Encryption Test Suite User Guide

BETA DRAFT

Version 1.80 BETA

For use with Encryption Test Suite versions 1.23 and above

Exported on 07/27/2018

Table of Contents

1 System Overview

This chapter contains the fundamental information for this module.

The component sections are as follows:

- Introduction – describes the main elements of the module.
- Feature Check – summarizes the main features of the module as bullet points.
- Packages and Documents – the *Packages* section lists the packages that you need in order to use this module. The *Documents* section lists the relevant user guides.
- Change History – lists the earlier versions of this manual, giving the software version that each manual describes.

1.1 Introduction

This guide is for those who want to use HCC'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): its variants are AES-CBC, AES-CBC RAW, AES-CFB, AES-CTR, AES-CCM, AES-CCM8, AES-GCM, AES-CMAC and AES-XCBC-MAC.
- Data Encryption Standard (DES) and DES RAW.
- Digital Signature Standard (DSS).
- Elliptical Curve Cryptography: ECDH and ECDSA.
- Ephemeral Diffie-Hellman (EDH).
- Message Digest Algorithm 5 (MD5): MD5, MD4, HMAC-MD5, and HMAC-MD5-96.
- RSA Signature Algorithm (RSA).
- Secure Hash Algorithm (SHA): SHA-1, SHA-256, SHA-384, SHA-512, SHA-1 HMAC, and SHA-1 HMAC-96.
- Tiger 128, Tiger 160, Tiger 192 and Tiger 192 HMAC.
- Triple Data Encryption Standard (3DES) and 3DES RAW.

In addition you can run 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.

Note: Before running any of these tests, you must configure certain options and run one or more functions within the relevant module to register its tests with this test module. For example, to run the Tiger tests, you must call the function **tiger_register_tests()** within the Tiger module.

1.2 Feature Check

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

- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Tests the Embedded Encryption Manager (EEM).
- Gives complete logical coverage test of each algorithm.

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, see Product Information on 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.

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.

Encryption Test Suite User Guide

This is this document.

1.4 Change History

This section describes past changes to this manual.

- To view or download earlier manuals, see Encryption PDFs.
- For the history of changes made to the package code itself, see History: enc_test.

The current version of this manual is 1.80 BETA. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.80B	2018-07-27	1.23	Removed *Test Data Functions* section.
1.70B	2018-02-22	1.22	Modified *Introduction*.
1.60B	2017-09-18	1.21	Added tests for CCM and CCM-8.
1.50B	2017-06-15	1.20	Added tests for MD4, GCM. New *Change History* format.
1.40B	2017-01-10	1.16	Added tests for Tiger hash, Tiger HMAC and AES CBC.
1.30B	2016-06-20	1.14	Added new tests.
1.20B	2016-05-06	1.10	Added new tests.
1.10B	2016-03-09	1.08	Added new tests.
1.00B	2016-02-23	1.06	First online version.

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.
Vectors.c and **Vectors.h**	Files for big number arithmetic.

2.4 Version File

The file **src/version/ver_enc_test.h** contains the version number of this 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 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 33.

ENC_DRIVER_TEST_OUTBUF_SIZE

The size of the output buffer. The default value is 520.

4 Application Programming Interface

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

> **Note:** Before running tests for any algorithm, you must run the relevant test registration function, as described in the algorithm's manual.

4.1 Module Management

These functions control operation of the test suite itself:

Function	Description
enc_test_init()	Initializes the Encryption Test Suite and allocates the required resources.
enc_test_reg_callback()	Registers the callback function that is called when an EEM test ends.

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:

Function	Description
enc_driver_test_init()	Initializes the algorithm tests and allocates the required resources.
enc_driver_test_register()	Registers an algorithm test.
enc_driver_reg_callback()	Registers the callback function that is called when an algorithm test ends.

These are followed by the Test Data Functions.

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	A pointer to the driver test structure.	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.

4.3 Types and Definitions

This section describes the main elements that are defined in the API Header file.

Test Types

The test types are as follows:

Type	Value	Description
ENC_DRVTEST_TEST_HASH	0	Test for hash.
ENC_DRVTEST_TEST_ENCRYPT	1	Test for encryption.
ENC_DRVTEST_TEST_DECRYPT	2	Test for decryption.
ENC_DRVTEST_TEST_ENDECRYPT	3	Test for encryption and decryption.
ENC_DRVTEST_TEST_SIGN	4	Test specifically for the DSS encryption algorithm.
ENC_DRVTEST_TEST_HASH_1MB	5	Test for hash using 1MB buffer (input data is copied to establish 1MB buffer).

Stateful Flag

This flag shows whether a hash algorithm is stateful.

Type	Value	Description
ENC_FLAG_STATEFULL	0x8000	If set, marks that a hash algorithm is stateful.

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.
edtc_encr_bs	uint16_t	The encryption input data block size. This is set to 0 if there is no restriction on input data length.
edtc_decr_bs	uint16_t	The decryption input data block size. This is set to 0 if there is no restriction on input data length.
edtc_hash_bs	uint16_t	The hash input data block size. This is set to 0 if there is no restriction on input data length.

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 to be 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	0	No error; driver test passed.
ENC_DRVTEST_CMP_FAILED	1	Driver test failed during compare.
ENC_DRVTEST_NOT_RUN	2	Test was not run.
ENC_DRVTEST_REGISTER_ERR	3	Error during driver registration.
ENC_DRVTEST_INIT_ERR	4	Error during driver initialization.
ENC_DRVTEST_START_ERR	5	Error during driver start.
ENC_DRVTEST_STOP_ERR	6	Error during driver stop.
ENC_DRVTEST_DELETE_ERR	7	Error during driver delete.
ENC_DRVTEST_DEREGISTER_ERR	8	Error during driver deregistration.
ENC_DRVTEST_ALLOC_ERR	9	Error during instance allocation.
ENC_DRVTEST_FREE_ERR	10	Error during free instance.
ENC_DRVTEST_HASH_ERR	11	Error during hash calculation.
ENC_DRVTEST_ENCRYPT_ERR	12	Error during encryption.
ENC_DRVTEST_DECRYPT_ERR	13	Error during decryption.
ENC_DRVTEST_OVERFLOW_ERROR	14	Buffer overflow error.
ENC_DRVTEST_OUTLEN_ERROR	15	Output length too long.
ENC_DRVTEST_ERROR	16	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	2
Mutexes	0
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. For full details of its functions and macros, see the *HCC Base Platform Support Package User Guide*.

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

Function	Package	Element	Description
psp_memcmp()	psp_base	psp_string	Compares two blocks of memory.
psp_memset()	psp_base	psp_string	Sets the specified area of memory to the defined value.
psp_getrand()	psp_base	psp_string	Returns a 32 bit random number.