Encryption Test Suite User Guide

Version 1.00 BETA

For use with Encryption Test Suite versions 1.06 and above

Date: 23-Feb-2016 14:03

Table of Contents

Version 1.00 BETA

For use with Encryption Test Suite versions 1.06 and above

Tip: Use the Contents list on the left to navigate within this manual.

Encryption Documents Home

1 System Overview

1.1 Introduction

This guide is for those who want to use HCC Embedded's Encryption Test Suite to exercise the Embedded Encryption Manager (EEM) and its range of encryption/hash algorithms.

The Encryption Test Suite has two components that you can use as follows:

- To test the EEM.
- To test the encryption, hash and IPsec algorithms used with the EEM. Each algorithm is registered with the EEM and obtains a handle at this point. This algorithm handle is needed to register an algorithm test.

The HCC encryption/hash algorithms that you can test are as follows:

- Advanced Encryption Standard (AES), AES RAW, and AES CTR.
- Data Encryption Standard (DES) and DES RAW.
- Digital Signature Standard (DSS)
- Ephemeral Diffie-Hellman (EDH)
- Message Digest Algorithm 5 (MD5)
- RSA Signature Algorithm (RSA)
- SHA-1, SHA-256, SHA-384 and SHA-512 Secure Hash Algorithms.
- Triple Data Encryption Standard (3DES) and 3DES RAW.

In addition you can use two tests for IPsec:

- IPsec NULL encryption driver test.
- IPsec NULL integrity check driver test.

This manual describes the Application Programming Interface (API) functions available for running tests.

1.2 Feature Check

The main features of the Encryption Test Suite are the following:

- It conforms to the HCC Advanced Embedded Framework.
- It can be used with or without an RTOS.
- It tests the Embedded Encryption Manager (EEM).
- It tests all the HCC encryption/hash algorithms. These are listed above.

1.3 Packages and Documents

Packages

The table below lists the packages that you need in order to use this module.

Package	Description
hcc_base_docs	This contains the two guides that will help you get started.
enc_base	The EEM base package.
enc_test	The Encryption Test Suite package.

Documents

For an overview of HCC verifiable embedded network encryption, refer to the Product Information section of the main HCC website.

Readers should note the points in the HCC Documentation Guidelines on the HCC documentation website.

HCC Firmware Quick Start Guide

This document describes how to install packages provided by HCC in the target development environment. Also follow the *Quick Start Guide* when HCC provides package updates.

HCC Source Tree Guide

This document describes the HCC source tree. It gives an overview of the system to make clear the logic behind its organization.

Encryption Test Suite User Guide

This is this document.

HCC Embedded Encryption Manager User Guide

This document describes the EEM.

HCC Algorithm User Guides

There is a separate document for each encryption/hash algorithm. For example, the *Triple Data Encryption Standard User Guide* describes the TDES module.

1.4 Change History

This section includes recent changes to this product. For a list of all the changes, refer to the file **src/history /enc/enc_test.txt** in the distribution package.

Version	Changes
1.06	Added tests AES CTR and SEA RAW. Added tests for SHA256 and SHA512. Added tests for TDES RAW and DES RAW. Moved tests of each driver to a separate file (test_aes, test_sha, ...). Added an output buffer length check (functions are checked if they fail when the output buffer length is smaller by 1 than the expected length.)
1.05	Corrected boundary test for decryption. Added new tests to AES component. Changed the driver initialization functions to macros that are defined in **config_enc_test.h**. This makes it easy to change the driver used for the current test.
1.04	Code simplification: • *psp_syserr* is now always an infinite loop. • Removed test checking of big number function check alignment and big number size. • Added a boundary test for drivers. • Corrected the false use test for ENC_DRVTEST_TEST_ENDECRYPT test.

2 Source File List

This section describes all the source code files included in the system. These files follow the HCC Embedded standard source tree system, described in the *HCC Source Tree Guide*. All references to file pathnames refer to locations within this standard source tree, not within the package you initially receive.

> **Note:** Do not modify any files except the configuration file.

2.1 API Header File

The file **src/api/api_enc_test.h** should be included by any application using the system. This is the only file that should be included by an application using this module. For details of the functions, see Application Programming Interface.

2.2 Configuration File

The file **src/config/config_enc_test.h** contains all the configurable parameters of the system. Configure these as required. This is the only file in the module that you should modify. For details of these options, see Configuration Options.

2.3 System Files

These files are in the directory **src/enc/test**. **These files should only be modified by HCC**.

File	Description
enc_driver_test.c	Test code for encryption/hash algorithms.
enc_test.c	EEM test code.
test_data.c	Functions used to obtain the configuration structure for each algorithm type.
test_aes.c	AES test source.
test_dss.c	DSS test source.
test_edh.c	EDH test source.
test_ipsec_null.c	IPsec NULL encryption driver source.
test_md5.c	MD5 test source.
test_rsa.c	RSA test source.
test_sha.c	SHA test source.
test_tdes.c	3DES test source.
Vectors.c and **Vectors.h**	Files for big number arithmetic.

2.4 Version Files

The file **src/version/ver_enc_test.h** contains the version number of the module. The version number is checked by all modules that use the module to ensure system consistency over upgrades.

3 Configuration Options

Set the system configuration options in the file **src/config/config_enc_test.h**. This section lists the available configuration options and their default values.

ENC_TEST_TASK_STACK_SIZE

The size of the test stack. The default value is 256.

ENC_DRIVER_TEST_TASK_STACK_SIZE

The size of the encryption/hash algorithms test stack. The default value is 256.

ENC_DRIVER_TEST_TEST_NUMBER

The maximum number of algorithms that can be tested. The default value is 16.

ENC_DRIVER_TEST_OUTBUF_SIZE

The size of the output buffer. The default value is 256.

ENC_DRIVER_TEST_AES128_EN
Keep this at the default of 1 to enable the AES 128 bit test.

ENC_DRIVER_TEST_AES256_EN
Keep this at the default of 1 to enable the AES 256 bit test.

> **Note:** The following options define the **init()** functions for the algorithm modules that will be tested.

ENC_DRIVER_TEST_AES_INITFN

The AES encryption algorithm module **init()** function. The default value is *&aes_init_fn*.

ENC_DRIVER_TEST_AES_RAW_INITFN

The AES RAW encryption algorithm module **init()** function. The default value is *&aes_raw_init_fn*.

ENC_DRIVER_TEST_AES_CTR_INITFN

The AES CTR encryption algorithm module **init()** function. The default value is *&aes_ctr_init_fn*.

ENC_DRIVER_TEST_TDES_INITFN

The Triple DES encryption algorithm module **init()** function. The default value is &*tdes_init_fn*.

ENC_DRIVER_TEST_TDES_RAW_INITFN

The Triple DES RAW encryption algorithm module **init()** function. The default value is &*tdes_raw_init_fn*.

ENC_DRIVER_TEST_DES_RAW_INITFN

The DES RAW encryption algorithm module **init()** function. The default value is &*des_raw_init_fn*.

ENC_DRIVER_TEST_DSS_INITFN

The DSS encryption algorithm module **init()** function. The default value is *&dss_init_fn*.

ENC_DRIVER_TEST_EDH_INITFN

The EDH encryption algorithm module **init()** function. The default value is *&edh_init_fn*.

ENC_DRIVER_TEST_RSA_INITFN

The RSA encryption algorithm module **init()** function. The default value is &*rsa_init_fn*.

ENC_DRIVER_TEST_SHA512_INITFN

The SHA-512 hash algorithm module **init()** function. The default value is *&sha512_init_fn*.

ENC_DRIVER_TEST_SHA384_INITFN

The SHA-384 hash algorithm module **init()** function. The default value is *&sha384_init_fn*.

ENC_DRIVER_TEST_SHA256_INITFN

The SHA-256 hash algorithm module **init()** function. The default value is *&sha256_init_fn*.

ENC_DRIVER_TEST_SHA1_INITFN

The SHA-1 hash algorithm module **init()** function. The default value is *&sha1_init_fn*.

ENC_DRIVER_TEST_MD5_INITFN

The MD5 hash algorithm module **init()** function. The default value is *&md5_init_fn*.

ENC_DRIVER_TEST_IPSEC_NULL_ENC_INITFN

The IPSec NULL encryption driver **init()** function. The default value is *&ipsec_null_enc_init_fn*.

ENC_DRIVER_TEST_IPSEC_NULL_INT_INITFN

The IPSec NULL integrity check driver **init()** function. The default value is *&ipsec_null_int_init_fn*.

4 Application Programming Interface

This section describes the Application Programming Interface (API). It describes all the functions that are available to the test suite.

4.1 Module Management

These functions control operation of the test suite itself.

enc_test_init

Use this function to initialize the Encryption Test Suite and allocate the required resources.

> **Note:** You must call this function first.

Format

```
t_enc_ret enc_test_init (void)
```

Arguments

Argument
None.

Return Values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
ENC_DRVTEST_INIT_ERROR	Initialization failed.
Else	See Error Codes.

enc_test_reg_callback

Use this function to register the callback function that is called when an EEM test ends.

> **Note:** This function cannot be called from a task.

Format

```
t_enc_ret enc_test_reg_callback ( t_enc_test_cb p_cb )
```

Arguments

Argument	Description	Type
p_cb	The callback function.	t_enc_test_cb

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

4.2 Algorithm Management

These functions control operation of the encryption/hash algorithm tests.

enc_driver_test_init

Use this function to initialize the algorithm tests and allocate the required resources.

> **Note:** This function cannot be called from a task. You must call this function before the other algorithm test functions.

Format

```
t_enc_ret enc_driver_test_init ( void )
```

Arguments

Argument
None.

Return Values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
ENC_DRVTEST_ERROR	Initialization failed.
Else	See Error Codes.

enc_driver_test_register

Use this function to register an algorithm test.

> **Note:** You must call **enc_driver_test_init()** before this.

Format

```
t_enc_ret enc_driver_test_register ( t_enc_drv_test * p_driver_test )
```

Arguments

Argument	Description	Type
p_driver_test	The algorithm handle.	t_enc_drv_test *

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

enc_driver_reg_callback

Use this function to register the callback function which is called when an algorithm test ends.

Format

```
t_enc_ret enc_driver_reg_callback ( t_enc_drvtest_cb p_cb )
```

Arguments

Argument	Description	Type
p_cb	The callback function.	t_enc_drvtest_cb

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

Functions for Getting Test Data

aes_get_drv_test_data

Use this function to obtain the test data for an AES test.

It returns a pointer to the AES algorithm test configuration.

Format

```
t_enc_drv_test * aes_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

des_get_drv_test_data

Use this function to obtain the test data for a DES test.

It returns a pointer to the DES algorithm test configuration.

Format

```
t_enc_drv_test * des_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

des_raw_get_drv_test_data

Use this function to obtain the test data for a DES RAW test.

It returns a pointer to the DES RAW algorithm test configuration.

Format

```
t_enc_drv_test * des_raw_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

tdes_raw_get_drv_test_data

Use this function to obtain the test data for a Triple DES RAW test.

It returns a pointer to the 3DES RAW algorithm test configuration.

Format

```
t_enc_drv_test * tdes_raw_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

dss_get_drv_test_data

Use this function to obtain the test data for a DSS test.

It returns a pointer to the DSS algorithm test configuration.

Format

```
t_enc_drv_test * dss_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

edh_get_drv_test_data

Use this function to obtain the test data for an EDH test.

It returns a pointer to the EDH algorithm test configuration.

Format

```
t_enc_drv_test * edh_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

md5_get_drv_test_data

Use this function to obtain the test data for an MD5 test.

It returns a pointer to the MD5 algorithm test configuration.

Format

```
t_enc_drv_test * md5_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

rsa_get_drv_test_data

Use this function to obtain the test data for an RSA test.

It returns a pointer to the RSA algorithm test configuration.

Format

```
t_enc_drv_test * rsa_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

sha1_get_drv_test_data

Use this function to obtain the test data for a Secure Hash Algorithm 1 (SHA-1) test.

It returns a pointer to the SHA-1 algorithm test configuration.

Format

```
t_enc_drv_test * sha1_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

sha256_get_drv_test_data

Use this function to obtain the test data for a Secure Hash Algorithm 256 (SHA-256) test.

It returns a pointer to the SHA-256 test configuration.

Format

```
t_enc_drv_test * sha256_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

sha384_get_drv_test_data

Use this function to obtain the test data for a Secure Hash Algorithm 384 (SHA-384) test.

It returns a pointer to the SHA-384 test configuration.

Format

```
t_enc_drv_test * sha384_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

sha512_get_drv_test_data

Use this function to obtain the test data for a Secure Hash Algorithm 512 (SHA-512) test.

It returns a pointer to the SHA-512 test configuration.

Format

```
t_enc_drv_test * sha512_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

aes_raw_get_drv_test_data

Use this function to obtain the test data for an AES RAW test.

It returns a pointer to the AES RAW algorithm test configuration.

Format

```
t_enc_drv_test * aes_raw_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

aes_ctr_get_drv_test_data

Use this function to obtain the test data for an AES CTR test.

It returns a pointer to the AES CTR algorithm test configuration.

Format

```
t_enc_drv_test * aes_ctr_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

ipsec_null_int_get_drv_test_data

Use this function to obtain the test data for an IPsec NULL integrity check driver test.

It returns a pointer to the test configuration.

Format

```
t_enc_drv_test * ipsec_null_int_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

ipsec_null_enc_get_drv_test_data

Use this function to obtain the test data for an IPsec NULL encryption driver test.

It returns a pointer to the test configuration.

Format

```
t_enc_drv_test * ipsec_null_enc_get_drv_test_data ( void )
```

Arguments

Argument
None.

Return values

Return value	Description
ENC_DRVTEST_SUCCESS	Successful execution.
Else	See Error Codes.

4.3 Types and Definitions

Test Types

The test types are as follows:

Type	Value	Description
ENC_DRVTEST_TEST_HASH	0U	Test for hash.
ENC_DRVTEST_TEST_ENCRYPT	1U	Test for encryption.
ENC_DRVTEST_TEST_DECRYPT	2U	Test for decryption.
ENC_DRVTEST_TEST_ENDECRYPT	3U	Test for encryption and decryption.
ENC_DRVTEST_TEST_SIGN	4U	Test specifically for the DSS encryption algorithm.

t_enc_drv_test

The *t_enc_drv_test* structure specifies the test format:

Element	Type	Description
edtc_init_fn	t_enc_drv_init_fn	The init function of an algorithm.
edtc_init_fn_ext	t_enc_drv_init_fn	The init function of a second algorithm that is needed by the first.
edtc_reg_fun	t_edt_register_fn	The register function for the algorithm.
edtc_name [ENC_DRVTEST_NAME_SIZE]	uint8_t	The amount of test data.
p_edtc_data	t_enc_drv_testdata *	The test data.
edtc_data_cnt	uint8_t	The amount of test data.
edtc_result	t_enc_drvtest_ret	The overall test result.

t_enc_drv_testdata

The structure *t_enc_drv_testdata* contains function pointers that are used by the module to run the driver:

Element	Type	Description
p_edtd_data_in	uint8_t *	The input data.
edtd_data_in_length	uint16_t	The length of the input data.
p_edtd_cypher	t_enc_cypher_data *	Cypher data for encryption.
p_edtd_cypher2	t_enc_cypher_data *	Cypher data for encryption.
p_edtd_data_out	uint8_t *	The output compare data.
edtd_data_out_length	uint16_t	The output data length.
edtd_test_type	uint8_t	The test type.
edtd_test_result	t_enc_drvtest_ret	The test result.

t_enc_test_cb

The **t_enc_test_cb** typedef defines the callback function called when the driver test finishes.

Format

```
typedef void ( * t_enc_test_cb ) (
    uint16_t    idx,
    t_enc_ret   res )
```

Arguments

Parameter	Description	Type
idx	The index.	uint16_t
res	The test result.	t_enc_ret

t_enc_drvtest_cb

The **t_enc_drvtest_cb** typedef defines the callback function called when the driver test finishes.

Format

```
typedef void ( * t_enc_drvtest_cb )( t_enc_drv_test * p_driver_test )
```

Arguments

Parameter	Description	Type
p_driver_test	A pointer to the driver test.	t_enc_drv_test *

4.4 Error Codes

The table below lists the error codes that may be generated by the API calls.

Error code	Value	Meaning
ENC_DRVTEST_SUCCESS	0U	No error; driver test passed.
ENC_DRVTEST_CMP_FAILED	1U	Driver test failed during compare.
ENC_DRVTEST_NOT_RUN	2U	Test was not run.
ENC_DRVTEST_REGISTER_ERR	3U	Error during driver registration.
ENC_DRVTEST_INIT_ERR	4U	Error during driver initialization.
ENC_DRVTEST_START_ERR	5U	Error during driver start.
ENC_DRVTEST_STOP_ERR	6U	Error during driver stop.
ENC_DRVTEST_DELETE_ERR	7U	Error during driver delete.
ENC_DRVTEST_DEREGISTER_ERR	8U	Error during driver deregistration.
ENC_DRVTEST_ALLOC_ERR	9U	Error during instance allocation.
ENC_DRVTEST_FREE_ERR	10U	Error during free instance.
ENC_DRVTEST_HASH_ERR	11U	Error during hash calculation.
ENC_DRVTEST_ENCRYPT_ERR	12U	Error during encryption.
ENC_DRVTEST_DECRYPT_ERR	13U	Error during decryption.
ENC_DRVTEST_OVERFLOW_ERROR	14U	Buffer overflow error.
ENC_DRVTEST_OUTLEN_ERROR	15U	Output length too long.
ENC_DRVTEST_ERROR	16U	General error.

5 Integration

The Encryption Test Suite is designed to be as open and as portable as possible. No assumptions are made about the functionality, the behavior, or even the existence, of the underlying operating system. For the system to work at its best, perform the porting outlined below. This is a straightforward task for an experienced engineer.

5.1 OS Abstraction Layer

The test suite uses the OS Abstraction Layer (OAL) that allows it to run seamlessly with a wide variety of RTOSes, or without an RTOS.

The test suite uses the following OAL components:

OAL Resource	Number Required
Tasks	0
Mutexes	1
Events	0

5.2 PSP Porting

The Platform Support Package (PSP) is designed to hold all platform-specific functionality, either because it relies on specific features of a target system, or because this provides the most efficient or flexible solution for the developer.

The test suite makes use of the following standard PSP function:

Function	Package	Element	Description
psp_memcmp()	psp_base	psp_string	Compares two blocks of memory.