLOCATETRANSIENTOBJECT 0x00000801
331 /* deprecated */
332 #define TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES 0x00000802
333 #define TEE_PANIC_ID_TEE_FREETRANSIENTOBJECT 0x00000803
334 #define TEE_PANIC_ID_TEE_GENERATEKEY 0x00000804
335 #define TEE_PANIC_ID_TEE_INITREFATTRIBUTE 0x00000805
336 #define TEE_PANIC_ID_TEE_INITVALUEATTRIBUTE 0x00000806
337 #define TEE_PANIC_ID_TEE_POPULATETRANSIENTOBJECT 0x00000807
338 #define TEE_PANIC_ID_TEE_RESETTRANSIENTOBJECT 0x00000808
339 #define TEE_PANIC_ID_TEE_COPYOBJECTATTRIBUTES1 0x00000809
340 /* Persistent Object */
341 /* deprecated */
342 #define TEE_PANIC_ID_TEE_CLOSEANDDELETETERSISTENTOBJECT 0x00000901
343 #define TEE_PANIC_ID_TEE_CREATEPERSISTENTOBJECT 0x00000902
344 #define TEE_PANIC_ID_TEE_OPENPERSISTENTOBJECT 0x00000903
345 #define TEE_PANIC_ID_TEE_RENAMEPERSISTENTOBJECT 0x00000904
346 #define TEE_PANIC_ID_TEE_CLOSEANDDELETETERSISTENTOBJECT1 0x00000905
347 /* Persistent Object Enumeration */
348 #define TEE_PANIC_ID_TEE_ALLOCATEPERSISTENTOBJECTENUMERATOR 0x00000A01
349 #define TEE_PANIC_ID_TEE_FREEPERSISTENTOBJECTENUMERATOR 0x00000A02
350 #define TEE_PANIC_ID_TEE_GETNEXTPERSISTENTOBJECT 0x00000A03
351 #define TEE_PANIC_ID_TEE_RESETPERSISTENTOBJECTENUMERATOR 0x00000A04
352 #define TEE_PANIC_ID_TEE_STARTPERSISTENTOBJECTENUMERATOR 0x00000A05
353 /* Data Stream Access */
354 #define TEE_PANIC_ID_TEE_READOBJECTDATA 0x00000B01
355 #define TEE_PANIC_ID_TEE_SEEKOBJECTDATA 0x00000B02
356 #define TEE_PANIC_ID_TEE_TRUNCATEOBJECTDATA 0x00000B03
357 #define TEE_PANIC_ID_TEE_WRITEOBJECTDATA 0x00000B04
358 /* Generic Operation */
359 #define TEE_PANIC_ID_TEE_ALLOCATEOPERATION 0x00000C01
360 #define TEE_PANIC_ID_TEE_COPYOPERATION 0x00000C02
361 #define TEE_PANIC_ID_TEE_FREEOPERATION 0x00000C03
362 #define TEE_PANIC_ID_TEE_GETOPERATIONINFO 0x00000C04
363 #define TEE_PANIC_ID_TEE_RESETOPERATION 0x00000C05
364 #define TEE_PANIC_ID_TEE_SETOPERATIONKEY 0x00000C06
365 #define TEE_PANIC_ID_TEE_SETOPERATIONKEY2 0x00000C07
366 #define TEE_PANIC_ID_TEE_GETOPERATIONINFORMULTIPLE 0x00000C08
367 /* Message Digest */
368 #define TEE_PANIC_ID_TEE_DIGESTDOFINAL 0x00000D01
369 #define TEE_PANIC_ID_TEE_DIGESTUPDATE 0x00000D02
370 /* Symmetric Cipher */
371 #define TEE_PANIC_ID_TEE_CIPHERDOFINAL 0x00000E01
372 #define TEE_PANIC_ID_TEE_CIPHERINIT 0x00000E02
373 #define TEE_PANIC_ID_TEE_CIPHERUPDATE 0x00000E03

```

```

374 /* MAC */
375 #define TEE_PANIC_ID_TEE_MACCOMPAREFINAL 0x00000F01
376 #define TEE_PANIC_ID_TEE_MACCOMPUTEFINAL 0x00000F02
377 #define TEE_PANIC_ID_TEE_MACINIT 0x00000F03
378 #define TEE_PANIC_ID_TEE_MACUPDATE 0x00000F04
379 /* Authenticated Encryption */
380 #define TEE_PANIC_ID_TEE_AEDECRYPTFINAL 0x00001001
381 #define TEE_PANIC_ID_TEE_AEENCRYPTFINAL 0x00001002
382 #define TEE_PANIC_ID_TEE_AEINIT 0x00001003
383 #define TEE_PANIC_ID_TEE_AEUPDATE 0x00001004
384 #define TEE_PANIC_ID_TEE_AEUPDATEAAD 0x00001005
385 /* Asymmetric */
386 #define TEE_PANIC_ID_TEE_ASYMMETRICDECRYPT 0x00001101
387 #define TEE_PANIC_ID_TEE_ASYMMETRICENCRYPT 0x00001102
388 #define TEE_PANIC_ID_TEE_ASYMMETRICSIGNDIGEST 0x00001103
389 #define TEE_PANIC_ID_TEE_ASYMMETRICVERIFYDIGEST 0x00001104
390 /* Key Derivation */
391 #define TEE_PANIC_ID_TEE_DERIVEKEY 0x00001201
392 /* Random Data Generation */
393 #define TEE_PANIC_ID_TEE_GENERATERANDOM 0x00001301
394 /* Time */
395 #define TEE_PANIC_ID_TEE_GETREETIME 0x00001401
396 #define TEE_PANIC_ID_TEE_GETSYSTEMTIME 0x00001402
397 #define TEE_PANIC_ID_TEE_GETTAPERSISTENTTIME 0x00001403
398 #define TEE_PANIC_ID_TEE_SETTAPERSISTENTTIME 0x00001404
399 #define TEE_PANIC_ID_TEE_WAIT 0x00001405
400 /* Memory Allocation and Size of Objects */
401 #define TEE_PANIC_ID_TEE_BIGINTFMMCONTEXTSIZEINU32 0x00001501
402 #define TEE_PANIC_ID_TEE_BIGINTFMMSIZEINU32 0x00001502
403 /* Initialization */
404 #define TEE_PANIC_ID_TEE_BIGINTINIT 0x00001601
405 #define TEE_PANIC_ID_TEE_BIGINTINITFMM 0x00001602
406 #define TEE_PANIC_ID_TEE_BIGINTINITFMMCONTEXT 0x00001603
407 /* Converter */
408 #define TEE_PANIC_ID_TEE_BIGINTCONVERTFROMOCTETSTRING 0x00001701
409 #define TEE_PANIC_ID_TEE_BIGINTCONVERTFROMS32 0x00001702
410 #define TEE_PANIC_ID_TEE_BIGINTCONVERTTOOCTETSTRING 0x00001703
411 #define TEE_PANIC_ID_TEE_BIGINTCONVERTTOS32 0x00001704
412 /* Logical Operation */
413 #define TEE_PANIC_ID_TEE_BIGINTCMP 0x00001801
414 #define TEE_PANIC_ID_TEE_BIGINTCMPS32 0x00001802
415 #define TEE_PANIC_ID_TEE_BIGINTGETBIT 0x00001803
416 #define TEE_PANIC_ID_TEE_BIGINTGETBITCOUNT 0x00001804
417 #define TEE_PANIC_ID_TEE_BIGINTSHIFTRIGHT 0x00001805
418 /* Basic Arithmetic */
419 #define TEE_PANIC_ID_TEE_BIGINTADD 0x00001901
420 #define TEE_PANIC_ID_TEE_BIGINTDIV 0x00001902
421 #define TEE_PANIC_ID_TEE_BIGINTMUL 0x00001903
422 #define TEE_PANIC_ID_TEE_BIGINTNEG 0x00001904
423 #define TEE_PANIC_ID_TEE_BIGINTSQUARE 0x00001905
424 #define TEE_PANIC_ID_TEE_BIGINTSUB 0x00001906
425 /* Modular Arithmetic */
426 #define TEE_PANIC_ID_TEE_BIGINTADDMOD 0x00001A01
427 #define TEE_PANIC_ID_TEE_BIGINTINVMOD 0x00001A02
428 #define TEE_PANIC_ID_TEE_BIGINTMOD 0x00001A03
429 #define TEE_PANIC_ID_TEE_BIGINTMULMOD 0x00001A04
430 #define TEE_PANIC_ID_TEE_BIGINTSQUAREMOD 0x00001A05
431 #define TEE_PANIC_ID_TEE_BIGINTSUBMOD 0x00001A06
432 /* Other Arithmetic */
433 #define TEE_PANIC_ID_TEE_BIGINTCOMPUTEEXTENDEDGCD 0x00001B01
434 #define TEE_PANIC_ID_TEE_BIGINTISPROBABLEPRIME 0x00001B02
435 #define TEE_PANIC_ID_TEE_BIGINTRELATIVEPRIME 0x00001B03
436 /* Fast Modular Multiplication */
437 #define TEE_PANIC_ID_TEE_BIGINTCOMPUTEFCMM 0x00001C01
438 #define TEE_PANIC_ID_TEE_BIGINTCONVERTFROMFCMM 0x00001C02
439 #define TEE_PANIC_ID_TEE_BIGINTCONVERTTOFCMM 0x00001C03
440
441 /*
442 * The macro TEE_PARAM_TYPES can be used to construct a value that you can
443 * compare against an incoming paramTypes to check the type of all the
444 * parameters in one comparison, like in the following example:
445 * if (paramTypes != TEE_PARAM_TYPES(TEE_PARAM_TYPE_MEMREF_INPUT,
446 *                                     TEE_PARAM_TYPE_MEMREF_OUTPUT,
447 *                                     TEE_PARAM_TYPE_NONE, TEE_PARAM_TYPE_NONE)) {
448 *     return TEE_ERROR_BAD_PARAMETERS;
449 * }
450 */
451 #define TEE_PARAM_TYPES(t0,t1,t2,t3) \
452 ((t0) | ((t1) << 4) | ((t2) << 8) | ((t3) << 12))
453
454 /*
455 * The macro TEE_PARAM_TYPE_GET can be used to extract the type of a given
456 * parameter from paramTypes if you need more fine-grained type checking.
457 */
458 #define TEE_PARAM_TYPE_GET(t, i) (((uint32_t)t) >> ((i)*4)) & 0xF
459
460 /*

```

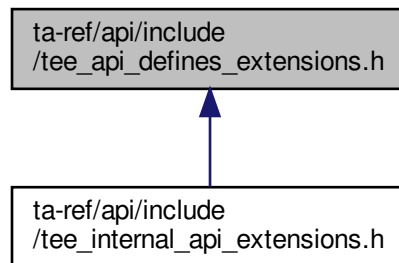
```

461  * The macro TEE_PARAM_TYPE_SET can be used to load the type of a given
462  * parameter from paramTypes without specifying all types (TEE_PARAM_TYPES)
463  */
464  #define TEE_PARAM_TYPE_SET(t, i) (((uint32_t)(t) & 0xF) << ((i)*4))
465
466  /* Not specified in the standard */
467  #define TEE_NUM_PARAMS 4
468
469  /* TEE Arithmetical APIs */
470
471  #define TEE_BigIntSizeInU32(n) (((n)+31)/32)+2
472
473  #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
474  #endif /* TEE_API_DEFINES_H */

```

10.11 ta-ref/api/include/tee_api_defines_extensions.h File Reference

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



10.12 tee_api_defines_extensions.h

[Go to the documentation of this file.](#)

```

1  /*
2  * Copyright (c) 2014, Linaro Limited
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25 * POSSIBILITY OF SUCH DAMAGE.
26 */
27
28 #ifndef TEE_API_DEFINES_EXTENSIONS_H
29 #define TEE_API_DEFINES_EXTENSIONS_H
30 #ifndef DOXYGEN_SHOULD_SKIP_THIS

```

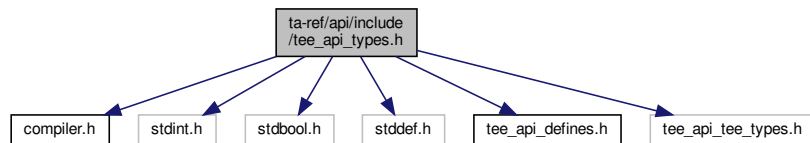
```

31
32 /*
33  * HMAC-based Extract-and-Expand Key Derivation Function (HKDF)
34  */
35
36 #define TEE_ALG_HKDF_MD5_DERIVE_KEY      0x800010C0
37 #define TEE_ALG_HKDF_SHA1_DERIVE_KEY     0x800020C0
38 #define TEE_ALG_HKDF_SHA224_DERIVE_KEY   0x800030C0
39 #define TEE_ALG_HKDF_SHA256_DERIVE_KEY   0x800040C0
40 #define TEE_ALG_HKDF_SHA384_DERIVE_KEY   0x800050C0
41 #define TEE_ALG_HKDF_SHA512_DERIVE_KEY   0x800060C0
42
43 #define TEE_TYPE_HKDF_IKM                 0xA10000C0
44
45 #define TEE_ATTR_HKDF_IKM                 0xC00001C0
46 #define TEE_ATTR_HKDF_SALT                0xD00002C0
47 #define TEE_ATTR_HKDF_INFO                0xD00003C0
48 #define TEE_ATTR_HKDF_OKM_LENGTH          0xF00004C0
49
50 /*
51  * Concatenation Key Derivation Function (Concat KDF)
52  * NIST SP 800-56A section 5.8.1
53  */
54
55 #define TEE_ALG_CONCAT_KDF_SHA1_DERIVE_KEY 0x800020C1
56 #define TEE_ALG_CONCAT_KDF_SHA224_DERIVE_KEY 0x800030C1
57 #define TEE_ALG_CONCAT_KDF_SHA256_DERIVE_KEY 0x800040C1
58 #define TEE_ALG_CONCAT_KDF_SHA384_DERIVE_KEY 0x800050C1
59 #define TEE_ALG_CONCAT_KDF_SHA512_DERIVE_KEY 0x800060C1
60
61 #define TEE_TYPE_CONCAT_KDF_Z              0xA10000C1
62
63 #define TEE_ATTR_CONCAT_KDF_Z              0xC00001C1
64 #define TEE_ATTR_CONCAT_KDF_OTHER_INFO    0xD00002C1
65 #define TEE_ATTR_CONCAT_KDF_DKM_LENGTH    0xF00003C1
66
67 /*
68  * PKCS #5 v2.0 Key Derivation Function 2 (PBKDF2)
69  * RFC 2898 section 5.2
70  * https://www.ietf.org/rfc/rfc2898.txt
71  */
72
73 #define TEE_ALG_PBKDF2_HMAC_SHA1_DERIVE_KEY 0x800020C2
74
75 #define TEE_TYPE_PBKDF2_PASSWORD           0xA10000C2
76
77 #define TEE_ATTR_PBKDF2_PASSWORD          0xC00001C2
78 #define TEE_ATTR_PBKDF2_SALT              0xD00002C2
79 #define TEE_ATTR_PBKDF2_ITERATION_COUNT   0xF00003C2
80 #define TEE_ATTR_PBKDF2_DKM_LENGTH        0xF00004C2
81
82 /*
83  * Implementation-specific object storage constants
84  */
85
86 /* Storage is provided by the Rich Execution Environment (REE) */
87 #define TEE_STORAGE_PRIVATE_REE            0x80000000
88 /* Storage is the Replay Protected Memory Block partition of an eMMC device */
89 #define TEE_STORAGE_PRIVATE_RPMB           0x80000100
90 /* Was TEE_STORAGE_PRIVATE_SQL, which isn't supported any longer */
91 #define TEE_STORAGE_PRIVATE_SQL_RESERVED   0x80000200
92
93 /*
94  * Extension of "Memory Access Rights Constants"
95  * #define TEE_MEMORY_ACCESS_READ          0x00000001
96  * #define TEE_MEMORY_ACCESS_WRITE         0x00000002
97  * #define TEE_MEMORY_ACCESS_ANY_OWNER     0x00000004
98  *
99  * TEE_MEMORY_ACCESS_NONSECURE : if set TEE_CheckMemoryAccessRights()
100  * successfully returns only if target vmem range is mapped non-secure.
101  *
102  * TEE_MEMORY_ACCESS_SECURE : if set TEE_CheckMemoryAccessRights()
103  * successfully returns only if target vmem range is mapped secure.
104  */
105 /*
106 #define TEE_MEMORY_ACCESS_NONSECURE        0x10000000
107 #define TEE_MEMORY_ACCESS_SECURE          0x20000000
108
109 #endif /* DOXYGEN_SHOULD_SKIP_THIS */
110 #endif /* TEE_API_DEFINES_EXTENSIONS_H */

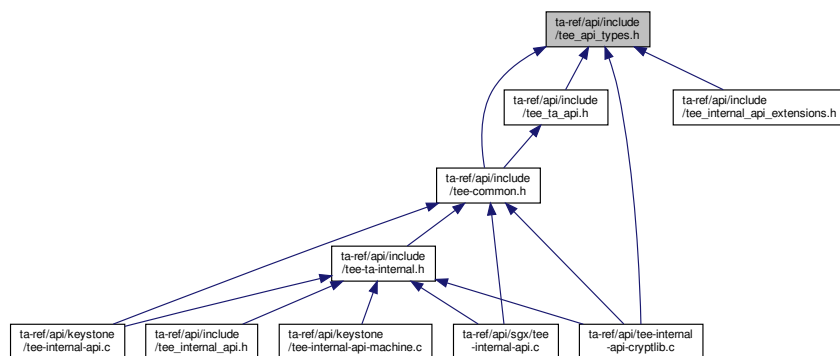
```

10.13 ta-ref/api/include/tee_api_types.h File Reference

```
#include <compiler.h>
#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
#include <tee_api_defines.h>
#include "tee_api_tee_types.h"
Include dependency graph for tee_api_types.h:
```



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



Classes

- struct [TEE_UUID](#)
- struct [TEE_Identity](#)
- union [TEE_Param](#)
- struct [TEE_ObjectInfo](#)
- struct [TEE_Attribute](#)
- struct [TEE_OperationInfo](#)
- struct [TEE_OperationInfoKey](#)
- struct [TEE_OperationInfoMultiple](#)
- struct [TEE_Time](#)
- struct [TEE_SEReaderProperties](#)
- struct [TEE_SEAID](#)
- struct [pollfd](#)
- struct [addrinfo](#)

Typedefs

- typedef uint32_t [TEE_Result](#)
- typedef struct __TEE_TASessionHandle * [TEE_TASessionHandle](#)
- typedef struct __TEE_PropSetHandle * [TEE_PropSetHandle](#)
- typedef struct __TEE_ObjectHandle * [TEE_ObjectHandle](#)
- typedef struct __TEE_ObjectEnumHandle * [TEE_ObjectEnumHandle](#)
- typedef struct __TEE_OperationHandle * [TEE_OperationHandle](#)
- typedef uint32_t [TEE_ObjectType](#)
- typedef uint32_t [TEE_BigInt](#)
- typedef uint32_t [TEE_BigIntFMM](#)
- typedef uint32_t TEE_BigIntFMMContext [__aligned](#)([__alignof__](#)(void *))
- typedef struct __TEE_SEServiceHandle * [TEE_SEServiceHandle](#)
- typedef struct __TEE_SEReaderHandle * [TEE_SEReaderHandle](#)
- typedef struct __TEE_SESessionHandle * [TEE_SESessionHandle](#)
- typedef struct __TEE_SEChannelHandle * [TEE_SEChannelHandle](#)
- typedef uint32_t [TEE_ErrorOrigin](#)
- typedef void * [TEE_Session](#)
- typedef unsigned long int [nfds_t](#)
- typedef unsigned int [socklen_t](#)

Enumerations

- enum [TEE_Whence](#) { [TEE_DATA_SEEK_SET](#) = 0 , [TEE_DATA_SEEK_CUR](#) = 1 , [TEE_DATA_SEEK_END](#) = 2 }
- enum [TEE_OperationMode](#) {
[TEE_MODE_ENCRYPT](#) = 0 , [TEE_MODE_DECRYPT](#) = 1 , [TEE_MODE_SIGN](#) = 2 , [TEE_MODE_VERIFY](#) = 3 ,
[TEE_MODE_MAC](#) = 4 , [TEE_MODE_DIGEST](#) = 5 , [TEE_MODE_DERIVE](#) = 6 }

10.13.1 Typedef Documentation

10.13.1.1 [__aligned](#) typedef uint32_t TEE_BigIntFMMContext [__aligned](#)([__alignof__](#)(void *))

10.13.1.2 [nfds_t](#) typedef unsigned long int [nfds_t](#)

10.13.1.3 [socklen_t](#) typedef unsigned int [socklen_t](#)

10.13.1.4 [TEE_BigInt](#) typedef uint32_t [TEE_BigInt](#)

10.13.1.5 TEE_BigIntFMM typedef uint32_t [TEE_BigIntFMM](#)

10.13.1.6 TEE_ErrorOrigin typedef uint32_t [TEE_ErrorOrigin](#)

10.13.1.7 TEE_ObjectEnumHandle typedef struct __TEE_ObjectEnumHandle* [TEE_ObjectEnumHandle](#)

10.13.1.8 TEE_ObjectHandle typedef struct __TEE_ObjectHandle* [TEE_ObjectHandle](#)

10.13.1.9 TEE_ObjectType typedef uint32_t [TEE_ObjectType](#)

10.13.1.10 TEE_OperationHandle typedef struct __TEE_OperationHandle* [TEE_OperationHandle](#)

10.13.1.11 TEE_PropSetHandle typedef struct __TEE_PropSetHandle* [TEE_PropSetHandle](#)

10.13.1.12 TEE_Result typedef uint32_t [TEE_Result](#)

10.13.1.13 TEE_SEChannelHandle typedef struct __TEE_SEChannelHandle* [TEE_SEChannelHandle](#)

10.13.1.14 TEE_SEReaderHandle typedef struct __TEE_SEReaderHandle* [TEE_SEReaderHandle](#)

10.13.1.15 TEE_SEServiceHandle typedef struct __TEE_SEServiceHandle* [TEE_SEServiceHandle](#)

10.13.1.16 TEE_SESessionHandle typedef struct __TEE_SESessionHandle* [TEE_SESessionHandle](#)

10.13.1.17 TEE_Session typedef void* [TEE_Session](#)

10.13.1.18 TEE_TASessionHandle typedef struct __TEE_TASessionHandle* [TEE_TASessionHandle](#)

10.13.2 Enumeration Type Documentation

10.13.2.1 TEE_OperationMode enum [TEE_OperationMode](#)

Enumerator

TEE_MODE_ENCRYPT	
TEE_MODE_DECRYPT	
TEE_MODE_SIGN	
TEE_MODE_VERIFY	
TEE_MODE_MAC	
TEE_MODE_DIGEST	
TEE_MODE_DERIVE	

10.13.2.2 TEE_Whence enum [TEE_Whence](#)

Enumerator

TEE_DATA_SEEK_SET	
TEE_DATA_SEEK_CUR	
TEE_DATA_SEEK_END	

10.14 tee_api.types.h

[Go to the documentation of this file.](#)

```
1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10 *
```



```

11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25 * POSSIBILITY OF SUCH DAMAGE.
26 */
27
28 /* Based on GP TEE Internal API Specification Version 0.11 */
29 #ifndef TEE_API_TYPES_H
30 #define TEE_API_TYPES_H
31
32 #include <compiler.h>
33 #include <stdint.h>
34 #include <stdbool.h>
35 #include <stddef.h>
36 #include <tee_api_defines.h>
37 #include "tee_api_tee_types.h"
38
39 /*
40 * Common Definitions
41 */
42
43 typedef uint32_t TEE_Result;
44
45 typedef struct {
46     uint32_t timeLow;
47     uint16_t timeMid;
48     uint16_t timeHiAndVersion;
49     uint8_t clockSeqAndNode[8];
50 } TEE_UUID;
51
52 /*
53 * The TEE.Identity structure defines the full identity of a Client:
54 * - login is one of the TEE.LOGIN_XXX constants
55 * - uuid contains the client UUID or Nil if not applicable
56 */
57 typedef struct {
58     uint32_t login;
59     TEE_UUID uuid;
60 } TEE_Identity;
61
62 /*
63 * This union describes one parameter passed by the Trusted Core Framework
64 * to the entry points TA.OpenSessionEntryPoint or
65 * TA.InvokeCommandEntryPoint or by the TA to the functions
66 * TEE.OpenTASession or TEE.InvokeTACommand.
67 *
68 * Which of the field value or memref to select is determined by the
69 * parameter type specified in the argument paramTypes passed to the entry
70 * point.
71 */
72 typedef union {
73     struct {
74         void *buffer;
75         uint32_t size;
76     } memref;
77     struct {
78         uint32_t a;
79         uint32_t b;
80     } value;
81 } TEE_Param;
82
83 /*
84 * The type of opaque handles on TA Session. These handles are returned by
85 * the function TEE.OpenTASession.
86 */
87 typedef struct __TEE_TASessionHandle *TEE_TASessionHandle;
88
89 /*
90 * The type of opaque handles on property sets or enumerators. These
91 * handles are either one of the pseudo handles TEE.PROPSET_XXX or are
92 * returned by the function TEE.AllocatePropertyEnumerator.
93 */
94 typedef struct __TEE_PropSetHandle *TEE_PropSetHandle;
95
96 typedef struct __TEE_ObjectHandle *TEE_ObjectHandle;
97 typedef struct __TEE_ObjectEnumHandle *TEE_ObjectEnumHandle;

```

```

98 typedef struct __TEEOperationHandle *TEEOperationHandle;
99
100 /*
101  * Storage Definitions
102  */
103
104 typedef uint32_t TEEObjectType;
105
106 typedef struct {
107     uint32_t objectType;
108     __extension__ union {
109         uint32_t keySize; /* used in 1.1 spec */
110         uint32_t objectSize; /* used in 1.1.1 spec */
111     };
112     __extension__ union {
113         uint32_t maxKeySize; /* used in 1.1 spec */
114         uint32_t maxObjectSize; /* used in 1.1.1 spec */
115     };
116     uint32_t objectUsage;
117     uint32_t dataSize;
118     uint32_t dataPosition;
119     uint32_t handleFlags;
120 } TEEObjectInfo;
121
122 typedef enum {
123     TEE_DATA_SEEK_SET = 0,
124     TEE_DATA_SEEK_CUR = 1,
125     TEE_DATA_SEEK_END = 2
126 } TEEWhence;
127
128 typedef struct {
129     uint32_t attributeID;
130     union {
131         struct {
132             void *buffer;
133             uint32_t length;
134         } ref;
135         struct {
136             uint32_t a, b;
137         } value;
138     } content;
139 } TEEAttribute;
140
141 #ifndef DOXYGEN_SHOULD_SKIP_THIS
142 #define DMREQ_FINISH 0
143 #define DMREQ_WRITE 1
144 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
145
146 /* Cryptographic Operations API */
147
148 typedef enum {
149     TEE_MODE_ENCRYPT = 0,
150     TEE_MODE_DECRYPT = 1,
151     TEE_MODE_SIGN = 2,
152     TEE_MODE_VERIFY = 3,
153     TEE_MODE_MAC = 4,
154     TEE_MODE_DIGEST = 5,
155     TEE_MODE_DERIVE = 6
156 } TEEOperationMode;
157
158 typedef struct {
159     uint32_t algorithm;
160     uint32_t operationClass;
161     uint32_t mode;
162     uint32_t digestLength;
163     uint32_t maxKeySize;
164     uint32_t keySize;
165     uint32_t requiredKeyUsage;
166     uint32_t handleState;
167 } TEEOperationInfo;
168
169 typedef struct {
170     uint32_t keySize;
171     uint32_t requiredKeyUsage;
172 } TEEOperationInfoKey;
173
174 typedef struct {
175     uint32_t algorithm;
176     uint32_t operationClass;
177     uint32_t mode;
178     uint32_t digestLength;
179     uint32_t maxKeySize;
180     uint32_t handleState;
181     uint32_t operationState;
182     uint32_t numberOfKeys;
183     TEEOperationInfoKey keyInformation[];
184 } TEEOperationInfoMultiple;

```

```

185
186 /* Time & Date API */
187
188 typedef struct {
189     uint32_t seconds;
190     uint32_t millis;
191 } TEE_Time;
192
193 /* TEE Arithmetical APIs */
194
195 typedef uint32_t TEE_BigInt;
196
197 typedef uint32_t TEE_BigIntFMM;
198
199 typedef uint32_t TEE_BigIntFMMContext __aligned(__alignof__(void *));
200
201 /* Tee Secure Element APIs */
202
203 typedef struct __TEE_SEServiceHandle *TEE_SEServiceHandle;
204 typedef struct __TEE_SEReaderHandle *TEE_SEReaderHandle;
205 typedef struct __TEE_SESessionHandle *TEE_SESessionHandle;
206 typedef struct __TEE_SEChannelHandle *TEE_SEChannelHandle;
207
208 typedef struct {
209     bool sePresent;
210     bool teeOnly;
211     bool selectResponseEnable;
212 } TEE_SEReaderProperties;
213
214 typedef struct {
215     uint8_t *buffer;
216     size_t bufferLen;
217 } TEE_SEAID;
218
219 /* Other definitions */
220 typedef uint32_t TEE_ErrorOrigin;
221 typedef void *TEE_Session;
222
223 #ifndef DOXYGEN_SHOULD_SKIP_THIS
224 #define TEE_MEM_INPUT 0x00000001
225 #define TEE_MEM_OUTPUT 0x00000002
226
227 #define TEE_MEMREF_0_USED 0x00000001
228 #define TEE_MEMREF_1_USED 0x00000002
229 #define TEE_MEMREF_2_USED 0x00000004
230 #define TEE_MEMREF_3_USED 0x00000008
231
232 #define TEE_SE_READER_NAME_MAX 20
233 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
234
235 typedef unsigned long int nfds_t;
236
237 struct pollfd
238 {
239     int fd; /* File descriptor to poll. */
240     short int events; /* Types of events poller cares about. */
241     short int revents; /* Types of events that actually occurred. */
242 };
243
244 typedef unsigned int socklen_t;
245
246 struct addrinfo {
247     int ai_flags;
248     int ai_family;
249     int ai_socktype;
250     int ai_protocol;
251     socklen_t ai_addrlen;
252     struct sockaddr *ai_addr;
253     char *ai_canonname;
254     struct addrinfo *ai_next;
255 };
256
257
258
259 #endif /* TEE_API_TYPES_H */

```

10.15 ta-ref/api/include/tee_client_api.h File Reference

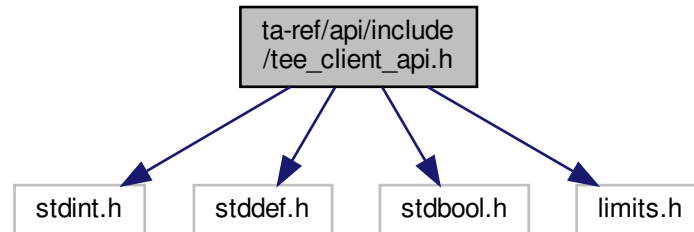
```

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

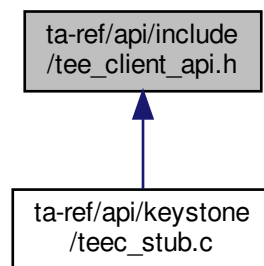
```

```
#include <limits.h>
```

Include dependency graph for tee_client_api.h:



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



Classes

- struct [TEEC_Context](#)
- struct [TEEC_UUID](#)
- struct [TEEC_SharedMemory](#)
- struct [TEEC_TempMemoryReference](#)
- struct [TEEC_RegisteredMemoryReference](#)
- struct [TEEC_Value](#)
- union [TEEC_Parameter](#)
- struct [TEEC_Session](#)
- struct [TEEC_Operation](#)

Typedefs

- typedef uint32_t [TEEC_Result](#)

Functions

- [TEEC_Result](#) [TEEC_InitializeContext](#) (const char *name, [TEEC_Context](#) *context)
- void [TEEC_FinalizeContext](#) ([TEEC_Context](#) *context)
- [TEEC_Result](#) [TEEC_OpenSession](#) ([TEEC_Context](#) *context, [TEEC_Session](#) *session, const [TEEC_UUID](#) *destination, uint32_t connectionMethod, const void *connectionData, [TEEC_Operation](#) *operation, uint32_t *returnOrigin)
- void [TEEC_CloseSession](#) ([TEEC_Session](#) *session)
- [TEEC_Result](#) [TEEC_InvokeCommand](#) ([TEEC_Session](#) *session, uint32_t commandID, [TEEC_Operation](#) *operation, uint32_t *returnOrigin)
- [TEEC_Result](#) [TEEC_RegisterSharedMemory](#) ([TEEC_Context](#) *context, [TEEC_SharedMemory](#) *sharedMem)
- [TEEC_Result](#) [TEEC_AllocateSharedMemory](#) ([TEEC_Context](#) *context, [TEEC_SharedMemory](#) *sharedMem)
- void [TEEC_ReleaseSharedMemory](#) ([TEEC_SharedMemory](#) *sharedMemory)
- void [TEEC_RequestCancellation](#) ([TEEC_Operation](#) *operation)

10.15.1 Typedef Documentation

10.15.1.1 TEEC_Result `typedef uint32_t TEEC_Result`

10.15.2 Function Documentation

10.15.2.1 TEEC_AllocateSharedMemory() `TEEC_Result TEEC_AllocateSharedMemory (
 TEEC_Context * context,
 TEEC_SharedMemory * sharedMem)`

[TEEC_AllocateSharedMemory\(\)](#) - Allocate shared memory for TEE.

Parameters

<i>context</i>	The initialized TEE context structure in which scope to open the session.
<i>sharedMem</i>	Pointer to the allocated shared memory.

Returns

[TEEC_SUCCESS](#) The registration was successful.
[TEEC_ERROR_OUT_OF_MEMORY](#) Memory exhaustion.
[TEEC_Result](#) Something failed.

10.15.2.2 TEEC_CloseSession() `void TEEC_CloseSession (
 TEEC_Session * session)`

[TEEC_CloseSession\(\)](#) - Closes the session which has been opened with the specific trusted application.

Parameters

<i>session</i>	The opened session to close.
----------------	------------------------------

10.15.2.3 TEEC_FinalizeContext() `void TEEC_FinalizeContext (`
`TEEC_Context * context)`

[TEEC_FinalizeContext\(\)](#) - Destroys a context holding connection information on the specific TEE.

This function destroys an initialized TEE context, closing the connection between the client application and the TEE. This function must only be called when all sessions related to this TEE context have been closed and all shared memory blocks have been released.

Parameters

<i>context</i>	The context to be destroyed.
----------------	------------------------------

[TEEC_FinalizeContext\(\)](#) - Destroys a context holding connection information on the specific TEE.

This function finalizes an initialized TEE context, closing the connection between the client application and the TEE. This function must only be called when all sessions related to this TEE context have been closed and all shared memory blocks have been released.

Parameters

<i>context</i>	The context to be finalized.
----------------	------------------------------

10.15.2.4 TEEC_InitializeContext() `TEEC_Result TEEC_InitializeContext (`
`const char * name,`
`TEEC_Context * context)`

[TEEC_InitializeContext\(\)](#) - Initializes a context holding connection information on the specific TEE, designated by the name string.

Parameters

<i>name</i>	A zero-terminated string identifying the TEE to connect to. If name is set to NULL, the default TEE is connected to. NULL is the only supported value in this version of the API implementation.
<i>context</i>	The context structure which is to be initialized.

Returns

TEEC.SUCCESS The initialization was successful.

TEEC.Result Something failed.

10.15.2.5 TEEC_InvokeCommand() `TEEC_Result TEEC_InvokeCommand (`

```
TEEC_Session * session,
uint32_t commandID,
TEEC_Operation * operation,
uint32_t * returnOrigin )
```

`TEEC_InvokeCommand()` - Executes a command in the specified trusted application.

Parameters

<i>session</i>	A handle to an open connection to the trusted application.
<i>commandID</i>	Identifier of the command in the trusted application to invoke.
<i>operation</i>	An operation structure to use in the invoke command. May be set to NULL to signify no operation structure needed.
<i>returnOrigin</i>	A parameter which will hold the error origin if this function returns any value other than TEEC.SUCCESS.

Returns

TEEC.SUCCESS OpenSession successfully opened a new session.

TEEC.Result Something failed.

10.15.2.6 TEEC_OpenSession() `TEEC_Result TEEC_OpenSession (`

```
TEEC_Context * context,
TEEC_Session * session,
const TEEC_UUID * destination,
uint32_t connectionMethod,
const void * connectionData,
TEEC_Operation * operation,
uint32_t * returnOrigin )
```

`TEEC_OpenSession()` - Opens a new session with the specified trusted application.

Parameters

<i>context</i>	The initialized TEE context structure in which scope to open the session.
<i>session</i>	The session to initialize.
<i>destination</i>	A structure identifying the trusted application with which to open a session.
<i>connectionMethod</i>	The connection method to use.
<i>connectionData</i>	Any data necessary to connect with the chosen connection method. Not supported, should be set to NULL.
Paramter list continued on next page	

<i>operation</i>	An operation structure to use in the session. May be set to NULL to signify no operation structure needed.
<i>returnOrigin</i>	A parameter which will hold the error origin if this function returns any value other than TEEC_SUCCESS.

Returns

TEEC_SUCCESS OpenSession successfully opened a new session.

TEEC_Result Something failed.

10.15.2.7 TEEC_RegisterSharedMemory() `TEEC_Result TEEC_RegisterSharedMemory (`
`TEEC_Context * context,`
`TEEC_SharedMemory * sharedMem)`

[TEEC_RegisterSharedMemory\(\)](#) - Register a block of existing memory as a shared block within the scope of the specified context.

Parameters

<i>context</i>	The initialized TEE context structure in which scope to open the session.
<i>sharedMem</i>	pointer to the shared memory structure to register.

Returns

TEEC_SUCCESS The registration was successful.

TEEC_ERROR_OUT_OF_MEMORY Memory exhaustion.

TEEC_Result Something failed.

10.15.2.8 TEEC_ReleaseSharedMemory() `void TEEC_ReleaseSharedMemory (`
`TEEC_SharedMemory * sharedMemory)`

[TEEC_ReleaseSharedMemory\(\)](#) - Free or deregister the shared memory.

Parameters

<i>sharedMem</i>	Pointer to the shared memory to be freed.
------------------	---

10.15.2.9 TEEC.RequestCancellation() `void TEEC.RequestCancellation (`
`TEEC.Operation * operation)`

[TEEC.RequestCancellation\(\)](#) - Request the cancellation of a pending open session or command invocation.

Parameters

<i>operation</i>	Pointer to an operation previously passed to open session or invoke.
------------------	--

10.16 tee_client_api.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  * Copyright (c) 2015, Linaro Limited
5  * All rights reserved.
6  *
7  * Redistribution and use in source and binary forms, with or without
8  * modification, are permitted provided that the following conditions are met:
9  *
10 * 1. Redistributions of source code must retain the above copyright notice,
11 * this list of conditions and the following disclaimer.
12 *
13 * 2. Redistributions in binary form must reproduce the above copyright notice,
14 * this list of conditions and the following disclaimer in the documentation
15 * and/or other materials provided with the distribution.
16 *
17 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
18 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
19 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
20 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
21 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
22 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
23 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
24 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
25 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
26 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
27 * POSSIBILITY OF SUCH DAMAGE.
28 */
29 #ifndef TEE_CLIENT_API_H
30 #define TEE_CLIENT_API_H
31
32 #ifdef __cplusplus
33 extern "C" {
34 #endif
35
36 #include <stdint.h>
37 #include <stddef.h>
38 #include <stdbool.h>
39 #include <limits.h>
40
41 #ifndef DOXYGEN_SHOULD_SKIP_THIS
42 /*
43  * Defines the number of available memory references in an open session or
44  * invoke command operation payload.
45  */
46 #define TEEC_CONFIG_PAYLOAD_REF_COUNT 4
47
48 #define TEEC_CONFIG_SHARED_MEM_MAX_SIZE ULONG_MAX
49
50 #define TEEC_NONE 0x00000000
51 #define TEEC_VALUE_INPUT 0x00000001
52 #define TEEC_VALUE_OUTPUT 0x00000002
53 #define TEEC_VALUE_INOUT 0x00000003
54 #define TEEC_MEMREF_TEMP_INPUT 0x00000005
55 #define TEEC_MEMREF_TEMP_OUTPUT 0x00000006
56 #define TEEC_MEMREF_TEMP_INOUT 0x00000007
57 #define TEEC_MEMREF_WHOLE 0x0000000C
58 #define TEEC_MEMREF_PARTIAL_INPUT 0x0000000D
59 #define TEEC_MEMREF_PARTIAL_OUTPUT 0x0000000E
60 #define TEEC_MEMREF_PARTIAL_INOUT 0x0000000F
61
62 #define TEEC_MEM_INPUT 0x00000001
63 #define TEEC_MEM_OUTPUT 0x00000002
64
65 #endif
66
67 #endif

```

```

135
168 #define TEEC_SUCCESS                0x00000000
169 #define TEEC_ERROR_GENERIC           0xFFFF0000
170 #define TEEC_ERROR_ACCESS_DENIED    0xFFFF0001
171 #define TEEC_ERROR_CANCEL            0xFFFF0002
172 #define TEEC_ERROR_ACCESS_CONFLICT  0xFFFF0003
173 #define TEEC_ERROR_EXCESS_DATA      0xFFFF0004
174 #define TEEC_ERROR_BAD_FORMAT        0xFFFF0005
175 #define TEEC_ERROR_BAD_PARAMETERS    0xFFFF0006
176 #define TEEC_ERROR_BAD_STATE         0xFFFF0007
177 #define TEEC_ERROR_ITEM_NOT_FOUND    0xFFFF0008
178 #define TEEC_ERROR_NOT_IMPLEMENTED   0xFFFF0009
179 #define TEEC_ERROR_NOT_SUPPORTED     0xFFFF000A
180 #define TEEC_ERROR_NO_DATA           0xFFFF000B
181 #define TEEC_ERROR_OUT_OF_MEMORY     0xFFFF000C
182 #define TEEC_ERROR_BUSY              0xFFFF000D
183 #define TEEC_ERROR_COMMUNICATION     0xFFFF000E
184 #define TEEC_ERROR_SECURITY          0xFFFF000F
185 #define TEEC_ERROR_SHORT_BUFFER      0xFFFF0010
186 #define TEEC_ERROR_EXTERNAL_CANCEL   0xFFFF0011
187 #define TEEC_ERROR_TARGET_DEAD       0xFFFF3024
188
202 #define TEEC_ORIGIN_API              0x00000001
203 #define TEEC_ORIGIN_COMMS           0x00000002
204 #define TEEC_ORIGIN_TEE             0x00000003
205 #define TEEC_ORIGIN_TRUSTED_APP     0x00000004
206
223 #define TEEC_LOGIN_PUBLIC            0x00000000
224 #define TEEC_LOGIN_USER              0x00000001
225 #define TEEC_LOGIN_GROUP             0x00000002
226 #define TEEC_LOGIN_APPLICATION       0x00000004
227 #define TEEC_LOGIN_USER_APPLICATION  0x00000005
228 #define TEEC_LOGIN_GROUP_APPLICATION 0x00000006
229
238 #define TEEC_PARAM_TYPES(p0, p1, p2, p3) \
239     ((p0) | ((p1) << 4) | ((p2) << 8) | ((p3) << 12))
240
247 #define TEEC_PARAM_TYPE_GET(p, i) (((p) >> (i * 4)) & 0xF)
248 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
249
250 typedef uint32_t TEEC_Result;
251
256 typedef struct {
257     /* Implementation defined */
258     int fd;
259     bool reg_mem;
260 } TEEC_Context;
261
267 typedef struct {
268     uint32_t timeLow;
269     uint16_t timeMid;
270     uint16_t timeHiAndVersion;
271     uint8_t clockSeqAndNode[8];
272 } TEEC_UUID;
273
290 typedef struct {
291     void *buffer;
292     size_t size;
293     uint32_t flags;
294     /*
295      * Implementation-Defined
296      */
297     int id;
298     size_t allocated_size;
299     void *shadow_buffer;
300     int registered_fd;
301     bool buffer_allocated;
302 } TEEC_SharedMemory;
303
316 typedef struct {
317     void *buffer;
318     size_t size;
319 } TEEC_TempMemoryReference;
320
336 typedef struct {
337     TEEC_SharedMemory *parent;
338     size_t size;
339     size_t offset;
340 } TEEC_RegisteredMemoryReference;
341
352 typedef struct {
353     uint32_t a;
354     uint32_t b;
355 } TEEC_Value;
356
371 typedef union {
372     TEEC_TempMemoryReference tmpref;

```

```

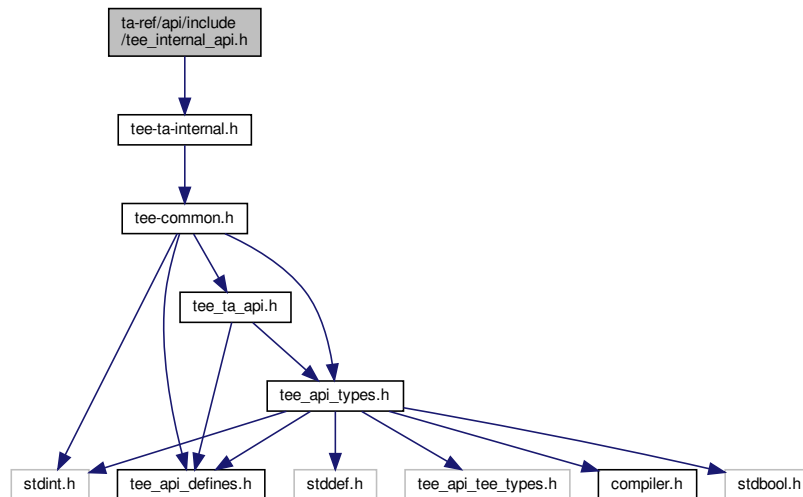
373     TEEC.RegisteredMemoryReference memref;
374     TEEC.Value value;
375 } TEEC.Parameter;
376
381 typedef struct {
382     /* Implementation defined */
383     TEEC.Context *ctx;
384     uint32_t session_id;
385 } TEEC.Session;
386
401 typedef struct {
402     uint32_t started;
403     uint32_t paramTypes;
404     TEEC.Parameter params[TEEC_CONFIG.PAYLOAD_REF_COUNT];
405     /* Implementation-Defined */
406     TEEC.Session *session;
407 } TEEC.Operation;
408
423 TEEC.Result TEEC.InitializeContext(const char *name, TEEC.Context *context);
424
436 void TEEC.FinalizeContext(TEEC.Context *context);
437
464 TEEC.Result TEEC.OpenSession(TEEC.Context *context,
465                             TEEC.Session *session,
466                             const TEEC.UUID *destination,
467                             uint32_t connectionMethod,
468                             const void *connectionData,
469                             TEEC.Operation *operation,
470                             uint32_t *returnOrigin);
471
478 void TEEC.CloseSession(TEEC.Session *session);
479
497 TEEC.Result TEEC.InvokeCommand(TEEC.Session *session,
498                               uint32_t commandID,
499                               TEEC.Operation *operation,
500                               uint32_t *returnOrigin);
501
514 TEEC.Result TEEC.RegisterSharedMemory(TEEC.Context *context,
515                                       TEEC.SharedMemory *sharedMem);
516
528 TEEC.Result TEEC.AllocateSharedMemory(TEEC.Context *context,
529                                       TEEC.SharedMemory *sharedMem);
530
536 void TEEC.ReleaseSharedMemory(TEEC.SharedMemory *sharedMemory);
537
545 void TEEC.RequestCancellation(TEEC.Operation *operation);
546
547 #ifdef __cplusplus
548 }
549 #endif
550
551 #endif

```

10.17 ta-ref/api/include/tee_internal_api.h File Reference

```
#include "tee-ta-internal.h"
```

Include dependency graph for tee_internal_api.h:



10.18 tee_internal_api.h

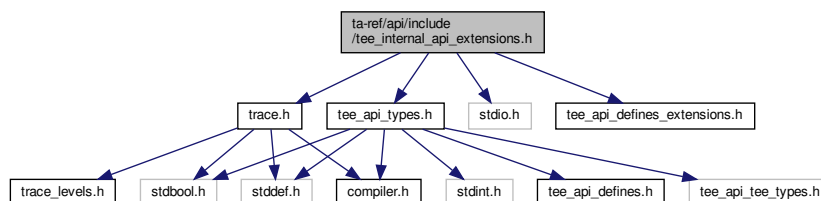
[Go to the documentation of this file.](#)

```
1 #include "tee-ta-internal.h"
```

10.19 ta-ref/api/include/tee_internal_api_extensions.h File Reference

```
#include <trace.h>
#include <stdio.h>
#include <tee_api_defines_extensions.h>
#include <tee_api_types.h>
```

Include dependency graph for tee_internal_api_extensions.h:



Functions

- void `tee_user_mem_mark_heap` (void)
- size_t `tee_user_mem_check_heap` (void)
- TEE_Result `TEE_CacheClean` (char *buf, size_t len)
- TEE_Result `TEE_CacheFlush` (char *buf, size_t len)
- TEE_Result `TEE_CacheInvalidate` (char *buf, size_t len)
- void * `tee_map_zi` (size_t len, uint32_t flags)
- TEE_Result `tee_unmap` (void *buf, size_t len)
- TEE_Result `tee_uuid_from_str` (TEE_UUID *uuid, const char *s)

10.19.1 Function Documentation

10.19.1.1 TEE_CacheClean() TEE_Result TEE_CacheClean (
char * *buf*,
size_t *len*)

10.19.1.2 TEE_CacheFlush() TEE_Result TEE_CacheFlush (
char * *buf*,
size_t *len*)

10.19.1.3 TEE_CacheInvalidate() TEE_Result TEE_CacheInvalidate (
char * *buf*,
size_t *len*)

10.19.1.4 tee_map_zi() void * tee_map_zi (
size_t *len*,
uint32_t *flags*)

10.19.1.5 tee_unmap() TEE_Result tee_unmap (
void * *buf*,
size_t *len*)

10.19.1.6 tee_user_mem_check_heap() size_t tee_user_mem_check_heap (
void)

10.19.1.7 tee_user_mem_mark_heap() void tee_user_mem_mark_heap (

void)

10.19.1.8 tee_uuid_from_str() TEE_Result tee_uuid_from_str (

TEE_UUID * uuid,

const char * s)

10.20 tee_internal_api_extensions.h

[Go to the documentation of this file.](#)

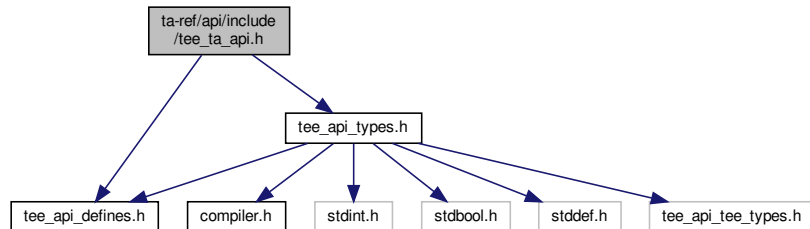
```

1 /* SPDX-License-Identifier: BSD-2-Clause */
2 /*
3  * Copyright (c) 2014, STMicroelectronics International N.V.
4  */
5
6 #ifndef TEE_INTERNAL_API_EXTENSIONS_H
7 #define TEE_INTERNAL_API_EXTENSIONS_H
8
9 /* trace support */
10 #include <trace.h>
11 #include <stdio.h>
12 #include <tee_api_defines_extensions.h>
13 #include <tee_api_types.h>
14
15 void tee_user_mem_mark_heap(void);
16 size_t tee_user_mem_check_heap(void);
17 /* Hint implementation defines */
18
19 #ifndef DOXYGEN_SHOULD_SKIP_THIS
20 #define TEE_USER_MEM_HINT_NO_FILL_ZERO 0x80000000
21 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
22
23 /*
24  * Cache maintenance support (TA requires the CACHE_MAINTENANCE property)
25  *
26  * TEE.CacheClean() Write back to memory any dirty data cache lines. The line
27  * is marked as not dirty. The valid bit is unchanged.
28  *
29  * TEE.CacheFlush() Purges any valid data cache lines. Any dirty cache lines
30  * are first written back to memory, then the cache line is
31  * invalidated.
32  *
33  * TEE.CacheInvalidate() Invalidate any valid data cache lines. Any dirty line
34  * are not written back to memory.
35  */
36 TEE_Result TEE.CacheClean(char *buf, size_t len);
37 TEE_Result TEE.CacheFlush(char *buf, size_t len);
38 TEE_Result TEE.CacheInvalidate(char *buf, size_t len);
39
40 /*
41  * tee_map_zi() - Map zero initialized memory
42  * @len:      Number of bytes
43  * @flags:    0 or TEE_MEMORY_ACCESS_ANY_OWNER to allow sharing with other TAs
44  *
45  * Returns valid pointer on success or NULL on error.
46  */
47 void *tee_map_zi(size_t len, uint32_t flags);
48
49 /*
50  * tee_unmap() - Unmap previously mapped memory
51  * @buf:      Buffer
52  * @len:      Number of bytes
53  *
54  * Note that supplied @buf and @len has to match exactly what has
55  * previously been returned by tee_map_zi().
56  *
57  * Return TEE_SUCCESS on success or TEE_ERROR_* on failure.
58  */
59 TEE_Result tee_unmap(void *buf, size_t len);
60
61 /*
62  * Convert a UUID string @s into a TEE_UUID @uuid
63  * Expected format for @s is: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
64  * 'x' being any hexadecimal digit (0-9a-fA-F)
65  */
66 TEE_Result tee_uuid_from_str(TEE_UUID *uuid, const char *s);
67
68 #endif

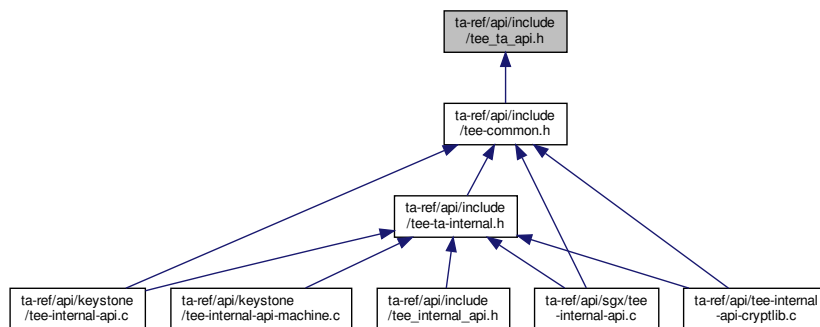
```

10.21 ta-ref/api/include/tee_ta_api.h File Reference

```
#include <tee_api_defines.h>
#include <tee_api_types.h>
Include dependency graph for tee_ta_api.h:
```



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



Functions

- [TEE_Result](#) TA_EXPORT [TA_CreateEntryPoint](#) (void)
- void TA_EXPORT [TA_DestroyEntryPoint](#) (void)
- [TEE_Result](#) TA_EXPORT [TA_OpenSessionEntryPoint](#) (uint32_t paramTypes, [TEE_Param](#) params[TEE_NUM_PARAMS], void **sessionContext)
- void TA_EXPORT [TA_CloseSessionEntryPoint](#) (void *sessionContext)
- [TEE_Result](#) TA_EXPORT [TA_InvokeCommandEntryPoint](#) (void *sessionContext, uint32_t commandID, uint32_t paramTypes, [TEE_Param](#) params[TEE_NUM_PARAMS])

10.21.1 Function Documentation

10.21.1.1 TA_CloseSessionEntryPoint() void TA_EXPORT TA_CloseSessionEntryPoint (void * sessionContext)

10.21.1.2 TA_CreateEntryPoint() `TEE_Result TA_EXPORT TA_CreateEntryPoint (`
`void)`

10.21.1.3 TA_DestroyEntryPoint() `void TA_EXPORT TA_DestroyEntryPoint (`
`void)`

10.21.1.4 TA_InvokeCommandEntryPoint() `TEE_Result TA_EXPORT TA_InvokeCommandEntryPoint (`
`void * sessionContext,`
`uint32_t commandID,`
`uint32_t paramTypes,`
`TEE_Param params[TEE_NUM_PARAMS])`

10.21.1.5 TA_OpenSessionEntryPoint() `TEE_Result TA_EXPORT TA_OpenSessionEntryPoint (`
`uint32_t paramTypes,`
`TEE_Param params[TEE_NUM_PARAMS],`
`void ** sessionContext)`

10.22 tee_ta_api.h

[Go to the documentation of this file.](#)

```
1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25 * POSSIBILITY OF SUCH DAMAGE.
26 */
27
28 /* Based on GP TEE Internal API Specification Version 0.22 */
29 #ifndef TEE_TA_API_H
30 #define TEE_TA_API_H
31
32 #include <tee_api_defines.h>
33 #include <tee_api_types.h>
34
35 #ifndef DOXYGEN_SHOULD_SKIP_THIS
36 /* This is a null define in STE TEE environment */
37 #define TA_EXPORT
38 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
39
40 /*
```



```

41 * TA Interface
42 *
43 * Each Trusted Application must provide the Implementation with a number
44 * of functions, collectively called the "TA interface". These functions
45 * are the entry points called by the Trusted Core Framework to create the
46 * instance, notify the instance that a new client is connecting, notify
47 * the instance when the client invokes a command, etc.
48 *
49 * Trusted Application Entry Points:
50 */
51
52 /*
53 * The function TAA.CreateEntryPoint is the Trusted Application's
54 * constructor, which the Framework calls when it creates a new instance of
55 * the Trusted Application. To register instance data, the implementation
56 * of this constructor can use either global variables or the function
57 * TEE.InstanceSetData.
58 *
59 * Return Value:
60 * - TEE.SUCCESS: if the instance is successfully created, the function
61 *   must return TEE.SUCCESS.
62 * - Any other value: if any other code is returned the instance is not
63 *   created, and no other entry points of this instance will be called.
64 *   The Framework MUST reclaim all resources and dereference all objects
65 *   related to the creation of the instance.
66 *
67 *   If this entry point was called as a result of a client opening a
68 *   session, the error code is returned to the client and the session is
69 *   not opened.
70 */
71 TEE_Result TAA_EXPORT TAA_CreateEntryPoint(void);
72
73 /*
74 * The function TAA.DestroyEntryPoint is the Trusted Applications
75 * destructor, which the Framework calls when the instance is being
76 * destroyed.
77 *
78 * When the function TAA.DestroyEntryPoint is called, the Framework
79 * guarantees that no client session is currently open. Once the call to
80 * TAA.DestroyEntryPoint has been completed, no other entry point of this
81 * instance will ever be called.
82 *
83 * Note that when this function is called, all resources opened by the
84 * instance are still available. It is only after the function returns that
85 * the Implementation MUST start automatically reclaiming resources left
86 * opened.
87 *
88 * Return Value:
89 * This function can return no success or error code. After this function
90 * returns the Implementation MUST consider the instance destroyed and
91 * reclaims all resources left open by the instance.
92 */
93 void TAA_EXPORT TAA_DestroyEntryPoint(void);
94
95 /*
96 * The Framework calls the function TAA.OpenSessionEntryPoint when a client
97 * requests to open a session with the Trusted Application. The open
98 * session request may result in a new Trusted Application instance being
99 * created as defined in section 4.5.
100 *
101 * The client can specify parameters in an open operation which are passed
102 * to the Trusted Application instance in the arguments paramTypes and
103 * params. These arguments can also be used by the Trusted Application
104 * instance to transfer response data back to the client. See section 4.3.6
105 * for a specification of how to handle the operation parameters.
106 *
107 * If this function returns TEE.SUCCESS, the client is connected to a
108 * Trusted Application instance and can invoke Trusted Application
109 * commands. When the client disconnects, the Framework will eventually
110 * call the TAA.CloseSessionEntryPoint entry point.
111 *
112 * If the function returns any error, the Framework rejects the connection
113 * and returns the error code and the current content of the parameters the
114 * client. The return origin is then set to TEE.ORIGIN.TRUSTED.APP.
115 *
116 * The Trusted Application instance can register a session data pointer by
117 * setting *pSessionContext. The value of this pointer is not interpreted
118 * by the Framework, and is simply passed back to other TA functions
119 * within this session. Note that *pSessionContext may be set with a pointer
120 * to a memory allocated by the Trusted Application instance or with
121 * anything else, like an integer, a handle etc. The Framework will not
122 * automatically free *pSessionContext when the session is closed; the
123 * Trusted Application instance is responsible for freeing memory if
124 * required.
125 *
126 * During the call to TAA.OpenSessionEntryPoint the client may request to
127 * cancel the operation. See section 4.10 for more details on

```

```

128 * cancellations. If the call to TA.OpenSessionEntryPoint returns
129 * TEE_SUCCESS, the client must consider the session as successfully opened
130 * and explicitly close it if necessary.
131 *
132 * Parameters:
133 * - paramTypes: the types of the four parameters.
134 * - params: a pointer to an array of four parameters.
135 * - sessionContext: A pointer to a variable that can be filled by the
136 *   Trusted Application instance with an opaque void* data pointer
137 *
138 * Return Value:
139 * - TEE_SUCCESS if the session is successfully opened.
140 * - Any other value if the session could not be open.
141 *   o The error code may be one of the pre-defined codes, or may be a new
142 *     error code defined by the Trusted Application implementation itself.
143 */
144 TEE_Result TA_EXPORT TA_OpenSessionEntryPoint(uint32_t paramTypes,
145                                             TEE_Param params[TEE_NUM_PARAMS],
146                                             void **sessionContext);
147
148 /*
149 * The Framework calls this function to close a client session. During the
150 * call to this function the implementation can use any session functions.
151 *
152 * The Trusted Application implementation is responsible for freeing any
153 * resources consumed by the session being closed. Note that the Trusted
154 * Application cannot refuse to close a session, but can hold the closing
155 * until it returns from TA.CloseSessionEntryPoint. This is why this
156 * function cannot return an error code.
157 *
158 * Parameters:
159 * - sessionContext: The value of the void* opaque data pointer set by the
160 *   Trusted Application in the function TA.OpenSessionEntryPoint for this
161 *   session.
162 */
163 void TA_EXPORT TA_CloseSessionEntryPoint(void *sessionContext);
164
165 /*
166 * The Framework calls this function when the client invokes a command
167 * within the given session.
168 *
169 * The Trusted Application can access the parameters sent by the client
170 * through the paramTypes and params arguments. It can also use these
171 * arguments to transfer response data back to the client.
172 *
173 * During the call to TA.InvokeCommandEntryPoint the client may request to
174 * cancel the operation.
175 *
176 * A command is always invoked within the context of a client session.
177 * Thus, any session function can be called by the command implementation.
178 *
179 * Parameter:
180 * - sessionContext: The value of the void* opaque data pointer set by the
181 *   Trusted Application in the function TA.OpenSessionEntryPoint.
182 * - commandID: A Trusted Application-specific code that identifies the
183 *   command to be invoked.
184 * - paramTypes: the types of the four parameters.
185 * - params: a pointer to an array of four parameters.
186 *
187 * Return Value:
188 * - TEE_SUCCESS: if the command is successfully executed, the function
189 *   must return this value.
190 * - Any other value: if the invocation of the command fails for any
191 *   reason.
192 *   o The error code may be one of the pre-defined codes, or may be a new
193 *     error code defined by the Trusted Application implementation itself.
194 */
195
196 TEE_Result TA_EXPORT TA_InvokeCommandEntryPoint(void *sessionContext,
197                                                uint32_t commandID,
198                                                uint32_t paramTypes,
199                                                TEE_Param params[TEE_NUM_PARAMS]);
200
201 /*
202 * Correspondance Client Functions <--> TA Functions
203 *
204 * TEE.OpenSession or TEE.OpenTASession:
205 * If a new Trusted Application instance is needed to handle the session,
206 * TA.CreateEntryPoint is called.
207 * Then, TA.OpenSessionEntryPoint is called.
208 *
209 *
210 * TEE.InvokeCommand or TEE.InvokeTACommand:
211 * TA.InvokeCommandEntryPoint is called.
212 *
213 *
214 * TEE.CloseSession or TEE.CloseTASession:

```

```
215 * TA_CloseSessionEntryPoint is called.
216 * For a multi-instance TA or for a single-instance, non keep-alive TA, if
217 * the session closed was the last session on the instance, then
218 * TADestroyEntryPoint is called. Otherwise, the instance is kept until
219 * the TEE shuts down.
220 *
221 */
222
223 #endif
```

10.23 ta-ref/api/include/test_dev_key.h File Reference

Variables

- static const unsigned char [_sanctum_dev_secret_key](#) []
- static const size_t [_sanctum_dev_secret_key_len](#) = 64
- static const unsigned char [_sanctum_dev_public_key](#) []
- static const size_t [_sanctum_dev_public_key_len](#) = 32

10.23.1 Variable Documentation

10.23.1.1 [_sanctum_dev_public_key](#) const unsigned char [_sanctum_dev_public_key](#)[] [static]

Initial value:

```
= {
    0x0f, 0xaa, 0xd4, 0xff, 0x01, 0x17, 0x85, 0x83, 0xba, 0xa5, 0x88, 0x96,
    0x6f, 0x7c, 0x1f, 0xf3, 0x25, 0x64, 0xdd, 0x17, 0xd7, 0xdc, 0x2b, 0x46,
    0xcb, 0x50, 0xa8, 0x4a, 0x69, 0x27, 0x0b, 0x4c
}
```

10.23.1.2 [_sanctum_dev_public_key_len](#) const size_t [_sanctum_dev_public_key_len](#) = 32 [static]

10.23.1.3 [_sanctum_dev_secret_key](#) const unsigned char [_sanctum_dev_secret_key](#)[] [static]

Initial value:

```
= {
    0x40, 0xa0, 0x99, 0x47, 0x8c, 0xce, 0xfa, 0x3a, 0x06, 0x63, 0xab, 0xc9,
    0x5e, 0x7a, 0x1e, 0xc9, 0x54, 0xb4, 0xf5, 0xf6, 0x45, 0xba, 0xd8, 0x04,
    0xdb, 0x13, 0xe7, 0xd7, 0x82, 0x6c, 0x70, 0x73, 0x57, 0x6a, 0x9a, 0xb6,
    0x21, 0x60, 0xd9, 0xd1, 0xc6, 0xae, 0xdc, 0x29, 0x85, 0x2f, 0xb9, 0x60,
    0xee, 0x51, 0x32, 0x83, 0x5a, 0x16, 0x89, 0xec, 0x06, 0xa8, 0x72, 0x34,
    0x51, 0xaa, 0x0e, 0x4a
}
```

10.23.1.4 [_sanctum_dev_secret_key_len](#) const size_t [_sanctum_dev_secret_key_len](#) = 64 [static]

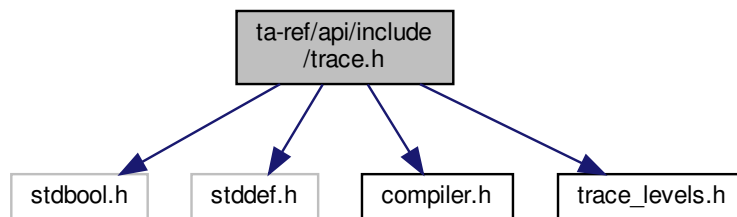
10.24 test_dev_key.h

[Go to the documentation of this file.](#)

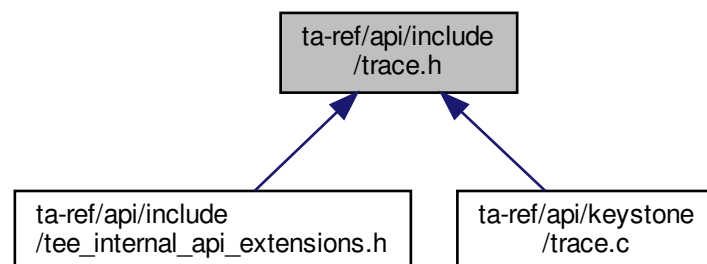
```
1 /* These are known device TESTING keys, use them for testing on platforms/qemu */
2
3 #warning Using TEST device root key. No integrity guarantee.
4 static const unsigned char _sanctum_dev_secret_key[] = {
5     0x40, 0xa0, 0x99, 0x47, 0x8c, 0xce, 0xfa, 0x3a, 0x06, 0x63, 0xab, 0xc9,
6     0x5e, 0x7a, 0x1e, 0xc9, 0x54, 0xb4, 0xf5, 0xf6, 0x45, 0xba, 0xd8, 0x04,
7     0xdb, 0x13, 0xe7, 0xd7, 0x82, 0x6c, 0x70, 0x73, 0x57, 0x6a, 0x9a, 0xb6,
8     0x21, 0x60, 0xd9, 0xd1, 0xc6, 0xae, 0xdc, 0x29, 0x85, 0x2f, 0xb9, 0x60,
9     0xee, 0x51, 0x32, 0x83, 0x5a, 0x16, 0x89, 0xec, 0x06, 0xa8, 0x72, 0x34,
10    0x51, 0xaa, 0x0e, 0x4a
11 };
12 static const size_t _sanctum_dev_secret_key_len = 64;
13
14 static const unsigned char _sanctum_dev_public_key[] = {
15     0x0f, 0xaa, 0xd4, 0xff, 0x01, 0x17, 0x85, 0x83, 0xba, 0xa5, 0x88, 0x96,
16     0x6f, 0x7c, 0x1f, 0xf3, 0x25, 0x64, 0xdd, 0x17, 0xd7, 0xdc, 0x2b, 0x46,
17     0xcb, 0x50, 0xa8, 0x4a, 0x69, 0x27, 0x0b, 0x4c
18 };
19 static const size_t _sanctum_dev_public_key_len = 32;
```

10.25 ta-ref/api/include/trace.h File Reference

```
#include <stdbool.h>
#include <stddef.h>
#include <compiler.h>
#include <trace_levels.h>
Include dependency graph for trace.h:
```



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



Functions

- void `trace_ext.puts` (const char *str)
- int `trace_ext.get_thread_id` (void)
- void `trace_set_level` (int level)
- int `trace_get_level` (void)
- void `trace_printf` (const char *func, int line, int level, bool level_ok, const char *fmt,...) `_printf(5)`
- void `dhex_dump` (const char *function, int line, int level, const void *buf, int len)

Variables

- int `trace_level`
- const char `trace_ext.prefix` []

10.25.1 Function Documentation

10.25.1.1 `dhex_dump()` void `dhex_dump` (
 const char * *function*,
 int *line*,
 int *level*,
 const void * *buf*,
 int *len*)

10.25.1.2 `trace_ext.get_thread_id()` int `trace_ext.get_thread_id` (
 void)

10.25.1.3 `trace_ext.puts()` void `trace_ext.puts` (
 const char * *str*)

10.25.1.4 `trace_get_level()` int `trace_get_level` (
 void)

10.25.1.5 `trace_printf()` void `trace_printf` (
 const char * *func*,
 int *line*,
 int *level*,
 bool *level_ok*,
 const char * *fmt*,
 ...)

10.25.1.6 trace_set_level() void trace_set_level (
int level)

10.25.2 Variable Documentation

10.25.2.1 trace_ext_prefix const char trace_ext_prefix[] [extern]

10.25.2.2 trace_level int trace_level [extern]

10.26 trace.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25 * POSSIBILITY OF SUCH DAMAGE.
26 */
27 #ifndef TRACE_H
28 #define TRACE_H
29
30 #include <stdbool.h>
31 #include <stddef.h>
32 #include <compiler.h>
33 #include <trace_levels.h>
34
35 #ifndef DOXYGEN_SHOULD_SKIP_THIS
36 #define MAX_PRINT_SIZE 256
37 #define MAX_FUNC_PRINT_SIZE 32
38
39 #ifndef TRACE_LEVEL
40 #define TRACE_LEVEL TRACE_MAX
41 #endif
42 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
43
44 /*
45  * Symbols provided by the entity that uses this API.
46  */
47 extern int trace_level;
48 extern const char trace_ext_prefix[];
49 void trace_ext_puts(const char *str);
50 int trace_ext_get_thread_id(void);
51 void trace_set_level(int level);
52 int trace_get_level(void);
53
54 /* Internal functions used by the macros below */
55 void trace_printf(const char *func, int line, int level, bool level_ok,
```

```

56         const char *fmt, ...) __printf(5, 6);
57
58 #ifndef DOXYGEN_SHOULD_SKIP_THIS
59 #define trace_printf_helper(level, level_ok, ...) \
60     trace_printf(__func__, __LINE__, (level), (level_ok), \
61         __VA_ARGS__)
62
63 /* Formatted trace tagged with level independent */
64 #if (TRACE_LEVEL <= 0)
65 #define MSG(...)    (void)0
66 #else
67 #define MSG(...)    trace_printf_helper(0, false, __VA_ARGS__)
68 #endif
69
70 /* Formatted trace tagged with TRACE_ERROR level */
71 #if (TRACE_LEVEL < TRACE_ERROR)
72 #define EMSG(...)    (void)0
73 #else
74 #define EMSG(...)    trace_printf_helper(TRACE_ERROR, true, __VA_ARGS__)
75 #endif
76
77 /* Formatted trace tagged with TRACE_INFO level */
78 #if (TRACE_LEVEL < TRACE_INFO)
79 #define IMSG(...)    (void)0
80 #else
81 #define IMSG(...)    trace_printf_helper(TRACE_INFO, true, __VA_ARGS__)
82 #endif
83
84 /* Formatted trace tagged with TRACE_DEBUG level */
85 #if (TRACE_LEVEL < TRACE_DEBUG)
86 #define DMSG(...)    (void)0
87 #else
88 #define DMSG(...)    trace_printf_helper(TRACE_DEBUG, true, __VA_ARGS__)
89 #endif
90
91 /* Formatted trace tagged with TRACE_FLOW level */
92 #if (TRACE_LEVEL < TRACE_FLOW)
93 #define FMSG(...)    (void)0
94 #else
95 #define FMSG(...)    trace_printf_helper(TRACE_FLOW, true, __VA_ARGS__)
96 #endif
97
98 /* Formatted trace tagged with TRACE_FLOW level and prefix with '>' */
99 #define INMSG(...)    FMSG("> " __VA_ARGS__)
100 /* Formatted trace tagged with TRACE_FLOW level and prefix with '<' */
101 #define OUTMSG(...)    FMSG("< " __VA_ARGS__)
102 /* Formatted trace tagged with TRACE_FLOW level and prefix with '<' and print
103  * an error message if r != 0 */
104 #define OUTRMSG(r) \
105     do { \
106         OUTMSG("r=[%x]", r); \
107         return r; \
108     } while (0)
109
110 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
111
112 void dhex_dump(const char *function, int line, int level,
113     const void *buf, int len);
114
115
116 #ifndef DOXYGEN_SHOULD_SKIP_THIS
117 #if (TRACE_LEVEL < TRACE_DEBUG)
118 #define DHEXDUMP(buf, len) (void)0
119 #else
120 #define DHEXDUMP(buf, len) dhex_dump(__func__, __LINE__, TRACE_DEBUG, \
121     buf, len)
122 #endif
123
124
125 /* Trace api without trace formatting */
126
127 #define trace_printf_helper_raw(level, level_ok, ...) \
128     trace_printf(NULL, 0, (level), (level_ok), __VA_ARGS__)
129
130 /* No formatted trace tagged with level independent */
131 #if (TRACE_LEVEL <= 0)
132 #define MSG_RAW(...)    (void)0
133 #else
134 #define MSG_RAW(...)    trace_printf_helper_raw(0, false, __VA_ARGS__)
135 #endif
136
137 /* No formatted trace tagged with TRACE_ERROR level */
138 #if (TRACE_LEVEL < TRACE_ERROR)
139 #define EMSG_RAW(...)    (void)0
140 #else
141 #define EMSG_RAW(...)    trace_printf_helper_raw(TRACE_ERROR, true, __VA_ARGS__)
142 #endif

```

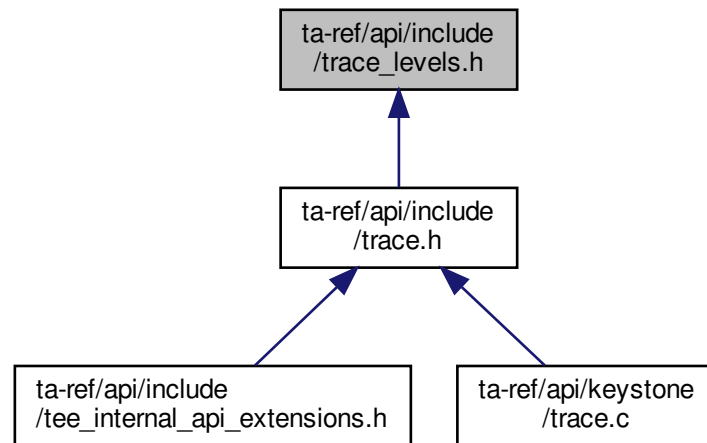
```

143
144 /* No formatted trace tagged with TRACE.INFO level */
145 #if (TRACE_LEVEL < TRACE.INFO)
146 #define MSG_RAW(...) (void)0
147 #else
148 #define MSG_RAW(...) trace_printf_helper.raw(TRACE.INFO, true, __VA_ARGS__)
149 #endif
150
151 /* No formatted trace tagged with TRACE.DEBUG level */
152 #if (TRACE_LEVEL < TRACE.DEBUG)
153 #define DMSG_RAW(...) (void)0
154 #else
155 #define DMSG_RAW(...) trace_printf_helper.raw(TRACE.DEBUG, true, __VA_ARGS__)
156 #endif
157
158 /* No formatted trace tagged with TRACE.FLOW level */
159 #if (TRACE_LEVEL < TRACE.FLOW)
160 #define FMSG_RAW(...) (void)0
161 #else
162 #define FMSG_RAW(...) trace_printf_helper.raw(TRACE.FLOW, true, __VA_ARGS__)
163 #endif
164
165 #if (TRACE_LEVEL <= 0)
166 #define MSG(...) (void)0
167 #else
168 /*
169  * Synchronised flushed trace, an Always message straight to HW trace IP.
170  * Current only supported inside OP-TEE kernel, will be just like an MSG()
171  * in another context.
172  */
173 #define MSG(...) \
174     trace_printf(__func__, __LINE__, TRACE.ERROR, true, __VA_ARGS__)
175
176 #endif /* TRACE_LEVEL */
177
178 #if defined(__KERNEL__) && defined(CFG_UNWIND)
179 #include <kernel/unwind.h>
180 #define _PRINT_STACK
181 #endif
182
183 #if defined(_PRINT_STACK) && (TRACE_LEVEL >= TRACE.ERROR)
184 #define EPRINT_STACK() print_kernel_stack(TRACE.ERROR)
185 #else
186 #define EPRINT_STACK() (void)0
187 #endif
188
189 #if defined(_PRINT_STACK) && (TRACE_LEVEL >= TRACE.INFO)
190 #define IPRINT_STACK() print_kernel_stack(TRACE.INFO)
191 #else
192 #define IPRINT_STACK() (void)0
193 #endif
194
195 #if defined(_PRINT_STACK) && (TRACE_LEVEL >= TRACE.DEBUG)
196 #define DPRINT_STACK() print_kernel_stack(TRACE.DEBUG)
197 #else
198 #define DPRINT_STACK() (void)0
199 #endif
200
201 #if defined(_PRINT_STACK) && (TRACE_LEVEL >= TRACE.FLOW)
202 #define FPRINT_STACK() print_kernel_stack(TRACE.FLOW)
203 #else
204 #define FPRINT_STACK() (void)0
205 #endif
206
207 #if defined(__KERNEL__) && defined(CFG_UNWIND)
208 #undef _PRINT_STACK
209 #endif
210
211 #endif /* DOXYGEN_SHOULD_SKIP_THIS */
212 #endif /* TRACE_H */

```


10.27 ta-ref/api/include/trace_levels.h File Reference

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



10.28 trace_levels.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2014, STMicroelectronics International N.V.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
6  * modification, are permitted provided that the following conditions are met:
7  *
8  * 1. Redistributions of source code must retain the above copyright notice,
9  * this list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
19 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
20 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
21 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
22 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
23 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25 * POSSIBILITY OF SUCH DAMAGE.
26 */
27 #ifndef TRACE_LEVELS_H
28 #define TRACE_LEVELS_H
29
30 /*
31  * Trace levels.
32  *
33  * ALWAYS is used when you always want a print to be seen, but it is not always
34  * an error.
35  *
36  * ERROR is used when some kind of error has happened, this is most likely the
37  * print you will use most of the time when you report some kind of error.
38  *
39  * INFO is used when you want to print some 'normal' text to the user.
40  * This is the default level.

```

```

41 *
42 * DEBUG is used to print extra information to enter deeply in the module.
43 *
44 * FLOW is used to print the execution flox, typically the in/out of functions.
45 *
46 */
47
48 #ifndef DOXYGEN_SHOULD_SKIP_THIS
49 #define TRACE_MIN 1
50 #define TRACE_ERROR TRACE_MIN
51 #define TRACE_INFO 2
52 #define TRACE_DEBUG 3
53 #define TRACE_FLOW 4
54 #define TRACE_MAX TRACE_FLOW
55
56 /* Trace level of the casual printf */
57 #define TRACE_PRINTF_LEVEL TRACE_ERROR
58
59 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
60 #endif /*TRACE_LEVELS_H*/
61

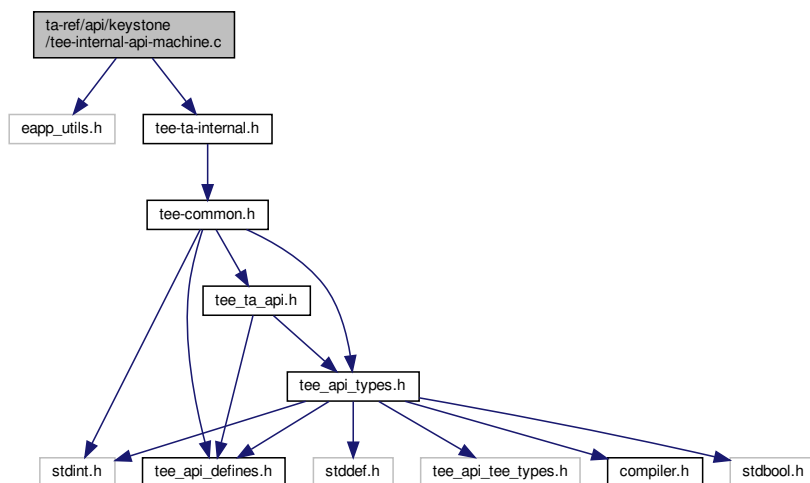
```

10.29 ta-ref/api/keystone/tee-internal-api-machine.c File Reference

```
#include "eapp_utils.h"
```

```
#include "tee-ta-internal.h"
```

Include dependency graph for tee-internal-api-machine.c:



Functions

- void `__attribute__((noreturn))`

10.29.1 Function Documentation

10.29.1.1 `__attribute__((noreturn)) void __attribute__((noreturn))` (

TEE.Panic() - Raises a panic in the Trusted Application instance.

When a Trusted Application calls the TEE.Panic function, the current instance shall be destroyed and all the resources opened by the instance shall be reclaimed. All sessions opened from the panicking instance on another TA shall be gracefully closed and all cryptographic objects and operations shall be closed properly.

Parameters

<code>code</code>	An informative panic code defined by the TA.
-------------------	--

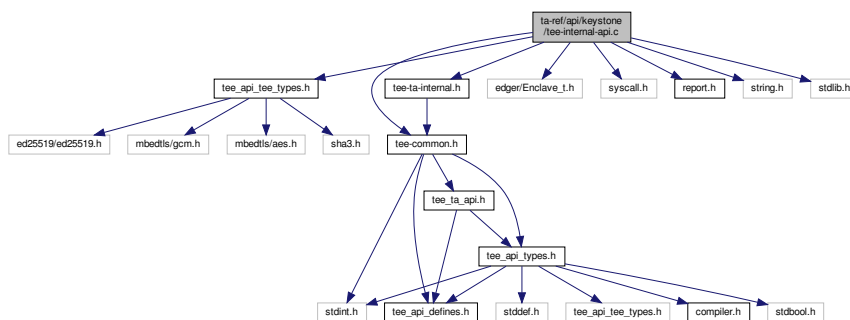
Returns

panic code will be returned.

10.30 ta-ref/api/keystone/tee-internal-api.c File Reference

```
#include "tee_api_tee_types.h"
#include "tee-common.h"
#include "tee-ta-internal.h"
#include "edger/Enclave_t.h"
#include "syscall.h"
#include "report.h"
#include <string.h>
#include <stdlib.h>
```

Include dependency graph for tee-internal-api.c:



Functions

- void * [TEE_Malloc](#) (uint32_t size, uint32_t hint)
- void * [TEE_Realloc](#) (void *buffer, uint32_t newSize)
- void [TEE_Free](#) (void *buffer)
- void [TEE_GetREETime](#) (TEE_Time *time)
Core Functions, Time Functions.
- void [TEE_GetSystemTime](#) (TEE_Time *time)
Core Functions, Time Functions.
- [TEE_Result GetRelTimeStart](#) (uint64_t start)
Core Functions, Time Functions.
- [TEE_Result GetRelTimeEnd](#) (uint64_t end)
Core Functions, Time Functions.
- static int [flags2flags](#) (int flags)
- static int [set_object_key](#) (void *id, unsigned int idlen, [TEE_ObjectHandle](#) object)
- static [TEE_Result](#) [OpenPersistentObject](#) (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, [TEE_ObjectHandle](#) *object, int ocreat)

- **TEE_Result TEE_CreatePersistentObject** (uint32_t storagelD, const void *objectID, uint32_t objectIDLen, uint32_t flags, **TEE_ObjectHandle** attributes, const void *initialData, uint32_t initialDataLen, **TEE_ObjectHandle** *object)
Core Functions, Secure Storage Functions (data is isolated for each TA)
- **TEE_Result TEE_OpenPersistentObject** (uint32_t storagelD, const void *objectID, uint32_t objectIDLen, uint32_t flags, **TEE_ObjectHandle** *object)
Core Functions, Secure Storage Functions (data is isolated for each TA)
- **TEE_Result TEE_GetObjectInfo1** (**TEE_ObjectHandle** object, **TEE_ObjectInfo** *objectInfo)
Core Functions, Secure Storage Functions (data is isolated for each TA)
- **TEE_Result TEE_WriteObjectData** (**TEE_ObjectHandle** object, const void *buffer, uint32_t size)
Core Functions, Secure Storage Functions (data is isolated for each TA)
- **TEE_Result TEE_ReadObjectData** (**TEE_ObjectHandle** object, void *buffer, uint32_t size, uint32_t *count)
Core Functions, Secure Storage Functions (data is isolated for each TA)
- void **TEE_CloseObject** (**TEE_ObjectHandle** object)
Core Functions, Secure Storage Functions (data is isolated for each TA)
- WC_RNG * **get_wc_rng** (void)
- int **wc_ocall_genseed** (void *nonce, uint32_t len)
- void **TEE_GenerateRandom** (void *randomBuffer, uint32_t randomBufferLen)
Crypto, common.

Variables

- static int **wc_rng_init** = 0
- static WC_RNG **rngstr**

10.30.1 Function Documentation

10.30.1.1 flags2flags() static int flags2flags (
int flags) [inline], [static]

flags2flags() - Checks the status for reading or writing of the file operational.

This function is used to check the status for reading or writing of the file operational.

Parameters

<i>flags</i>	Flags of the referencing node.
--------------	--------------------------------

Returns

ret if success.

10.30.1.2 get_wc_rng() WC_RNG * get_wc_rng (
void)

[get_wc_rng\(\)](#) - Gets the seed (from OS) and key cipher for rng (random number genertor).

This function returns the random number or unique number of "rngstr".

Returns

random number if success else error occured.

10.30.1.3 GetRelTimeEnd() TEE_Result GetRelTimeEnd (
uint64_t end)

Core Functions, Time Functions.

[GetRelTimeEnd\(\)](#) - finds the real time of the end timing.

This function prints the ending time.

Parameters

<i>end</i>	End timing
------------	------------

Returns

0 If success

10.30.1.4 GetRelTimeStart() TEE_Result GetRelTimeStart (
uint64_t start)

Core Functions, Time Functions.

[GetRelTimeStart\(\)](#) - Gets the real time of the start timing.

This function prints the starting time.

Parameters

<i>start</i>	Start timing
--------------	--------------

Returns

0 on success

10.30.1.5 OpenPersistentObject() static `TEE.Result` OpenPersistentObject (
 uint32_t storageID,
 const void * objectID,
 uint32_t objectIDLen,
 uint32_t flags,
 TEE.ObjectHandle * object,
 int ocreat) [static]

`OpenPersistentObject()` - Opens a handle on an existing persistent object.

The flags parameter is a set of flags that controls the access rights and sharing permissions with which the object handle is opened. The value of the flags parameter is constructed by a bitwise-inclusive OR of flags TEE_DATA_FLAG_ACCESS_READ, the object is opened with the read access right. This allows the Trusted Application to call the function TEE.ReadObjectData. TEE_DATA_FLAG_ACCESS_WRITE, the object is opened with the write access right. TEE_DATA_FLAG_ACCESS_WRITE_META, the object is opened with the write-meta access right.

Parameters

<i>storageID</i>	The storage to use.
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	length of the identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion.

Returns

0 if success else error occurred.

10.30.1.6 set_object_key() static int set_object_key (
 void * id,
 unsigned int idlen,
 TEE.ObjectHandle object) [static]

`set_object_key()` - Initialize report and then attest enclave with file.

This function describes the initialization of report, attest the enclave with file id and its length then assigned to ret. Based on "mbedtls" key encryption and decryption position of the object will be copied. Finally ret value returns on success else signature too short error will appear on failure.

Parameters

<i>id</i>	id of the object.
<i>idlen</i>	length of the id.
<i>object</i>	TEE_ObjectHandle type handle.

Returns

ret if success.

10.30.1.7 TEE_CloseObject() `void TEE_CloseObject (`
`TEE_ObjectHandle object)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_CloseObject\(\)](#) - Closes an opened object handle.

The object can be persistent or transient. For transient objects, TEE_CloseObject is equivalent to TEE_Free↔TransientObject.

Parameters

<i>object</i>	Handle of the object.
---------------	-----------------------

Returns

TEE_SUCCESS if success else error occurred.

10.30.1.8 TEE_CreatePersistentObject() `TEE_Result TEE_CreatePersistentObject (`
`uint32_t storageID,`
`const void * objectID,`
`uint32_t objectIDLen,`
`uint32_t flags,`
`TEE_ObjectHandle attributes,`
`const void * initialData,`
`uint32_t initialDataLen,`
`TEE_ObjectHandle * object)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_CreatePersistentObject\(\)](#) - Creates a persistent object with initial attributes.

In this function an initial data stream content returns either a handle on the created object or TEE_HANDLE_NULL upon failure.

Parameters

<i>storageID</i>	The storage to use.
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>attributes</i>	A handle on a persistent object or an initialized transient object from which to take the persistent object attributes
<i>initialData</i>	The initial data content of the persistent object
<i>initialDataLen</i>	The initial data content of the persistent object
<i>object</i>	A pointer to the handle which contains the opened handle upon successful completion

Returns

0 if success else error occurred.

10.30.1.9 TEE.Free() `void TEEFree (`
`void * buffer)`

[TEE.Free\(\)](#) - causes the space pointed to by *buffer* to be deallocated; that is made available for further allocation.

This function describes if *buffer* is a NULL pointer, `TEE.Free` does nothing. Otherwise, it is a Programmer Error if the argument does not match a pointer previously returned by the `TEE.Malloc` or `TEE.Realloc` if the space has been deallocated by a call to `TEE.Free` or `TEE.Realloc`.

Parameters

<i>buffer</i>	The pointer to the memory block to be freed.
---------------	--

10.30.1.10 TEE.GenerateRandom() `void TEEGenerateRandom (`
`void * randomBuffer,`
`uint32_t randomBufferLen)`

Crypto, common.

[TEE.GenerateRandom\(\)](#) - Generates random data.

This function generates random data of random buffer length and is stored in to random Buffer by calling `wc_RNG↵_GenerateBlock()`. If *ret* is not equal to 0 then `TEE.Panic` is called.

Parameters

<i>randomBuffer</i>	Reference to generated random data
<i>randomBufferLen</i>	Byte length of requested random data

Returns

random data random data will be returned.

10.30.1.11 TEE_GetObjectInfo1() `TEE_Result TEE_GetObjectInfo1 (`
 `TEE_ObjectHandle object,`
 `TEE_ObjectInfo * objectInfo)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_GetObjectInfo1\(\)](#) - Returns the characteristics of an object.

This function returns a handle which can be used to access the object's attributes and data stream.

Parameters

<i>objectInfo</i>	Pointer to a structure filled with the object information
<i>object</i>	Handle of the object

Returns

0 if success else error occurred.

10.30.1.12 TEE_GetREETime() `void TEE_GetREETime (`
 `TEE_Time * time)`

Core Functions, Time Functions.

[TEE_GetREETime\(\)](#) - Retrieves the current REE system time.

This function retrieves the current time as seen from the point of view of the REE.

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds
-------------	--

10.30.1.13 TEE_GetSystemTime() `void TEE_GetSystemTime (`
`TEE_Time * time)`

Core Functions, Time Functions.

[TEE_GetSystemTime\(\)](#) - Retrieves the current system time.

This function describes the system time has an arbitrary implementation defined origin that can vary across TA instances. The minimum guarantee is that the system time shall be monotonic for a given TA instance.

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds
-------------	--

10.30.1.14 TEE_Malloc() `void * TEE_Malloc (`
`uint32_t size,`
`uint32_t hint)`

[TEE_Malloc\(\)](#) - Allocates space for an object whose size in bytes is specified in the parameter size.

This function describes the pointer returned is guaranteed to be aligned such that it may be assigned as a pointer to any basic C type. The valid hint values are a bitmask and can be independently set. This parameter allows Trusted Applications to refer to various pools of memory or to request special characteristics for the allocated memory by using an implementation-defined hint. Future versions of this specification may introduce additional standard hints.

Parameters

<i>size</i>	The size of the buffer to be allocated.
<i>hint</i>	A hint to the allocator.

Returns

Upon successful completion, with size not equal to zero, the function returns a pointer to the allocated space.

10.30.1.15 TEE_OpenPersistentObject() `TEE_Result TEE_OpenPersistentObject (`
`uint32_t storageID,`
`const void * objectID,`
`uint32_t objectIDLen,`
`uint32_t flags,`
`TEE_ObjectHandle * object)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_OpenPersistentObject\(\)](#) - Opens a handle on an existing persistent object.

This function returns a handle which can be used to access the object's attributes and data stream.

Parameters

<i>storageID</i>	The storage to use
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion

Returns

0 if success else error occurred.

10.30.1.16 TEE_ReadObjectData() `TEE_Result TEE_ReadObjectData (`
 `TEE_ObjectHandle object,`
 `void * buffer,`
 `uint32_t size,`
 `uint32_t * count)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_ReadObjectData\(\)](#) - Attempts to read size bytes from the data stream associated with the object into the buffer pointed to by buffer.

The bytes are read starting at the position in the data stream currently stored in the object handle. The handle's position is incremented by the number of bytes actually read. On completion of TEE_ReadObjectData sets the number of bytes actually read in the "uint32_t" pointed to by count. The value written to *count may be less than size if the number of bytes until the end-of-stream is less than size. It is set to 0 if the position at the start of the read operation is at or beyond the end-of-stream. These are the only cases where *count may be less than size.

Parameters

<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write
<i>count</i>	size of the buffer.

Returns

TEE_SUCCESS if success else error occurred.

10.30.1.17 TEE_Realloc() `void * TEE_Realloc (`
`void * buffer,`
`uint32_t newSize)`

[TEE_Realloc\(\)](#) - Changes the size of the memory object pointed to by *buffer* to the size specified by *new size*.

This function describes the content of the object remains unchanged up to the lesser of the new and old sizes. Space in excess of the old size contains unspecified content. If the new size of the memory object requires movement of the object, the space for the previous instantiation of the object is deallocated. If the space cannot be allocated, the original object remains allocated, and this function returns a NULL pointer.

Parameters

<i>buffer</i>	The pointer to the object to be reallocated.
<i>newSize</i>	The new size required for the object

Returns

Upon successful completion, TEE_Realloc returns a pointer to the (possibly moved) allocated space. If there is not enough available memory, TEE_Realloc returns a NULL pointer and the original buffer is still allocated and unchanged.

10.30.1.18 TEE_WriteObjectData() `TEE_Result TEE_WriteObjectData (`
`TEE_ObjectHandle object,`
`const void * buffer,`
`uint32_t size)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_WriteObjectData\(\)](#) - Writes the buffer data in to persistent objects.

In this function it checks if object is present or not, the encryption/ decryption buffer is taken by calling `mbedtls.aes->.crypt_cbc()` then that buffer data is encrypted and mapped to object. On the base of object creation TEE_SUCCESS appears else TEE_ERROR.GENERIC appears.

Parameters

<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write

Returns

TEE_SUCCESS if success else error occurred.

10.30.1.19 wc_ocall_genseed() `int wc_ocall_genseed (`
 `void * nonce,`
 `uint32_t len)`

[wc_ocall_genseed\(\)](#) To generate random data.

This function describes the return value of random generated data. if generated random value is not equal to length of buffer then panic reason occurs.

Parameters

<i>nonce</i>	pointer of buffer
<i>len</i>	length of the buffer.

Returns

0 on success else error will occur based on panic raised inside trusted application.

10.30.2 Variable Documentation

10.30.2.1 rngstr `WC_RNG rngstr [static]`

10.30.2.2 wc_rng_init `int wc_rng_init = 0 [static]`

`ocall_getrandom()` - For getting random data.

This function describes that the retval is returned based on the size of buffer by calling the functions `ocall_getrandom196` and `ocall_getrandom16`

Parameters

<i>buf</i>	character type buffer
<i>len</i>	size of the buffer
<i>flags</i>	unassigned integer flag

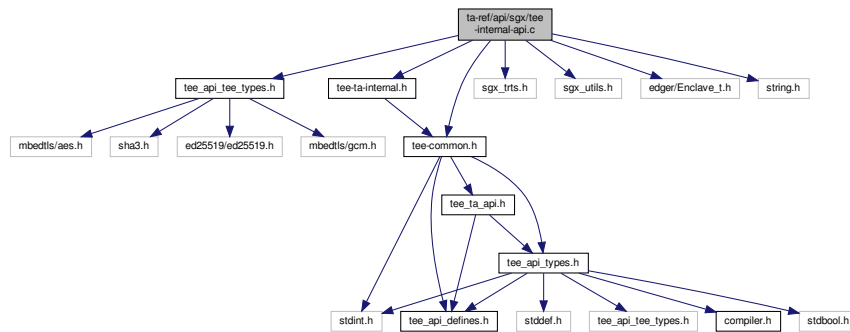
Returns

retval value will be returned based on length of buffer.

10.31 ta-ref/api/sgx/tee-internal-api.c File Reference

```
#include "tee_api_types.h"
#include "tee-common.h"
```

```
#include "tee-ta-internal.h"
#include "sgx_trts.h"
#include "sgx_utils.h"
#include "edger/Enclave_t.h"
#include <string.h>
Include dependency graph for tee-internal-api.c:
```



Functions

- void `__attribute__((noreturn))`
- void `TEE_GetREETime (TEE_Time *time)`
Core Functions, Time Functions.
- void `TEE_GetSystemTime (TEE_Time *time)`
Core Functions, Time Functions.
- `TEE_Result GetRelTimeStart (uint64_t start)`
Core Functions, Time Functions.
- `TEE_Result GetRelTimeEnd (uint64_t end)`
Core Functions, Time Functions.
- static int `flags2flags (int flags)`
- static int `set_object_key (const void *id, unsigned int idlen, TEE_ObjectHandle object)`
- static `TEE_Result OpenPersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle *object, int ocreat)`
- `TEE_Result TEE_CreatePersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle attributes, const void *initialData, uint32_t initialDataLen, TEE_ObjectHandle *object)`
Core Functions, Secure Storage Functions (data is isolated for each TA)
- `TEE_Result TEE_OpenPersistentObject (uint32_t storageID, const void *objectID, uint32_t objectIDLen, uint32_t flags, TEE_ObjectHandle *object)`
Core Functions, Secure Storage Functions (data is isolated for each TA)
- `TEE_Result TEE_GetObjectInfo1 (TEE_ObjectHandle object, TEE_ObjectInfo *objectInfo)`
Core Functions, Secure Storage Functions (data is isolated for each TA)
- `TEE_Result TEE_WriteObjectData (TEE_ObjectHandle object, const void *buffer, uint32_t size)`
Core Functions, Secure Storage Functions (data is isolated for each TA)
- `TEE_Result TEE_ReadObjectData (TEE_ObjectHandle object, void *buffer, uint32_t size, uint32_t *count)`
Core Functions, Secure Storage Functions (data is isolated for each TA)
- void `TEE_CloseObject (TEE_ObjectHandle object)`
Core Functions, Secure Storage Functions (data is isolated for each TA)
- void `TEE_GenerateRandom (void *randomBuffer, uint32_t randomBufferLen)`
Crypto, common.
- static `WC_RNG * get_wc_rng (void)`

Variables

- static int `wc_rng_init` = 0
- static WC_RNG `rngstr`

10.31.1 Function Documentation

10.31.1.1 `__attribute__((noret)) void __attribute__((noret))` (

TEE_Panic() - Raises a Panic in the Trusted Application instance

When a Trusted Application calls the TEE_Panic function, the current instance shall be destroyed and all the resources opened by the instance shall be reclaimed.

Parameters

<code>ec</code>	An informative panic code defined by the TA. May be displayed in traces if traces are available.
-----------------	--

10.31.1.2 `flags2flags() static int flags2flags (`
`int flags) [inline], [static]`

`flags2flags()` - Checks the status for reading or writing of the file operational.

This function is to check the status for reading or writing of the file operational.

Parameters

<code>flags</code>	Flags of the referencing node.
--------------------	--------------------------------

Returns

0 if success else error occurred.

10.31.1.3 `get_wc_rng() static WC_RNG * get_wc_rng (`
`void) [static]`

`get_wc_rng()` - Gets the seed (from OS) and key cipher for rng(random number genertor).

This function returns the random number or unique number of "rngstr".

Returns

random number if success else error occurred.

10.31.1.4 GetRelTimeEnd() `TEE_Result GetRelTimeEnd (`
`uint64_t end)`

Core Functions, Time Functions.

[GetRelTimeStart\(\)](#) - find the real time of the end timing.

This function prints the End timing.

Parameters

<i>end</i>	End timing
------------	------------

Returns

0 if success else error occurred

10.31.1.5 GetRelTimeStart() `TEE_Result GetRelTimeStart (`
`uint64_t start)`

Core Functions, Time Functions.

[GetRelTimeStart\(\)](#) - Gets the real time of the start timing.

This function prints the start timing.

Parameters

<i>start</i>	start timing
--------------	--------------

Returns

0 if success else error occurred.

10.31.1.6 OpenPersistentObject() `static TEE_Result OpenPersistentObject (`
`uint32_t storageID,`
`const void * objectID,`

```
uint32_t objectIDLen,
uint32_t flags,
TEE_ObjectHandle * object,
int ocreat ) [static]
```

OpenPersistentObject() - Opens a handle on an existing persistent object.

The flags parameter is a set of flags that controls the access rights and sharing permissions with which the object handle is opened. The value of the flags parameter is constructed by a bitwise-inclusive OR of flags TEE_DATA_FLAG_ACCESS_READ, the object is opened with the read access right. This allows the Trusted Application to call the function TEE_ReadObjectData. TEE_DATA_FLAG_ACCESS_WRITE, the object is opened with the write access right. TEE_DATA_FLAG_ACCESS_WRITE_META, the object is opened with the write-meta access right.

Parameters

<i>storageID</i>	The storage to use.
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	length of the identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion.

Returns

0 if success else error occurred.

10.31.1.7 set_object_key() static int set_object_key (
const void * id,
unsigned int idlen,
TEE_ObjectHandle object) [static]

set_object_key - To initialize report and then attest enclave with file.

This function describes objectID as key_id to make the key dependent on it sgx report key is 128-bit. Fill another 128-bit with seal key. seal key doesn't change with enclave. Better than nothing, though. random nonce can not use for AES here because of persistency. the digest of attestation report and objectID as the last resort has been used.

Parameters

<i>id</i>	id of the object.
<i>idlen</i>	length of the id.
<i>object</i>	TEE_ObjectHandle type handle.

Returns

0 if success else error occurred.

10.31.1.8 TEE_CloseObject() `void TEE_CloseObject (`
`TEE_ObjectHandle object)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_CloseObject\(\)](#) - Function closes an opened object handle.

The object can be persistent or transient. For transient objects, TEE_CloseObject is equivalent to TEE_Free↔TransientObject.

Parameters

<i>object</i>	Handle of the object
---------------	----------------------

Returns

TEE_SUCCESS if success else error occurred.

10.31.1.9 TEE_CreatePersistentObject() `TEE_Result TEE_CreatePersistentObject (`
`uint32_t storageID,`
`const void * objectID,`
`uint32_t objectIDLen,`
`uint32_t flags,`
`TEE_ObjectHandle attributes,`
`const void * initialData,`
`uint32_t initialDataLen,`
`TEE_ObjectHandle * object)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_CreatePersistentObject\(\)](#) - Creates a persistent object with initial attributes.

An initial data stream content, and optionally returns either a handle on the created object, or TEE_HANDLE_NULL upon failure.

Parameters

<i>storageID</i>	The storage to use.
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>attributes</i>	A handle on a persistent object or an initialized transient object from which to take the persistent object attributes
<i>initialData</i>	The initial data content of the persistent object
<i>initialDataLen</i>	The initial data content of the persistent object
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion

Returns

0 if success, else error occurred.

10.31.1.10 TEE_GenerateRandom() `void TEE_GenerateRandom (`
`void * randomBuffer,`
`uint32_t randomBufferLen)`

Crypto, common.

[TEE_GenerateRandom\(\)](#) - Generates random data.

This function generates random data of random bufferlength and is stored in to randomBuffer by calling `sgx_read_rand()`.

Parameters

<i>randomBuffer</i>	Reference to generated random data
<i>randomBufferLen</i>	Byte length of requested random data

10.31.1.11 TEE_GetObjectInfo1() `TEE_Result TEE_GetObjectInfo1 (`
`TEE_ObjectHandle object,`
`TEE_ObjectInfo * objectInfo)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_GetObjectInfo1\(\)](#) - Function returns the characteristics of an object.

It returns a handle that can be used to access the object's attributes and data stream.

Parameters

<i>objectInfo</i>	Pointer to a structure filled with the object information
<i>object</i>	Handle of the object

Returns

0 if success else error occurred.

10.31.1.12 TEE_GetREETime() `void TEE_GetREETime (`
`TEE_Time * time)`

Core Functions, Time Functions.

[TEE_GetREETime\(\)](#) - Function retrieves the current REE system time.

This function retrieves the current time as seen from the point of view of the REE.

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds.
-------------	---

10.31.1.13 TEE_GetSystemTime() `void TEE_GetSystemTime (
 TEE_Time * time)`

Core Functions, Time Functions.

[TEE_GetSystemTime\(\)](#) - Retrieves the current system time.

The system time has an arbitrary implementation-defined origin that can vary across TA instances

Parameters

<i>time</i>	Filled with the number of seconds and milliseconds.
-------------	---

10.31.1.14 TEE_OpenPersistentObject() `TEE_Result TEE_OpenPersistentObject (
 uint32_t storageID,
 const void * objectID,
 uint32_t objectIDLen,
 uint32_t flags,
 TEE_ObjectHandle * object)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_OpenPersistentObject\(\)](#) - Opens a handle on an existing persistent object.

This function returns a handle that can be used to access the object's attributes and data stream.

Parameters

<i>storageID</i>	The storage to use.
<i>objectID</i>	The object identifier
<i>objectIDLen</i>	The object identifier
<i>flags</i>	The flags which determine the settings under which the object is opened.
<i>object</i>	A pointer to the handle, which contains the opened handle upon successful completion

Returns

0 if success, else error occurred.

10.31.1.15 TEE_ReadObjectData() `TEE_Result TEE_ReadObjectData (`
 `TEE_ObjectHandle object,`
 `void * buffer,`
 `uint32_t size,`
 `uint32_t * count)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_ReadObjectData\(\)](#) - Attempts to read size bytes from the data stream associated with the object object into the buffer pointed to by buffer.

The bytes are read starting at the position in the data stream currently stored in the object handle. The handle's position is incremented by the number of bytes actually read. On completion TEE_ReadObjectData sets the number of bytes actually read in the uint32_t pointed to by count. The value written to *count may be less than size if the number of bytes until the end-of-stream is less than size. It is set to 0 if the position at the start of the read operation is at or beyond the end-of-stream. These are the only cases where *count may be less than size.

Parameters

<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write
<i>count</i>	size of the buffer.

Returns

TEE_SUCCESS if success, else error occurred.

10.31.1.16 TEE_WriteObjectData() `TEE_Result TEE_WriteObjectData (`
 `TEE_ObjectHandle object,`
 `const void * buffer,`
 `uint32_t size)`

Core Functions, Secure Storage Functions (data is isolated for each TA)

[TEE_WriteObjectData\(\)](#) - writes size bytes from the buffer pointed to by buffer to the data stream associated with the open object handle object.

If the current data position points before the end-of-stream, then size bytes are written to the data stream, overwriting bytes starting at the current data position. If the current data position points beyond the stream's end, then the data stream is first extended with zero bytes until the length indicated by the data position indicator is reached, and then size bytes are written to the stream.

Parameters

<i>object</i>	Handle of the object
<i>buffer</i>	The buffer containing the data to be written
<i>size</i>	The number of bytes to write

Returns

TEE_SUCCESS if success else error occurred.

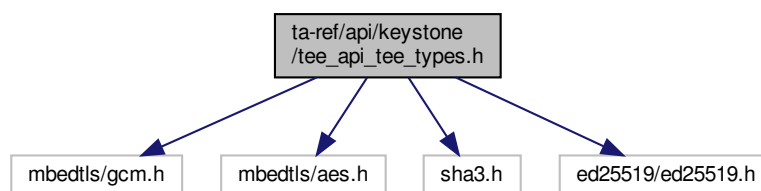
10.31.2 Variable Documentation

10.31.2.1 rngstr WC_RNG rngstr [static]

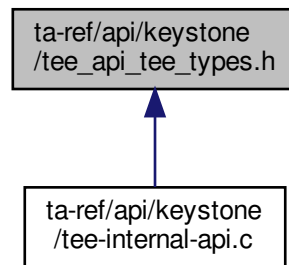
10.31.2.2 wc_rng_init int wc_rng_init = 0 [static]

10.32 ta-ref/api/keystone/tee_api_tee_types.h File Reference

```
#include "mbedtls/gcm.h"
#include "mbedtls/aes.h"
#include "sha3.h"
#include "ed25519/ed25519.h"
Include dependency graph for tee_api_tee_types.h:
```



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



Classes

- struct `__TEE_OperationHandle`
- struct `__TEE_ObjectHandle`

10.33 tee_api_tee_types.h

[Go to the documentation of this file.](#)

```

1 /*
2  * SPDX-License-Identifier: BSD-2-Clause
3  *
4  * Copyright (C) 2019 National Institute of Advanced Industrial Science
5  *                               and Technology (AIST)
6  * All rights reserved.
7  *
8  * Redistribution and use in source and binary forms, with or without
9  * modification, are permitted provided that the following conditions are met:
10 *
11 * 1. Redistributions of source code must retain the above copyright notice,
12 * this list of conditions and the following disclaimer.
13 *
14 * 2. Redistributions in binary form must reproduce the above copyright notice,
15 * this list of conditions and the following disclaimer in the documentation
16 * and/or other materials provided with the distribution.
17 *
18 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
22 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
28 * POSSIBILITY OF SUCH DAMAGE.
29 */
30
31 #ifndef TEE_API_TYPES_KESTONE_H
32 #define TEE_API_TYPES_KESTONE_H
33
34 #ifndef DOXYGEN_SHOULD_SKIP_THIS
35 #define MBEDCRYPT 1
36 #define WOLFCRYPT 2
37 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
38
39 #if CRYPTLIB==MBEDCRYPT
40 #ifndef DOXYGEN_SHOULD_SKIP_THIS
41 #define MBEDTLS_CONFIG_FILE "mbed-crypto-config.h"
42 #define AES256 1
43 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/

```



```

44 # include "mbedtls/gcm.h"
45 # include "mbedtls/aes.h"
46 # include "sha3.h"
47 # include "ed25519/ed25519.h"
48 #elif CRYPTLIB==WOLFCRYPT
49 #ifndef DOXYGEN_SHOULD_SKIP_THIS
50 # define HAVE_AESGCM 1
51 # define HAVE_AES_CBC 1
52 # define HAVE_AES_DECRYPT 1
53 # define HAVE_FIPS 1
54 # define HAVE_FIPS_VERSION 2
55 # define HAVE_ED25519 1
56 # define HAVE_ED25519_SIGN 1
57 # define HAVE_ED25519_VERIFY 1
58 # define WOLFSSL_SHA512 1
59 # define WOLFSSL_SHA3 1
60 # define WOLFSSL_SHA3_SMALL 1
61 # define WOLFCRYPT_ONLY 1
62 # define WOLF_CRYPT_PORT_H
63 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
64 # include "wolfssl/wolfcrypt/sha3.h"
65 # include "wolfssl/wolfcrypt/aes.h"
66 # include "wolfssl/wolfcrypt/sha512.h"
67 # include "wolfssl/wolfcrypt/ed25519.h"
68 #else
69 # include "sha3.h"
70 # include "ed25519/ed25519.h"
71 # include "tiny_AES.c/aes.h"
72 #ifndef DOXYGEN_SHOULD_SKIP_THIS
73 # define AES256 1
74 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
75 #endif
76
77 #ifndef DOXYGEN_SHOULD_SKIP_THIS
78 #define SHA_LENGTH (256/8)
79 #define TEE_OBJECT_NONCE_SIZE 16
80 #define TEE_OBJECT_KEY_SIZE 32
81 #define TEE_OBJECT_SKEY_SIZE 64
82 #define TEE_OBJECT_AAD_SIZE 16
83 #define TEE_OBJECT_TAG_SIZE 16
84 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
85
86 struct __TEEOperationHandle
87 {
88     int mode;
89     int flags;
90     int alg;
91     #if CRYPTLIB==MBEDCRYPT
92     sha3_ctx_t ctx;
93     mbedtls_aes_context aectx;
94     mbedtls_gcm_context aegmctx;
95     #elif CRYPTLIB==WOLFCRYPT
96     wcSha3 ctx;
97     Aes aectx;
98     Aes aegmctx;
99     unsigned int aegm_aadsz;
100     unsigned char aegm_aad[TEE_OBJECT_AAD_SIZE];
101     ed25519_key key;
102     #else
103     sha3_ctx_t ctx;
104     struct AES_ctx aectx;
105     #endif
106     int aegm_state;
107     unsigned char aeiv[TEE_OBJECT_NONCE_SIZE];
108     unsigned char aekey[32];
109     unsigned char pubkey[TEE_OBJECT_KEY_SIZE];
110     unsigned char prikey[TEE_OBJECT_SKEY_SIZE];
111 };
112
113 struct __TEEObjectHandle
114 {
115     unsigned int type;
116     int flags;
117     int desc;
118     #if CRYPTLIB==MBEDCRYPT
119     mbedtls_aes_context persist_ctx;
120     unsigned char persist_iv[TEE_OBJECT_NONCE_SIZE];
121     #elif CRYPTLIB==WOLFCRYPT
122     Aes persist_ctx;
123     unsigned char persist_iv[TEE_OBJECT_NONCE_SIZE];
124     ed25519_key key;
125     #else
126     struct AES_ctx persist_ctx;
127     #endif
128     unsigned char public_key[TEE_OBJECT_KEY_SIZE];
129     unsigned char private_key[TEE_OBJECT_SKEY_SIZE];
130 };

```

```

131
132 // defined in tee_api_defines.h
133 // enum Data.Flag.Constants {
134 //     TEE_DATA_FLAG_ACCESS_READ = 0x00000001,
135 //     TEE_DATA_FLAG_ACCESS_WRITE = 0x00000002,
136 //     //TEE_DATA_FLAG_ACCESS_WRITE_META = 0x00000004,
137 //     //TEE_DATA_FLAG_SHARE_READ = 0x00000010,
138 //     //TEE_DATA_FLAG_SHARE_WRITE = 0x00000020,
139 //     TEE_DATA_FLAG_OVERWRITE = 0x00000400
140 // };
141 // enum Data.Flag.Constants {
142 //     TEE_DATA_FLAG_ACCESS_READ = 0x00000001,
143 //     TEE_DATA_FLAG_ACCESS_WRITE = 0x00000002,
144 //     //TEE_DATA_FLAG_ACCESS_WRITE_META = 0x00000004,
145 //     //TEE_DATA_FLAG_SHARE_READ = 0x00000010,
146 //     //TEE_DATA_FLAG_SHARE_WRITE = 0x00000020,
147 //     TEE_DATA_FLAG_OVERWRITE = 0x00000400
148 // };
149 #endif

```

10.34 ta-ref/api/optee/tee_api_tee.types.h File Reference

10.35 tee_api_tee.types.h

[Go to the documentation of this file.](#)

```
1 // empty
```

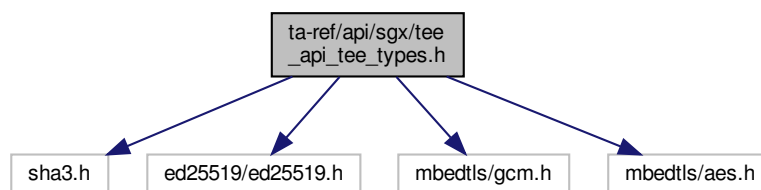
10.36 ta-ref/api/sgx/tee_api_tee.types.h File Reference

```

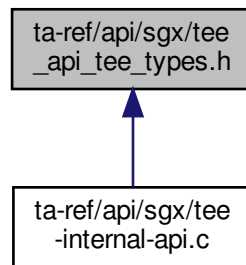
#include "sha3.h"
#include "ed25519/ed25519.h"
#include "mbedtls/gcm.h"
#include "mbedtls/aes.h"

```

Include dependency graph for tee_api_tee.types.h:



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



Classes

- struct `__TEE_OperationHandle`
- struct `__TEE_ObjectHandle`

10.37 tee_api_tee_types.h

[Go to the documentation of this file.](#)

```

1 /*
2  * SPDX-License-Identifier: BSD-2-Clause
3  *
4  * Copyright (C) 2019 National Institute of Advanced Industrial Science
5  *                               and Technology (AIST)
6  * All rights reserved.
7  *
8  * Redistribution and use in source and binary forms, with or without
9  * modification, are permitted provided that the following conditions are met:
10 *
11 * 1. Redistributions of source code must retain the above copyright notice,
12 * this list of conditions and the following disclaimer.
13 *
14 * 2. Redistributions in binary form must reproduce the above copyright notice,
15 * this list of conditions and the following disclaimer in the documentation
16 * and/or other materials provided with the distribution.
17 *
18 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
22 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
28 * POSSIBILITY OF SUCH DAMAGE.
29 */
30
31 #ifndef TEE_API_TYPES_KEystone_H
32 #define TEE_API_TYPES_KEystone_H
33
34 #ifndef DOXYGEN_SHOULD_SKIP_THIS
35 #define MBEDCRYPT 1
36 #define WOLFCRYPT 2
37 #define SHA_LENGTH (256/8)
38 #define AES256 1
39 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
40
41 #include "sha3.h"
42 #include "ed25519/ed25519.h"
43

```

```

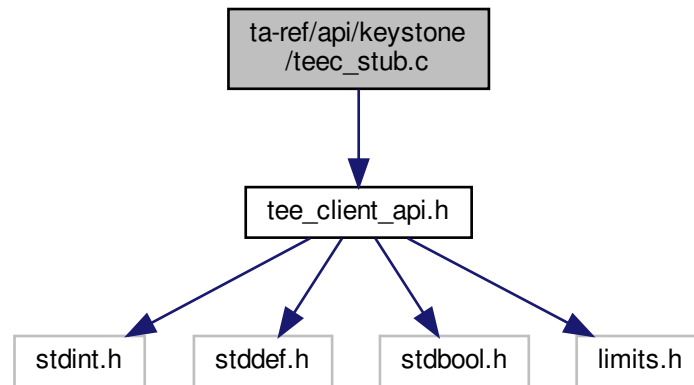
44 #if CRYPTLIB==MBEDCRYPT
45 #ifndef DOXYGEN_SHOULD_SKIP_THIS
46 # define MBEDTLS_CONFIG_FILE "mbed-crypto-config.h"
47 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
48 # include "mbedtls/gcm.h"
49 # include "mbedtls/aes.h"
50 #elif CRYPTLIB==WOLFCRYPT
51 #ifndef DOXYGEN_SHOULD_SKIP_THIS
52 # define HAVE_AESGCM 1
53 # define HAVE_AES_CBC 1
54 # define HAVE_AES_DECRYPT 1
55 # define HAVE_FIPS 1
56 # define HAVE_FIPS_VERSION 2
57 # define HAVE_ED25519 1
58 # define HAVE_ED25519_SIGN 1
59 # define HAVE_ED25519_VERIFY 1
60 # define WOLFSSL_SHA3 1
61 # define WOLF_CRYPT_PORT_H
62 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
63 # include "wolfssl/wolfcrypt/sha3.h"
64 # include "wolfssl/wolfcrypt/aes.h"
65 # include "wolfssl/wolfcrypt/sha512.h"
66 # include "wolfssl/wolfcrypt/ed25519.h"
67 #else
68 # include "tiny_AES.c/aes.h"
69 #endif
70
71 #ifndef DOXYGEN_SHOULD_SKIP_THIS
72 #define TEE_OBJECT_NONCE_SIZE 16
73 #define TEE_OBJECT_KEY_SIZE 32
74 #define TEE_OBJECT_SKEY_SIZE 64
75 #define TEE_OBJECT_AAD_SIZE 16
76 #define TEE_OBJECT_TAG_SIZE 16
77 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
78
79 struct __TEEOperationHandle
80 {
81     int mode;
82     int flags;
83     int alg;
84     #if CRYPTLIB==MBEDCRYPT
85         sha3_ctx_t ctx;
86         mbedtls_aes_context aectx;
87         mbedtls_gcm_context aegcmctx;
88     #elif CRYPTLIB==WOLFCRYPT
89         wc_Sha3 ctx;
90         Aes aectx;
91         Aes aegcmctx;
92         unsigned int aegcm_aadsz;
93         unsigned char aegcm_aad[TEE_OBJECT_AAD_SIZE];
94         ed25519_key key;
95     #else
96         sha3_ctx_t ctx;
97         struct AES_ctx aectx;
98     #endif
99     int aegcm_state;
100     unsigned char aeiv[TEE_OBJECT_NONCE_SIZE];
101     unsigned char aekey[32];
102     unsigned char pubkey[TEE_OBJECT_KEY_SIZE];
103     unsigned char prikey[TEE_OBJECT_SKEY_SIZE];
104 };
105
106 struct __TEEObjectHandle
107 {
108     unsigned int type;
109     int flags;
110     int desc;
111     #if CRYPTLIB==MBEDCRYPT
112         mbedtls_aes_context persist_ctx;
113         unsigned char persist_iv[TEE_OBJECT_NONCE_SIZE];
114     #elif CRYPTLIB==WOLFCRYPT
115         Aes persist_ctx;
116         unsigned char persist_iv[TEE_OBJECT_NONCE_SIZE];
117         ed25519_key key;
118     #else
119         struct AES_ctx persist_ctx;
120     #endif
121     unsigned char public_key[TEE_OBJECT_KEY_SIZE];
122     unsigned char private_key[TEE_OBJECT_SKEY_SIZE];
123 };
124
125 // Minimal constant definitions
126 #ifndef DOXYGEN_SHOULD_SKIP_THIS
127 #define TEE_HANDLE_NULL 0
128 #endif /*DOXYGEN_SHOULD_SKIP_THIS*/
129
130 #endif

```

10.38 ta-ref/api/keystone/tee_client_api.h File Reference

```
#include <tee_client_api.h>
```

Include dependency graph for tee_client_api.h:



Functions

- [TEEC_Result TEEC_InitializeContext](#) (const char *name, [TEEC_Context](#) *context)
- void [TEEC_FinalizeContext](#) ([TEEC_Context](#) *context)
- [TEEC_Result TEEC_OpenSession](#) ([TEEC_Context](#) *context, [TEEC_Session](#) *session, const [TEEC_UUID](#) *destination, uint32_t connectionMethod, const void *connectionData, [TEEC_Operation](#) *operation, uint32_t *returnOrigin)
- void [TEEC_CloseSession](#) ([TEEC_Session](#) *session)
- [TEEC_Result TEEC_RegisterSharedMemory](#) ([TEEC_Context](#) *context, [TEEC_SharedMemory](#) *sharedMem)
- [TEEC_Result TEEC_AllocateSharedMemory](#) ([TEEC_Context](#) *context, [TEEC_SharedMemory](#) *sharedMem)
- void [TEEC_ReleaseSharedMemory](#) ([TEEC_SharedMemory](#) *sharedMemory)
- void [TEEC_RequestCancellation](#) ([TEEC_Operation](#) *operation)

10.38.1 Function Documentation

10.38.1.1 TEEC_AllocateSharedMemory() [TEEC_Result](#) TEEC_AllocateSharedMemory (
[TEEC_Context](#) * context,
[TEEC_SharedMemory](#) * sharedMem)

[TEEC_AllocateSharedMemory\(\)](#) - Allocate shared memory for TEE.

Parameters

<i>context</i>	The initialized TEE context structure in which scope to open the session.
<i>sharedMem</i>	Pointer to the allocated shared memory.

Returns

TEEC_SUCCESS The registration was successful.

TEEC_ERROR_OUT_OF_MEMORY Memory exhaustion.

TEEC_Result Something failed.

10.38.1.2 TEEC.CloseSession() `void TEEC_CloseSession (`
`TEEC_Session * session)`

[TEEC.CloseSession\(\)](#) - Closes the session which has been opened with the specific trusted application.

Parameters

<i>session</i>	The opened session to close.
----------------	------------------------------

10.38.1.3 TEEC.FinalizeContext() `void TEEC_FinalizeContext (`
`TEEC_Context * context)`

[TEEC.FinalizeContext\(\)](#) - Destroys a context holding connection information on the specific TEE.

This function finalizes an initialized TEE context, closing the connection between the client application and the TEE. This function must only be called when all sessions related to this TEE context have been closed and all shared memory blocks have been released.

Parameters

<i>context</i>	The context to be finalized.
----------------	------------------------------

10.38.1.4 TEEC.InitializeContext() `TEEC_Result TEEC_InitializeContext (`
`const char * name,`
`TEEC_Context * context)`

[TEEC.InitializeContext\(\)](#) - Initializes a context holding connection information on the specific TEE, designated by the name string.

Parameters

<i>name</i>	A zero-terminated string identifying the TEE to connect to. If name is set to NULL, the default TEE is connected to. NULL is the only supported value in this version of the API implementation.
<i>context</i>	The context structure which is to be initialized.

Returns

TEEC_SUCCESS The initialization was successful.

TEEC_Result Something failed.

10.38.1.5 TEEC_OpenSession() `TEEC_Result TEEC_OpenSession (`

```
TEEC_Context * context,
TEEC_Session * session,
const TEEC_UUID * destination,
uint32_t connectionMethod,
const void * connectionData,
TEEC_Operation * operation,
uint32_t * returnOrigin )
```

`TEEC_OpenSession()` - Opens a new session with the specified trusted application.

Parameters

<i>context</i>	The initialized TEE context structure in which scope to open the session.
<i>session</i>	The session to initialize.
<i>destination</i>	A structure identifying the trusted application with which to open a session.
<i>connectionMethod</i>	The connection method to use.
<i>connectionData</i>	Any data necessary to connect with the chosen connection method. Not supported, should be set to NULL.
<i>operation</i>	An operation structure to use in the session. May be set to NULL to signify no operation structure needed.
<i>returnOrigin</i>	A parameter which will hold the error origin if this function returns any value other than TEEC_SUCCESS.

Returns

TEEC_SUCCESS OpenSession successfully opened a new session.

TEEC_Result Something failed.

10.38.1.6 TEEC_RegisterSharedMemory() `TEEC_Result TEEC_RegisterSharedMemory (`

```
TEEC_Context * context,
TEEC_SharedMemory * sharedMem )
```

`TEEC_RegisterSharedMemory()` - Register a block of existing memory as a shared block within the scope of the specified context.

Parameters

<i>context</i>	The initialized TEE context structure in which scope to open the session.
<i>sharedMem</i>	pointer to the shared memory structure to register.

Returns

TEEC_SUCCESS The registration was successful.

TEEC_ERROR_OUT_OF_MEMORY Memory exhaustion.

TEEC_Result Something failed.

10.38.1.7 TEEC.ReleaseSharedMemory() `void TEEC_ReleaseSharedMemory (`
`TEEC_SharedMemory * sharedMemory)`

[TEEC.ReleaseSharedMemory\(\)](#) - Free or deregister the shared memory.

Parameters

<i>sharedMem</i>	Pointer to the shared memory to be freed.
------------------	---

10.38.1.8 TEEC.RequestCancellation() `void TEEC_RequestCancellation (`
`TEEC_Operation * operation)`

[TEEC.RequestCancellation\(\)](#) - Request the cancellation of a pending open session or command invocation.

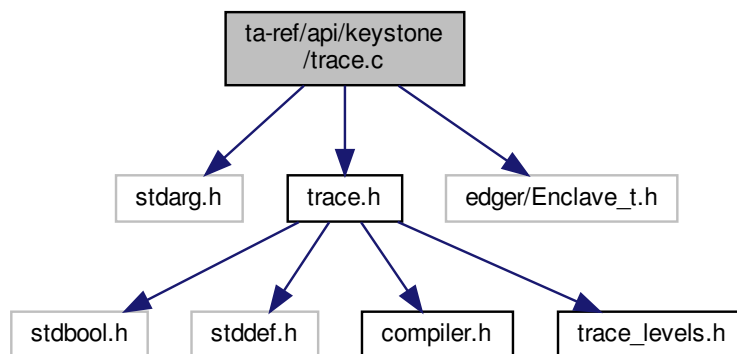
Parameters

<i>operation</i>	Pointer to an operation previously passed to open session or invoke.
------------------	--

10.39 ta-ref/api/keystone/trace.c File Reference

```
#include <stdarg.h>
#include "trace.h"
#include "edger/Enclave.t.h"
```


Include dependency graph for trace.c:



Functions

- void [trace_vprintf](#) (const char *func, int line, int level, bool level_ok, const char *fmt, va_list ap)
- void [trace_printf](#) (const char *func, int line, int level, bool level_ok, const char *fmt,...)

10.39.1 Function Documentation

10.39.1.1 trace_printf() void trace_printf (

```

    const char * func,
    int line,
    int level,
    bool level_ok,
    const char * fmt,
    ... )

```

[trace_printf\(\)](#) - Prints the formatted data to stdout.

This function returns the value of ap by calling va_end().

Parameters

<i>func</i>	Pointer to a buffer where the resulting C-string is stored.
<i>line</i>	integer type of line
<i>level_ok</i>	boolean value
<i>fmt</i>	C string that contains a format string
<i>ap</i>	A value identifying a variable arguments list

Returns

Total number of characters is returned.

10.39.1.2 trace_vprintf() void trace_vprintf (

```

    const char * func,
    int line,
    int level,
    bool level_ok,
    const char * fmt,
    va_list ap )

```

[trace_vprintf\(\)](#) - Writes the formatted data from variable argument list to sized buffer.

This function returns the buffer character by calling ocall_print_string()

Parameters

<i>func</i>	Pointer to a buffer where the resulting C-string is stored.
<i>line</i>	integer type of line
<i>level_ok</i>	boolean value
<i>fmt</i>	C string that contains a format string
<i>ap</i>	A value identifying a variable arguments list

Returns

buf The total number of characters written is returned.

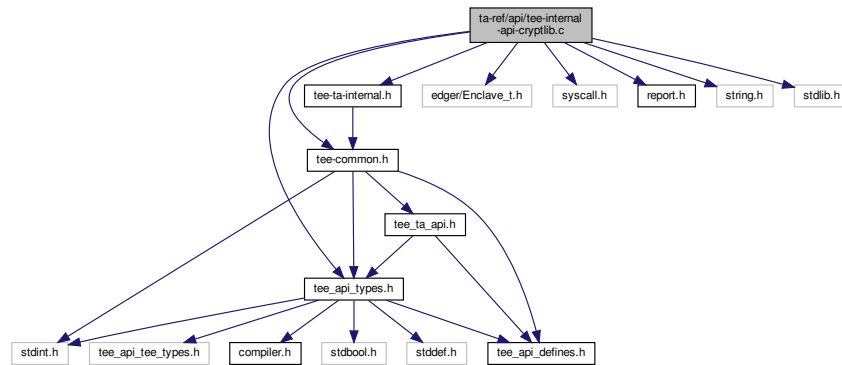
10.40 ta-ref/api/tee-internal-api-cryptlib.c File Reference

```

#include "tee_api_types.h"
#include "tee-common.h"
#include "tee-ta-internal.h"
#include "edger/Enclave_t.h"
#include "syscall.h"
#include "report.h"
#include <string.h>
#include <stdlib.h>

```

Include dependency graph for tee-internal-api-cryptlib.c:



Functions

- void [wolfSSL_Free](#) (void *p)
- void * [wolfSSL_Malloc](#) (size_t n)
- [TEE_Result TEE_AllocateOperation](#) ([TEE.OperationHandle](#) *operation, uint32_t algorithm, uint32_t mode, uint32_t maxKeySize)
Crypto, for all Crypto Functions.
- void [TEE_FreeOperation](#) ([TEE.OperationHandle](#) operation)
Crypto, for all Crypto Functions.
- void [TEE_DigestUpdate](#) ([TEE.OperationHandle](#) operation, const void *chunk, uint32_t chunkSize)
Crypto, Message Digest Functions.
- [TEE_Result TEE_DigestDoFinal](#) ([TEE.OperationHandle](#) operation, const void *chunk, uint32_t chunkLen, void *hash, uint32_t *hashLen)
- [TEE_Result TEE_SetOperationKey](#) ([TEE.OperationHandle](#) operation, [TEE.ObjectHandle](#) key)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- [TEE_Result TEE_AEInit](#) ([TEE.OperationHandle](#) operation, const void *nonce, uint32_t nonceLen, uint32_t tagLen, uint32_t AADLen, uint32_t payloadLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- void [TEE_AEUpdateAAD](#) ([TEE.OperationHandle](#) operation, const void *AADdata, uint32_t AADdataLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- [TEE_Result TEE_AEUpdate](#) ([TEE.OperationHandle](#) operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- [TEE_Result TEE_AEEncryptFinal](#) ([TEE.OperationHandle](#) operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen, void *tag, uint32_t *tagLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- [TEE_Result TEE_AEDecryptFinal](#) ([TEE.OperationHandle](#) operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen, void *tag, uint32_t tagLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- void [TEE_CipherInit](#) ([TEE.OperationHandle](#) operation, const void *nonce, uint32_t nonceLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- [TEE_Result TEE_CipherUpdate](#) ([TEE.OperationHandle](#) operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen)
Crypto, Authenticated Encryption with Symmetric key Verification Functions.
- [TEE_Result TEE_CipherDoFinal](#) ([TEE.OperationHandle](#) operation, const void *srcData, uint32_t srcLen, void *destData, uint32_t *destLen)

- **TEE_Result TEE_GenerateKey** (**TEE_ObjectHandle** object, **uint32_t** keySize, **const TEE_Attribute** *params, **uint32_t** paramCount)
Crypto, Asymmetric key Verification Functions.
- **TEE_Result TEE_AllocateTransientObject** (**TEE_ObjectType** objectType, **uint32_t** maxKeySize, **TEE_ObjectHandle** *object)
Crypto, Asymmetric key Verification Functions.
- **void TEE_InitRefAttribute** (**TEE_Attribute** *attr, **uint32_t** attributeID, **const void** *buffer, **uint32_t** length)
Crypto, Asymmetric key Verification Functions.
- **void TEE_InitValueAttribute** (**TEE_Attribute** *attr, **uint32_t** attributeID, **uint32_t** a, **uint32_t** b)
Crypto, Asymmetric key Verification Functions.
- **void TEE_FreeTransientObject** (**TEE_ObjectHandle** object)
Crypto, Asymmetric key Verification Functions.
- **TEE_Result TEE_AsymmetricSignDigest** (**TEE_OperationHandle** operation, **const TEE_Attribute** *params, **uint32_t** paramCount, **const void** *digest, **uint32_t** digestLen, **void** *signature, **uint32_t** *signatureLen)
Crypto, Asymmetric key Verification Functions.
- **TEE_Result TEE_AsymmetricVerifyDigest** (**TEE_OperationHandle** operation, **const TEE_Attribute** *params, **uint32_t** paramCount, **const void** *digest, **uint32_t** digestLen, **const void** *signature, **uint32_t** signatureLen)
Crypto, Asymmetric key Verification Functions.

10.40.1 Function Documentation

10.40.1.1 TEE_AEDecryptFinal() **TEE_Result** TEE_AEDecryptFinal (
TEE_OperationHandle operation,
const void * srcData,
uint32_t srcLen,
void * destData,
uint32_t * destLen,
void * tag,
uint32_t tagLen)

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

TEE_AEDecryptFinal() - Processes data that has not been processed by previous calls to **TEE_AEUpdate** as well as data supplied in **srcData**.

This function completes the AE operation and compares the computed tag with the tag supplied in the parameter **tag**. The operation handle can be reused or newly initialized. The buffers **srcData** and **destData** shall be either completely disjoint or equal in their starting positions. The operation may be in either initial or active state and enters initial state afterwards.

Parameters

<i>operation</i>	Handle of a running AE operation
<i>srcData</i>	Reference to final chunk of input data to be encrypted
<i>srcLen</i>	length of the input data
<i>destData</i>	Output buffer. Can be omitted if the output is to be discarded.
<i>destLen</i>	length of the buffer.
<i>tag</i>	Output buffer filled with the computed tag
<i>tagLen</i>	length of the tag.

Returns

0 on success.

TEE_ERROR_SHORT_BUFFER If the output buffer is not large enough to contain the output

TEE_ERROR_MAC_INVALID If the computed tag does not match the supplied tag

10.40.1.2 TEE_AEEncryptFinal() `TEE_Result TEE_AEEncryptFinal (`
`TEE_OperationHandle operation,`
`const void * srcData,`
`uint32_t srcLen,`
`void * destData,`
`uint32_t * destLen,`
`void * tag,`
`uint32_t * tagLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

TEE_AEEncryptFinal() - processes data that has not been processed by previous calls to TEE_AEUpdate as well as data supplied in srcData .

TEE_AEEncryptFinal completes the AE operation and computes the tag. The operation handle can be reused or newly initialized. The buffers srcData and destData SHALL be either completely disjoint or equal in their starting positions. The operation may be in either initial or active state and enters initial state afterwards.

Parameters

<i>operation</i>	Handle of a running AE operation
<i>srcData</i>	Reference to final chunk of input data to be encrypted
<i>srcLen</i>	length of the input data
<i>destData</i>	Output buffer. Can be omitted if the output is to be discarded.
<i>destLen</i>	length of the buffer.
<i>tag</i>	Output buffer filled with the computed tag
<i>tagLen</i>	length of the tag.

Returns

0 on success.

TEE_ERROR_SHORT_BUFFER If the output or tag buffer is not large enough to contain the output.

10.40.1.3 TEE_AEInit() `TEE_Result TEE_AEInit (`
`TEE_OperationHandle operation,`
`const void * nonce,`
`uint32_t nonceLen,`
`uint32_t tagLen,`
`uint32_t AADLen,`
`uint32_t payloadLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

[TEE.AEInit\(\)](#) - Initializes an Authentication Encryption operation.

The operation must be in initial state and remains in the initial state afterwards.

Parameters

<i>operation</i>	A handle on the operation.
<i>nonce</i>	The operation nonce or IV
<i>nonceLen</i>	length of nonce
<i>tagLen</i>	Size in bits of the tag
<i>AADLen</i>	Length in bytes of the AAD
<i>payloadLen</i>	Length in bytes of the payload.

Returns

0 on success.

TEE_ERROR_NOT_SUPPORTED If the tag length is not supported by the algorithm.

10.40.1.4 TEE.AEUpdate() `TEE_Result TEE.AEUpdate (`
`TEE.OperationHandle operation,`
`const void * srcData,`
`uint32_t srcLen,`
`void * destData,`
`uint32_t * destLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

[TEE.AEUpdate\(\)](#) - Accumulates data for an Authentication Encryption operation

This function describes Input data does not have to be a multiple of block size. Subsequent calls to this function are possible. Unless one or more calls of this function have supplied sufficient input data, no output is generated. when using this routine to decrypt the returned data may be corrupt since the integrity check is not performed until all the data has been processed. If this is a concern then only use the [TEE.AEDecryptFinal](#) routine.

Parameters

<i>operation</i>	Handle of a running AE operation.
<i>srcData</i>	Input data buffer to be encrypted or decrypted
<i>srcLen</i>	length of the input buffer.
<i>destData</i>	Output buffer
<i>destLen</i>	length of the out put buffer.

Returns

0 on success.

TEE_ERROR_SHORT_BUFFER if the output buffer is not large enough to contain the output.

10.40.1.5 TEE.AEUpdateAAD() void TEE_AEUpdateAAD (
 TEE_OperationHandle operation,
 const void * AADdata,
 uint32_t AADdataLen)

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

[TEE.AEUpdateAAD\(\)](#) - Feeds a new chunk of Additional Authentication Data (AAD) to the AE operation. Subsequent calls to this function are possible.

The TEE_AEUpdateAAD function feeds a new chunk of Additional Authentication Data (AAD) to the AE operation. Subsequent calls to this function are possible. The buffers srcData and destData shall be either completely disjoint or equal in their starting positions. The operation SHALL be in initial state and remains in initial state afterwards.

Parameters

<i>operation</i>	Handle on the AE operation
<i>AADdata</i>	Input buffer containing the chunk of AAD
<i>AADdataLen</i>	length of the chunk of AAD.

10.40.1.6 TEE.AllocateOperation() TEE_Result TEE_AllocateOperation (
 TEE_OperationHandle * operation,
 uint32_t algorithm,
 uint32_t mode,
 uint32_t maxKeySize)

Crypto, for all Crypto Functions.

[TEE.AllocateOperation\(\)](#) - Allocates a handle for a new cryptographic operation and sets the mode and algorithm type.

If this function does not return with TEE_SUCCESS then there is no valid handle value. Once a cryptographic operation has been created, the implementation shall guarantee that all resources necessary for the operation are allocated and that any operation with a key of at most maxKeySize bits can be performed. For algorithms that take multiple keys, for example the AES XTS algorithm, the maxKeySize parameter specifies the size of the largest key. It is up to the implementation to properly allocate space for multiple keys if the algorithm so requires.

Parameters

<i>operation</i>	reference to generated operation handle.
<i>algorithm</i>	One of the cipher algorithms.
<i>mode</i>	The operation mode.
<i>maxKeySize</i>	Maximum key size in bits for the operation.

Returns

0 in case of success

TEE_ERROR_OUT_OF_MEMORY If there are not enough resources to allocate the operation.

TEE_ERROR_NOT_SUPPORTED If the mode is not compatible with the algorithm or key size or if the algorithm is not one of the listed algorithms or if maxKeySize is not appropriate for the algorithm.

10.40.1.7 TEE_AllocateTransientObject() `TEE_Result TEE_AllocateTransientObject (`
`TEE_ObjectType objectType,`
`uint32_t maxKeySize,`
`TEE_ObjectHandle * object)`

Crypto, Asymmetric key Verification Functions.

[TEE_AllocateTransientObject\(\)](#) - Allocates an uninitialized transient object. Transient objects are used to hold a cryptographic object (key or key-pair).

The value TEE_KEYSIZE_NO_KEY should be used for maxObjectSize for object types that do not require a key so that all the container resources can be pre-allocated. As allocated, the container is uninitialized. It can be initialized by subsequently importing the object material, generating an object, deriving an object, or loading an object from the Trusted Storage.

Parameters

<i>objectType</i>	Type of uninitialized object container to be created
<i>maxKeySize</i>	Key Size of the object.
<i>object</i>	Filled with a handle on the newly created key container.

Returns

0 on success

TEE_ERROR_OUT_OF_MEMORY If not enough resources are available to allocate the object handle.

TEE_ERROR_NOT_SUPPORTED If the key size is not supported or the object type is not supported.

10.40.1.8 TEE_AsymmetricSignDigest() `TEE_Result TEE_AsymmetricSignDigest (`
`TEE_OperationHandle operation,`
`const TEE_Attribute * params,`
`uint32_t paramCount,`
`const void * digest,`
`uint32_t digestLen,`
`void * signature,`
`uint32_t * signatureLen)`

Crypto, Asymmetric key Verification Functions.

[TEE_AsymmetricSignDigest\(\)](#) - Signs a message digest within an asymmetric operation.

Parameters

<i>operation</i>	Handle on the operation, which SHALL have been suitably set up with an operation key.
<i>params</i>	Optional operation parameters
<i>paramCount</i>	size of param
<i>digest</i>	Input buffer containing the input message digest
<i>digestLen</i>	length of input buffer.
<i>signature</i>	Output buffer written with the signature of the digest
<i>signatureLen</i>	length of output buffer.

Returns

0 on success

TEE_ERROR_SHORT_BUFFER If the signature buffer is not large enough to hold the result

10.40.1.9 TEE_AsymmetricVerifyDigest() `TEE_Result` TEE_AsymmetricVerifyDigest (
 `TEE_OperationHandle` operation,
 const `TEE_Attribute` * params,
 uint32_t paramCount,
 const void * digest,
 uint32_t digestLen,
 const void * signature,
 uint32_t signatureLen)

Crypto, Asymmetric key Verification Functions.

[TEE_AsymmetricVerifyDigest\(\)](#) - verifies a message digest signature within an asymmetric operation.

This function describes the message digest signature verify by calling ed25519_verify().

Parameters

<i>operation</i>	Handle on the operation, which SHALL have been suitably set up with an operation key.
<i>params</i>	Optional operation parameters
<i>paramCount</i>	size of param.
<i>digest</i>	Input buffer containing the input message digest
<i>digestLen</i>	length of input buffer.
<i>signature</i>	Output buffer written with the signature of the digest
<i>signatureLen</i>	length of output buffer.

Returns

TEE_SUCCESS on success

TEE_ERROR_SIGNATURE_INVALID if the signature is invalid.

10.40.1.10 TEE_CipherDoFinal() `TEE_Result TEE_CipherDoFinal (`
`TEE_OperationHandle operation,`
`const void * srcData,`
`uint32_t srcLen,`
`void * destData,`
`uint32_t * destLen)`

[TEE_CipherDoFinal\(\)](#) - Finalizes the cipher operation, processing data that has not been processed by previous calls to [TEE_CipherUpdate](#) as well as data supplied in `srcData` .

This function describes The operation handle can be reused or re-initialized. The buffers `srcData` and `destData` shall be either completely disjoint or equal in their starting positions. The operation SHALL be in active state and is set to initial state afterwards.

Parameters

<i>operation</i>	Handle of a running Cipher operation
<i>srcData</i>	Input data buffer to be encrypted or decrypted
<i>srcLen</i>	length of input buffer
<i>destData</i>	output buffer
<i>destLen</i>	ouput buffer length.

Returns

0 on success

TEE_ERROR_SHORT_BUFFER If the output buffer is not large enough to contain the output

10.40.1.11 TEE_CipherInit() `void TEE_CipherInit (`
`TEE_OperationHandle operation,`
`const void * nonce,`
`uint32_t nonceLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

[TEE_CipherInit\(\)](#) - starts the symmetric cipher operation.

The operation shall have been associated with a key. If the operation is in active state, it is reset and then initialized. If the operation is in initial state, it is moved to active state.

Parameters

<i>operation</i>	A handle on an opened cipher operation setup with a key
<i>nonce</i>	Buffer containing the operation Initialization Vector as appropriate.
<i>nonceLen</i>	length of the buffer

10.40.1.12 TEE_CipherUpdate() `TEE_Result TEE_CipherUpdate (`
`TEE.OperationHandle operation,`
`const void * srcData,`
`uint32_t srcLen,`
`void * destData,`
`uint32_t * destLen)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

[TEE_CipherUpdate\(\)](#) - encrypts or decrypts input data.

Input data does not have to be a multiple of block size. Subsequent calls to this function are possible. Unless one or more calls of this function have supplied sufficient input data, no output is generated. The cipher operation is finalized with a call to `TEE_CipherDoFinal`. The buffers `srcData` and `destData` SHALL be either completely disjoint or equal in their starting positions. The operation SHALL be in active state.

Parameters

<i>operation</i>	Handle of a running Cipher operation
<i>srcData</i>	Input data buffer to be encrypted or decrypted
<i>srcLen</i>	length of input buffer
<i>destData</i>	output buffer
<i>destLen</i>	output buffer length.

Returns

0 on success else

`TEE_ERROR_SHORT_BUFFER` If the output buffer is not large enough to contain the output. In this case, the input is not fed into the algorithm.

10.40.1.13 TEE_DigestDoFinal() `TEE_Result TEE_DigestDoFinal (`
`TEE.OperationHandle operation,`
`const void * chunk,`
`uint32_t chunkLen,`
`void * hash,`
`uint32_t * hashLen)`

[TEE_DigestDoFinal\(\)](#) - Finalizes the message digest operation and produces the message hash.

This function finalizes the message digest operation and produces the message hash. Afterwards the Message Digest operation is reset to initial state and can be reused.

Parameters

<i>operation</i>	Handle of a running Message Digest operation.
<i>chunk</i>	Chunk of data to be hashed.
<i>chunkLen</i>	size of the chunk.
<i>hash</i>	Output buffer filled with the message hash.
<i>hashLen</i>	length of the message hash.

Returns

0 on success

TEE_ERROR_SHORT_BUFFER If the output buffer is too small. In this case, the operation is not finalized.

10.40.1.14 TEE_DigestUpdate() void TEE_DigestUpdate (
 TEE_OperationHandle operation,
 const void * chunk,
 uint32_t chunkSize)

Crypto, Message Digest Functions.

[TEE_DigestUpdate\(\)](#) - Accumulates message data for hashing.

This function describes the message does not have to be block aligned. Subsequent calls to this function are possible. The operation may be in either initial or active state and becomes active.

Parameters

<i>operation</i>	Handle of a running Message Digest operation.
<i>chunk</i>	Chunk of data to be hashed
<i>chunkSize</i>	size of the chunk.

10.40.1.15 TEE_FreeOperation() void TEE_FreeOperation (
 TEE_OperationHandle operation)

Crypto, for all Crypto Functions.

[TEE_FreeOperation\(\)](#) - Deallocates all resources associated with an operation handle.

This function deallocates all resources associated with an operation handle. After this function is called, the operation handle is no longer valid. All cryptographic material in the operation is destroyed. The function does nothing if operation is TEE_HANDLE_NULL.

Parameters

<i>operation</i>	Reference to operation handle.
------------------	--------------------------------

Returns

nothing after the operation free.

10.40.1.16 TEE_FreeTransientObject() `void TEE_FreeTransientObject (`
`TEE_ObjectHandle object)`

Crypto, Asymmetric key Verification Functions.

[TEE_FreeTransientObject\(\)](#) - Deallocates a transient object previously allocated with [TEE_AllocateTransientObject](#) .

this function describes the object handle is no longer valid and all resources associated with the transient object shall have been reclaimed after the [TEE_AllocateTransientObject\(\)](#) call.

Parameters

<i>object</i>	Handle on the object to free.
---------------	-------------------------------

10.40.1.17 TEE_GenerateKey() `TEE_Result TEE_GenerateKey (`
`TEE_ObjectHandle object,`
`uint32_t keySize,`
`const TEE_Attribute * params,`
`uint32_t paramCount)`

Crypto, Asymmetric key Verification Functions.

[TEE_GenerateKey \(\)](#) - Generates a random key or a key-pair and populates a transient key object with the generated key material.

The size of the desired key is passed in the *keySize* parameter and shall be less than or equal to the maximum key size specified when the transient object was created.

Parameters

<i>object</i>	Handle on an uninitialized transient key to populate with the generated key.
<i>keySize</i>	Requested key size shall be less than or equal to the maximum key size specified when the object container was created
<i>params</i>	Parameters for the key generation.
<i>paramCount</i>	The values of all parameters are copied into the object so that the params array and all the memory buffers it points to may be freed after this routine returns without affecting the object.

Returns

0 on succes

[TEE_ERROR_BAD_PARAMETERS](#) If an incorrect or inconsistent attribute is detected. The checks that are performed depend on the implementation.

10.40.1.18 TEE_InitRefAttribute() `void TEE_InitRefAttribute (`
`TEE_Attribute * attr,`

```
uint32_t attributeID,
const void * buffer,
uint32_t length )
```

Crypto, Asymmetric key Verification Functions.

[TEE_InitRefAttribute\(\)](#) - The helper function can be used to populate a single attribute either with a reference to a buffer or with integer values.

In `TEE_InitRefAttribute()` only the buffer pointer is copied, not the content of the buffer. This means that the attribute structure maintains a pointer back to the supplied buffer. It is the responsibility of the TA author to ensure that the contents of the buffer maintain their value until the attributes array is no longer in use.

Parameters

<i>attr</i>	attribute structure to initialize.
<i>attributeID</i>	Identifier of the attribute to populate.
<i>buffer</i>	input buffer that holds the content of the attribute.
<i>length</i>	buffer length.

10.40.1.19 TEE_InitValueAttribute() `void TEE_InitValueAttribute (`
`TEE_Attribute * attr,`
`uint32_t attributeID,`
`uint32_t a,`
`uint32_t b)`

Crypto, Asymmetric key Verification Functions.

[TEE_InitValueAttribute\(\)](#) - The helper function can be used to populate a single attribute either with a reference to a buffer or with integer values.

Parameters

<i>attr</i>	attribute structure to initialize.
<i>attributeID</i>	Identifier of the attribute to populate.
<i>a</i>	unsigned integer value to assign to the a member of the attribute structure.
<i>b</i>	unsigned integer value to assign to the b member of the attribute structure

10.40.1.20 TEE_SetOperationKey() `TEE_Result TEE_SetOperationKey (`
`TEE_OperationHandle operation,`
`TEE_ObjectHandle key)`

Crypto, Authenticated Encryption with Symmetric key Verification Functions.

[TEE_SetOperationKey\(\)](#) - Programs the key of an operation; that is, it associates an operation with a key.

The key material is copied from the key object handle into the operation. After the key has been set, there is no longer any link between the operation and the key object. The object handle can be closed or reset and this will not affect the operation. This copied material exists until the operation is freed using `TEE_FreeOperation` or another key is set into the operation.

Parameters

<i>operation</i>	Operation handle.
<i>key</i>	A handle on a key object.

Returns

0 on success return

`TEE_ERROR_CORRUPT_OBJECT` If the object is corrupt. The object handle is closed.

`TEE_ERROR_STORAGE_NOT_AVAILABLE` If the persistent object is stored in a storage area which is currently inaccessible.

10.40.1.21 `wolfSSL_Free()` `void wolfSSLFree (`
`void * p)`

[`wolfSSL_Free\(\)`](#) - Deallocates the memory which allocated previously.

Parameters

<i>p</i>	This is the pointer to a memory block.
----------	--

10.40.1.22 `wolfSSL_Malloc()` `void * wolfSSLMalloc (`
`size_t n)`

[`wolfSSL_Malloc\(\)`](#) - Allocates the requested memory and returns a pointer to it.

Parameters

<i>n</i>	size of the memory block.
----------	---------------------------

10.41 ta-ref/docs/building.md File Reference

10.42 ta-ref/docs/gp_api.md File Reference

10.43 ta-ref/docs/how_to_program_on_ta-ref.md File Reference

10.44 ta-ref/docs/overview_of_ta-ref.md File Reference

10.45 ta-ref/docs/preparation.md File Reference

10.46 ta-ref/docs/running_on_dev_boards.md File Reference

Index

- `__TEE_ObjectHandle`, 36
 - `desc`, 37
 - `flags`, 37
 - `persist_ctx`, 37
 - `persist_iv`, 37
 - `private_key`, 37
 - `public_key`, 37
 - `type`, 37
- `__TEE_OperationHandle`, 38
 - `aectx`, 38
 - `aegcm_state`, 38
 - `aegcmctx`, 38
 - `aeiv`, 38
 - `aekey`, 38
 - `alg`, 38
 - `ctx`, 38
 - `flags`, 39
 - `mode`, 39
 - `prikey`, 39
 - `pubkey`, 39
- `__aligned`
 - `tee_api_types.h`, 105
- `__attribute__`
 - `tee-internal-api-machine.c`, 133
 - `tee-internal-api.c`, 147
 - `tee-ta-internal.h`, 72
- `_sanctum_dev_public_key`
 - `test_dev_key.h`, 126
- `_sanctum_dev_public_key_len`
 - `test_dev_key.h`, 126
- `_sanctum_dev_secret_key`
 - `test_dev_key.h`, 126
- `_sanctum_dev_secret_key_len`
 - `test_dev_key.h`, 126
- a
 - `TEE_Attribute`, 44
 - `TEE_Param`, 51
 - `TEEC_Value`, 63
- `addrinfo`, 39
 - `ai_addr`, 40
 - `ai_addrlen`, 40
 - `ai_canonname`, 40
 - `ai_family`, 40
 - `ai_flags`, 40
 - `ai_next`, 40
 - `ai_protocol`, 40
 - `ai_socktype`, 40
- `aectx`
 - `__TEE_OperationHandle`, 38
- `aegcm_state`
 - `__TEE_OperationHandle`, 38
- `aegcmctx`
 - `__TEE_OperationHandle`, 38
- `aeiv`
 - `__TEE_OperationHandle`, 38
- `aekey`
 - `__TEE_OperationHandle`, 38
- `ai_addr`
 - `addrinfo`, 40
- `ai_addrlen`
 - `addrinfo`, 40
- `ai_canonname`
 - `addrinfo`, 40
- `ai_family`
 - `addrinfo`, 40
- `ai_flags`
 - `addrinfo`, 40
- `ai_next`
 - `addrinfo`, 40
- `ai_protocol`
 - `addrinfo`, 40
- `ai_socktype`
 - `addrinfo`, 40
- `alg`
 - `__TEE_OperationHandle`, 38
- `algorithm`
 - `TEE_OperationInfo`, 47
 - `TEE_OperationInfoMultiple`, 49
- `allocated_size`
 - `TEEC_SharedMemory`, 60
- `attributeID`
 - `TEE_Attribute`, 44
- b
 - `TEE_Attribute`, 44
 - `TEE_Param`, 51
 - `TEEC_Value`, 63
- `buffer`
 - `TEE_Attribute`, 44
 - `TEE_Param`, 51
 - `TEE_SEAID`, 52
 - `TEEC_SharedMemory`, 60
 - `TEEC_TempMemoryReference`, 61
- `buffer_allocated`
 - `TEEC_SharedMemory`, 60
- `bufferLen`
 - `TEE_SEAID`, 52
- `clockSeqAndNode`
 - `TEE_UUID`, 53
 - `TEEC_UUID`, 62
- `content`
 - `TEE_Attribute`, 44
- `ctx`
 - `__TEE_OperationHandle`, 38
 - `TEEC_Session`, 59
- `data`
 - `enclave_report`, 41
- `data_len`

- enclave_report, 41
- dataPosition
 - TEE_ObjectInfo, 46
- dataSize
 - TEE_ObjectInfo, 46
- desc
 - __TEE_ObjectHandle, 37
- dev_public_key
 - report, 42
- dhex_dump
 - trace.h, 128
- digestLength
 - TEE_OperationInfo, 48
 - TEE_OperationInfoMultiple, 50
- enclave
 - report, 42
- enclave_report, 40
 - data, 41
 - data_len, 41
 - hash, 41
 - signature, 41
- events
 - pollfd, 41
- fd
 - pollfd, 42
 - TEEC_Context, 54
- flags
 - __TEE_ObjectHandle, 37
 - __TEE_OperationHandle, 39
 - TEEC_SharedMemory, 60
- flags2flags
 - tee-internal-api.c, 136, 147
- get_wc_rng
 - tee-internal-api.c, 136, 147
- GetRelTimeEnd
 - tee-internal-api.c, 137, 148
 - tee-ta-internal.h, 73
- GetRelTimeStart
 - tee-internal-api.c, 137, 148
 - tee-ta-internal.h, 73
- handleFlags
 - TEE_ObjectInfo, 46
- handleState
 - TEE_OperationInfo, 48
 - TEE_OperationInfoMultiple, 50
- hash
 - enclave_report, 41
 - sm_report, 43
- id
 - TEEC_SharedMemory, 60
- keyInformation
 - TEE_OperationInfoMultiple, 50
- keySize
 - TEE_ObjectInfo, 46
- TEE_OperationInfo, 48
 - TEE_OperationInfoKey, 49
- length
 - TEE_Attribute, 44
- login
 - TEE.Identity, 45
- maxKeySize
 - TEE_ObjectInfo, 47
 - TEE_OperationInfo, 48
 - TEE_OperationInfoMultiple, 50
- maxObjectSize
 - TEE_ObjectInfo, 47
- memref
 - TEE_Param, 51
 - TEEC_Parameter, 57
- millis
 - TEE_Time, 53
- mode
 - __TEE_OperationHandle, 39
 - TEE_OperationInfo, 48
 - TEE_OperationInfoMultiple, 50
- nfds_t
 - tee_api_types.h, 105
- numberOfKeys
 - TEE_OperationInfoMultiple, 50
- objectSize
 - TEE_ObjectInfo, 47
- objectType
 - TEE_ObjectInfo, 47
- objectUsage
 - TEE_ObjectInfo, 47
- offset
 - TEEC_RegisteredMemoryReference, 58
- OpenPersistentObject
 - tee-internal-api.c, 138, 148
- operationClass
 - TEE_OperationInfo, 48
 - TEE_OperationInfoMultiple, 50
- operationState
 - TEE_OperationInfoMultiple, 50
- params
 - TEEC_Operation, 55
- paramTypes
 - TEEC_Operation, 55
- parent
 - TEEC_RegisteredMemoryReference, 58
- persist_ctx
 - __TEE_ObjectHandle, 37
- persist_iv
 - __TEE_ObjectHandle, 37
- pollfd, 41
 - events, 41
 - fd, 42
 - revents, 42

- prikey
 - __TEE_OperationHandle, 39
- private_key
 - __TEE_ObjectHandle, 37
- pubkey
 - __TEE_OperationHandle, 39
- public_key
 - __TEE_ObjectHandle, 37
 - sm_report, 43
- ref
 - TEE_Attribute, 44
- reg_mem
 - TEEC_Context, 54
- registered_fd
 - TEEC_SharedMemory, 61
- report, 42
 - dev_public_key, 42
 - enclave, 42
 - sm, 43
- requiredKeyUsage
 - TEE_OperationInfo, 48
 - TEE_OperationInfoKey, 49
- revents
 - pollfd, 42
- rngstr
 - tee-internal-api.c, 145, 154
- seconds
 - TEE_Time, 53
- selectResponseEnable
 - TEE_SEReaderProperties, 52
- sePresent
 - TEE_SEReaderProperties, 52
- session
 - TEEC_Operation, 56
- session_id
 - TEEC_Session, 59
- set_object_key
 - tee-internal-api.c, 138, 149
- shadow_buffer
 - TEEC_SharedMemory, 61
- signature
 - enclave_report, 41
 - sm_report, 43
- size
 - TEE_Param, 51
 - TEEC_RegisteredMemoryReference, 58
 - TEEC_SharedMemory, 61
 - TEEC_TempMemoryReference, 62
- sm
 - report, 43
- sm_report, 43
 - hash, 43
 - public_key, 43
 - signature, 43
- socklen_t
 - tee_api_types.h, 105
- started
 - TEEC_Operation, 56
- ta-ref/api/include/compiler.h, 64
- ta-ref/api/include/report.h, 67
- ta-ref/api/include/tee-common.h, 68, 69
- ta-ref/api/include/tee-ta-internal.h, 69, 94
- ta-ref/api/include/tee_api_defines.h, 96
- ta-ref/api/include/tee_api_defines_extensions.h, 102
- ta-ref/api/include/tee_api_types.h, 104, 107
- ta-ref/api/include/tee_client_api.h, 110, 116
- ta-ref/api/include/tee_internal_api.h, 119
- ta-ref/api/include/tee_internal_api_extensions.h, 119, 121
- ta-ref/api/include/tee_ta_api.h, 122, 123
- ta-ref/api/include/test_dev_key.h, 126, 127
- ta-ref/api/include/trace.h, 127, 129
- ta-ref/api/include/trace_levels.h, 132
- ta-ref/api/keystone/tee-internal-api-machine.c, 133
- ta-ref/api/keystone/tee-internal-api.c, 135
- ta-ref/api/keystone/tee_api_tee_types.h, 154, 155
- ta-ref/api/keystone/teec_stub.c, 160
- ta-ref/api/keystone/trace.c, 163
- ta-ref/api/optee/tee_api_tee_types.h, 157
- ta-ref/api/sgx/tee-internal-api.c, 145
- ta-ref/api/sgx/tee_api_tee_types.h, 157, 158
- ta-ref/api/tee-internal-api-cryptlib.c, 165
- ta-ref/docs/building.md, 179
- ta-ref/docs/gp_api.md, 179
- ta-ref/docs/how_to_program_on_ta-ref.md, 179
- ta-ref/docs/overview_of_ta-ref.md, 179
- ta-ref/docs/preparation.md, 179
- ta-ref/docs/running_on_dev_boards.md, 179
- TA_CloseSessionEntryPoint
 - tee_ta_api.h, 122
- TA_CreateEntryPoint
 - tee_ta_api.h, 122
- TA_DestroyEntryPoint
 - tee_ta_api.h, 123
- TA_InvokeCommandEntryPoint
 - tee_ta_api.h, 123
- TA_OpenSessionEntryPoint
 - tee_ta_api.h, 123
- tee-internal-api-cryptlib.c
 - TEE_AEDecryptFinal, 167
 - TEE_AEEncryptFinal, 168
 - TEE_AEInit, 168
 - TEE_AEUpdate, 169
 - TEE_AEUpdateAAD, 170
 - TEE_AllocateOperation, 170
 - TEE_AllocateTransientObject, 171
 - TEE_AsymmetricSignDigest, 171
 - TEE_AsymmetricVerifyDigest, 172
 - TEE_CipherDoFinal, 172
 - TEE_CipherInit, 173
 - TEE_CipherUpdate, 173
 - TEE_DigestDoFinal, 174
 - TEE_DigestUpdate, 175
 - TEE_FreeOperation, 175
 - TEE_FreeTransientObject, 175
 - TEE_GenerateKey, 176

- TEE_InitRefAttribute, 176
- TEE_InitValueAttribute, 177
- TEE_SetOperationKey, 177
- wolfSSL_Free, 178
- wolfSSL_Malloc, 178
- tee-internal-api-machine.c
 - __attribute__, 133
- tee-internal-api.c
 - __attribute__, 147
 - flags2flags, 136, 147
 - get_wc_rng, 136, 147
 - GetRelTimeEnd, 137, 148
 - GetRelTimeStart, 137, 148
 - OpenPersistentObject, 138, 148
 - rngstr, 145, 154
 - set_object_key, 138, 149
 - TEE_CloseObject, 139, 149
 - TEE_CreatePersistentObject, 139, 150
 - TEE_Free, 140
 - TEE_GenerateRandom, 140, 151
 - TEE_GetObjectInfo1, 141, 151
 - TEE_GetREETime, 141, 151
 - TEE_GetSystemTime, 142, 152
 - TEE_Malloc, 142
 - TEE_OpenPersistentObject, 142, 152
 - TEE_ReadObjectData, 143, 153
 - TEE_Realloc, 143
 - TEE_WriteObjectData, 144, 153
 - wc_ocall_genseed, 144
 - wc_rng_init, 145, 154
- tee-ta-internal.h
 - __attribute__, 72
 - GetRelTimeEnd, 73
 - GetRelTimeStart, 73
 - TEE_AEDecryptFinal, 74
 - TEE_AEEncryptFinal, 75
 - TEE_AEInit, 76
 - TEE_AEUpdate, 76
 - TEE_AEUpdateAAD, 78
 - TEE_AllocateOperation, 78
 - TEE_AllocateTransientObject, 79
 - TEE_AsymmetricSignDigest, 80
 - TEE_AsymmetricVerifyDigest, 80
 - TEE_CipherInit, 81
 - TEE_CipherUpdate, 82
 - TEE_CloseObject, 82
 - TEE_CreatePersistentObject, 83
 - TEE_DigestDoFinal, 84
 - TEE_DigestUpdate, 85
 - TEE_FreeOperation, 85
 - TEE_FreeTransientObject, 86
 - TEE_GenerateKey, 86
 - TEE_GenerateRandom, 87
 - TEE_GetObjectInfo1, 88
 - TEE_GetREETime, 88
 - TEE_GetSystemTime, 89
 - TEE_InitRefAttribute, 89
 - TEE_InitValueAttribute, 90
 - TEE_OpenPersistentObject, 90
 - TEE_ReadObjectData, 91
 - TEE_SetOperationKey, 92
 - TEE_WriteObjectData, 93
 - TEE_AEDecryptFinal
 - tee-internal-api-cryptlib.c, 167
 - tee-ta-internal.h, 74
 - TEE_AEEncryptFinal
 - tee-internal-api-cryptlib.c, 168
 - tee-ta-internal.h, 75
 - TEE_AEInit
 - tee-internal-api-cryptlib.c, 168
 - tee-ta-internal.h, 76
 - TEE_AEUpdate
 - tee-internal-api-cryptlib.c, 169
 - tee-ta-internal.h, 76
 - TEE_AEUpdateAAD
 - tee-internal-api-cryptlib.c, 170
 - tee-ta-internal.h, 78
 - TEE_AllocateOperation
 - tee-internal-api-cryptlib.c, 170
 - tee-ta-internal.h, 78
 - TEE_AllocateTransientObject
 - tee-internal-api-cryptlib.c, 171
 - tee-ta-internal.h, 79
- tee.api_types.h
 - __aligned, 105
 - nfds_t, 105
 - socklen_t, 105
 - TEE_BigInt, 105
 - TEE_BigIntFMM, 105
 - TEE_DATA_SEEK_CUR, 107
 - TEE_DATA_SEEK_END, 107
 - TEE_DATA_SEEK_SET, 107
 - TEE_ErrorOrigin, 106
 - TEE_MODE_DECRYPT, 107
 - TEE_MODE_DERIVE, 107
 - TEE_MODE_DIGEST, 107
 - TEE_MODE_ENCRYPT, 107
 - TEE_MODE_MAC, 107
 - TEE_MODE_SIGN, 107
 - TEE_MODE_VERIFY, 107
 - TEE_ObjectEnumHandle, 106
 - TEE_ObjectHandle, 106
 - TEE_ObjectType, 106
 - TEE_OperationHandle, 106
 - TEE_OperationMode, 107
 - TEE_PropSetHandle, 106
 - TEE_Result, 106
 - TEE_SEChannelHandle, 106
 - TEE_SEReaderHandle, 106
 - TEE_SEServiceHandle, 106
 - TEE_SESessionHandle, 106
 - TEE_Session, 107
 - TEE_TASessionHandle, 107
 - TEE_Whence, 107
 - TEE_AsymmetricSignDigest
 - tee-internal-api-cryptlib.c, 171

- tee-ta-internal.h, 80
- TEE.AsymmetricVerifyDigest
 - tee-internal-api-cryptlib.c, 172
 - tee-ta-internal.h, 80
- TEE.Attribute, 43
 - a, 44
 - attributeID, 44
 - b, 44
 - buffer, 44
 - content, 44
 - length, 44
 - ref, 44
 - value, 44
- TEE.BigInt
 - tee_api_types.h, 105
- TEE.BigIntFMM
 - tee_api_types.h, 105
- TEE.CacheClean
 - tee_internal_api_extensions.h, 120
- TEE.CacheFlush
 - tee_internal_api_extensions.h, 120
- TEE.CacheInvalidate
 - tee_internal_api_extensions.h, 120
- TEE.CipherDoFinal
 - tee-internal-api-cryptlib.c, 172
- TEE.CipherInit
 - tee-internal-api-cryptlib.c, 173
 - tee-ta-internal.h, 81
- TEE.CipherUpdate
 - tee-internal-api-cryptlib.c, 173
 - tee-ta-internal.h, 82
- tee_client_api.h
 - TEEC.AllocateSharedMemory, 112
 - TEEC.CloseSession, 112
 - TEEC.FinalizeContext, 113
 - TEEC.InitializeContext, 113
 - TEEC.InvokeCommand, 114
 - TEEC.OpenSession, 114
 - TEEC.RegisterSharedMemory, 115
 - TEEC.ReleaseSharedMemory, 115
 - TEEC.RequestCancellation, 115
 - TEEC.Result, 112
- TEE.CloseObject
 - tee-internal-api.c, 139, 149
 - tee-ta-internal.h, 82
- TEE.CreatePersistentObject
 - tee-internal-api.c, 139, 150
 - tee-ta-internal.h, 83
- TEE.DATA_SEEK_CUR
 - tee_api_types.h, 107
- TEE.DATA_SEEK_END
 - tee_api_types.h, 107
- TEE.DATA_SEEK_SET
 - tee_api_types.h, 107
- TEE.DigestDoFinal
 - tee-internal-api-cryptlib.c, 174
 - tee-ta-internal.h, 84
- TEE.DigestUpdate
 - tee-internal-api-cryptlib.c, 175
 - tee-ta-internal.h, 85
- TEE.ErrorOrigin
 - tee_api_types.h, 106
- TEE.Free
 - tee-internal-api.c, 140
- TEE.FreeOperation
 - tee-internal-api-cryptlib.c, 175
 - tee-ta-internal.h, 85
- TEE.FreeTransientObject
 - tee-internal-api-cryptlib.c, 175
 - tee-ta-internal.h, 86
- TEE.GenerateKey
 - tee-internal-api-cryptlib.c, 176
 - tee-ta-internal.h, 86
- TEE.GenerateRandom
 - tee-internal-api.c, 140, 151
 - tee-ta-internal.h, 87
- TEE.GetObjectInfo1
 - tee-internal-api.c, 141, 151
 - tee-ta-internal.h, 88
- TEE.GetREETime
 - tee-internal-api.c, 141, 151
 - tee-ta-internal.h, 88
- TEE.GetSystemTime
 - tee-internal-api.c, 142, 152
 - tee-ta-internal.h, 89
- TEE.Identity, 45
 - login, 45
 - uuid, 45
- TEE.InitRefAttribute
 - tee-internal-api-cryptlib.c, 176
 - tee-ta-internal.h, 89
- TEE.InitValueAttribute
 - tee-internal-api-cryptlib.c, 177
 - tee-ta-internal.h, 90
- tee_internal_api_extensions.h
 - TEE.CacheClean, 120
 - TEE.CacheFlush, 120
 - TEE.CacheInvalidate, 120
 - tee_map_zi, 120
 - tee_unmap, 120
 - tee_user_mem_check_heap, 120
 - tee_user_mem_mark_heap, 120
 - tee_uuid_from_str, 121
- TEE.Malloc
 - tee-internal-api.c, 142
- tee_map_zi
 - tee_internal_api_extensions.h, 120
- TEE.MODE_DECRYPT
 - tee_api_types.h, 107
- TEE.MODE_DERIVE
 - tee_api_types.h, 107
- TEE.MODE_DIGEST
 - tee_api_types.h, 107
- TEE.MODE_ENCRYPT
 - tee_api_types.h, 107
- TEE.MODE_MAC

- tee_api_types.h, 107
- TEE.MODE_SIGN
 - tee_api_types.h, 107
- TEE.MODE_VERIFY
 - tee_api_types.h, 107
- TEE.ObjectEnumHandle
 - tee_api_types.h, 106
- TEE.ObjectHandle
 - tee_api_types.h, 106
- TEE.ObjectInfo, 46
 - dataPosition, 46
 - dataSize, 46
 - handleFlags, 46
 - keySize, 46
 - maxKeySize, 47
 - maxObjectSize, 47
 - objectSize, 47
 - objectType, 47
 - objectUsage, 47
- TEE.ObjectType
 - tee_api_types.h, 106
- TEE.OpenPersistentObject
 - tee-internal-api.c, 142, 152
 - tee-ta-internal.h, 90
- TEE.OperationHandle
 - tee_api_types.h, 106
- TEE.OperationInfo, 47
 - algorithm, 47
 - digestLength, 48
 - handleState, 48
 - keySize, 48
 - maxKeySize, 48
 - mode, 48
 - operationClass, 48
 - requiredKeyUsage, 48
- TEE.OperationInfoKey, 48
 - keySize, 49
 - requiredKeyUsage, 49
- TEE.OperationInfoMultiple, 49
 - algorithm, 49
 - digestLength, 50
 - handleState, 50
 - keyInformation, 50
 - maxKeySize, 50
 - mode, 50
 - numberOfKeys, 50
 - operationClass, 50
 - operationState, 50
- TEE.OperationMode
 - tee_api_types.h, 107
- TEE.Param, 50
 - a, 51
 - b, 51
 - buffer, 51
 - memref, 51
 - size, 51
 - value, 51
- TEE.PropSetHandle
 - tee_api_types.h, 106
- TEE.ReadObjectData
 - tee-internal-api.c, 143, 153
 - tee-ta-internal.h, 91
- TEE.Realloc
 - tee-internal-api.c, 143
- TEE.Result
 - tee_api_types.h, 106
- TEE.SEAID, 51
 - buffer, 52
 - bufferLen, 52
- TEE.SEChannelHandle
 - tee_api_types.h, 106
- TEE.SEReaderHandle
 - tee_api_types.h, 106
- TEE.SEReaderProperties, 52
 - selectResponseEnable, 52
 - sePresent, 52
 - teeOnly, 52
- TEE.SEServiceHandle
 - tee_api_types.h, 106
- TEE.SESessionHandle
 - tee_api_types.h, 106
- TEE.Session
 - tee_api_types.h, 107
- TEE.SetOperationKey
 - tee-internal-api-cryptlib.c, 177
 - tee-ta-internal.h, 92
- tee.ta.api.h
 - TA.CloseSessionEntryPoint, 122
 - TA.CreateEntryPoint, 122
 - TA.DestroyEntryPoint, 123
 - TA.InvokeCommandEntryPoint, 123
 - TA.OpenSessionEntryPoint, 123
- TEE.TASessionHandle
 - tee_api_types.h, 107
- TEE.Time, 53
 - millis, 53
 - seconds, 53
- tee_unmap
 - tee_internal_api_extensions.h, 120
- tee_user_mem_check_heap
 - tee_internal_api_extensions.h, 120
- tee_user_mem_mark_heap
 - tee_internal_api_extensions.h, 120
- TEE.UUID, 53
 - clockSeqAndNode, 53
 - timeHiAndVersion, 53
 - timeLow, 54
 - timeMid, 54
- tee_uuid_from_str
 - tee_internal_api_extensions.h, 121
- TEE.Whence
 - tee_api_types.h, 107
- TEE.WriteObjectData
 - tee-internal-api.c, 144, 153
 - tee-ta-internal.h, 93
- TEEC.AllocateSharedMemory

- tee_client_api.h, 112
- teec_stub.c, 160
- TEEC_CloseSession
 - tee_client_api.h, 112
 - teec_stub.c, 161
- TEEC_Context, 54
 - fd, 54
 - reg_mem, 54
- TEEC_FinalizeContext
 - tee_client_api.h, 113
 - teec_stub.c, 161
- TEEC_InitializeContext
 - tee_client_api.h, 113
 - teec_stub.c, 161
- TEEC_InvokeCommand
 - tee_client_api.h, 114
- TEEC_OpenSession
 - tee_client_api.h, 114
 - teec_stub.c, 162
- TEEC_Operation, 55
 - params, 55
 - paramTypes, 55
 - session, 56
 - started, 56
- TEEC_Parameter, 56
 - memref, 57
 - tmpref, 57
 - value, 57
- TEEC_RegisteredMemoryReference, 57
 - offset, 58
 - parent, 58
 - size, 58
- TEEC_RegisterSharedMemory
 - tee_client_api.h, 115
 - teec_stub.c, 162
- TEEC_ReleaseSharedMemory
 - tee_client_api.h, 115
 - teec_stub.c, 163
- TEEC_RequestCancellation
 - tee_client_api.h, 115
 - teec_stub.c, 163
- TEEC_Result
 - tee_client_api.h, 112
- TEEC_Session, 59
 - ctx, 59
 - session_id, 59
- TEEC_SharedMemory, 59
 - allocated_size, 60
 - buffer, 60
 - buffer_allocated, 60
 - flags, 60
 - id, 60
 - registered_fd, 61
 - shadow_buffer, 61
 - size, 61
- teec_stub.c
 - TEEC_AllocateSharedMemory, 160
 - TEEC_CloseSession, 161
 - TEEC_FinalizeContext, 161
 - TEEC_InitializeContext, 161
 - TEEC_OpenSession, 162
 - TEEC_RegisterSharedMemory, 162
 - TEEC_ReleaseSharedMemory, 163
 - TEEC_RequestCancellation, 163
 - TEEC_TempMemoryReference, 61
 - buffer, 61
 - size, 62
 - TEEC_UUID, 62
 - clockSeqAndNode, 62
 - timeHiAndVersion, 62
 - timeLow, 62
 - timeMid, 62
 - TEEC_Value, 63
 - a, 63
 - b, 63
 - teeOnly
 - TEE_SEReaderProperties, 52
- test_dev_key.h
 - _sanctum_dev_public_key, 126
 - _sanctum_dev_public_key_len, 126
 - _sanctum_dev_secret_key, 126
 - _sanctum_dev_secret_key_len, 126
- timeHiAndVersion
 - TEE_UUID, 53
 - TEEC_UUID, 62
- timeLow
 - TEE_UUID, 54
 - TEEC_UUID, 62
- timeMid
 - TEE_UUID, 54
 - TEEC_UUID, 62
- tmpref
 - TEEC_Parameter, 57
- trace.c
 - trace_printf, 164
 - trace_vprintf, 165
- trace.h
 - dhex_dump, 128
 - trace_ext_get_thread_id, 128
 - trace_ext_prefix, 129
 - trace_ext_puts, 128
 - trace_get_level, 128
 - trace_level, 129
 - trace_printf, 128
 - trace_set_level, 128
- trace_ext_get_thread_id
 - trace.h, 128
- trace_ext_prefix
 - trace.h, 129
- trace_ext_puts
 - trace.h, 128
- trace_get_level
 - trace.h, 128
- trace_level
 - trace.h, 129
- trace_printf

- trace.c, [164](#)
 - trace.h, [128](#)
- trace_set_level
 - trace.h, [128](#)
- trace_vprintf
 - trace.c, [165](#)
- type
 - _TEE_ObjectHandle, [37](#)
- uuid
 - TEE_Identity, [45](#)
- value
 - TEE_Attribute, [44](#)
 - TEE_Param, [51](#)
 - TEEC_Parameter, [57](#)
- wc_ocall_genseed
 - tee-internal-api.c, [144](#)
- wc_rng_init
 - tee-internal-api.c, [145](#), [154](#)
- wolfSSL_Free
 - tee-internal-api-cryptlib.c, [178](#)
- wolfSSL_Malloc
 - tee-internal-api-cryptlib.c, [178](#)