Digital Signature Standard User Guide

Version 1.50 BETA

For use with Digital Signature Standard (DSS) module versions 1.09 and above

Date: 22-Feb-2018 10:20

All rights reserved. This document and the associated software are the sole property of HCC Embedded. Reproduction or duplication by any means of any portion of this document without the prior written consent of HCC Embedded is expressly forbidden.

HCC Embedded reserves the right to make changes to this document and to the related software at any time and without notice. The information in this document has been carefully checked for its accuracy; however, HCC Embedded makes no warranty relating to the correctness of this document.

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 implement encryption using the Digital Signature Standard (DSS). The DSS module implements the Digital Signature Standard, which uses the Digital Signature Algorithm (DSA).

Note: This algorithm is rarely used now. ECDSA has superseded it; for details of this, see the *HCC Elliptic Curve Cryptography User Guide*.

Overview

DSS is a digital signature algorithm that is used to sign data. The algorithm uses modular calculation over big numbers to calculate the signature value.

For proper operation, the SHA-1 hash algorithm must be registered with the DSS driver. Input data is hashed internally by DSS (using SHA-1), then the hash value is used to generate a signature.

The algorithm is not stateful.

enc_driver_encrypt()

The EEM function **enc_driver_encrypt()** is used to sign input data.

p_in[] points to the data to be signed. The length of the data (*in_len*) does not need to be aligned.

In this case the relevant parts of the *t_enc_cypher_data* structure are as follows:

Element	Type	Description
p_ecd_key	void *	A pointer to the buffer storing the DER-encoded private key for DSA as specified by the X.509 standard.
ecd_key_size	uint16_t	The length of the key in bytes.

Other fields are discarded but should be set to NULL.

The output data from **enc_driver_encrypt()** is the signature, stored in *p_out[]*.

You must set the output length, *p_out_len*, to the output buffer size. The output buffer must be able to store DER-encoded ECC points. In the worst case its size must be 9 + 40.

enc_driver_decrypt()

The EEM function **enc_driver_decrypt()** is used to check the signature of given data.

p_in[] points to the data to be checked.

In this case the relevant parts of the *t_enc_cypher_data* structure are as follows:

Element	Type	Description
p_ecd_key	void *	A pointer to the buffer storing the DSA public key (X.509 certificate subject public key information).
ecd_key_size	uint16_t	The length of the public key in bytes.

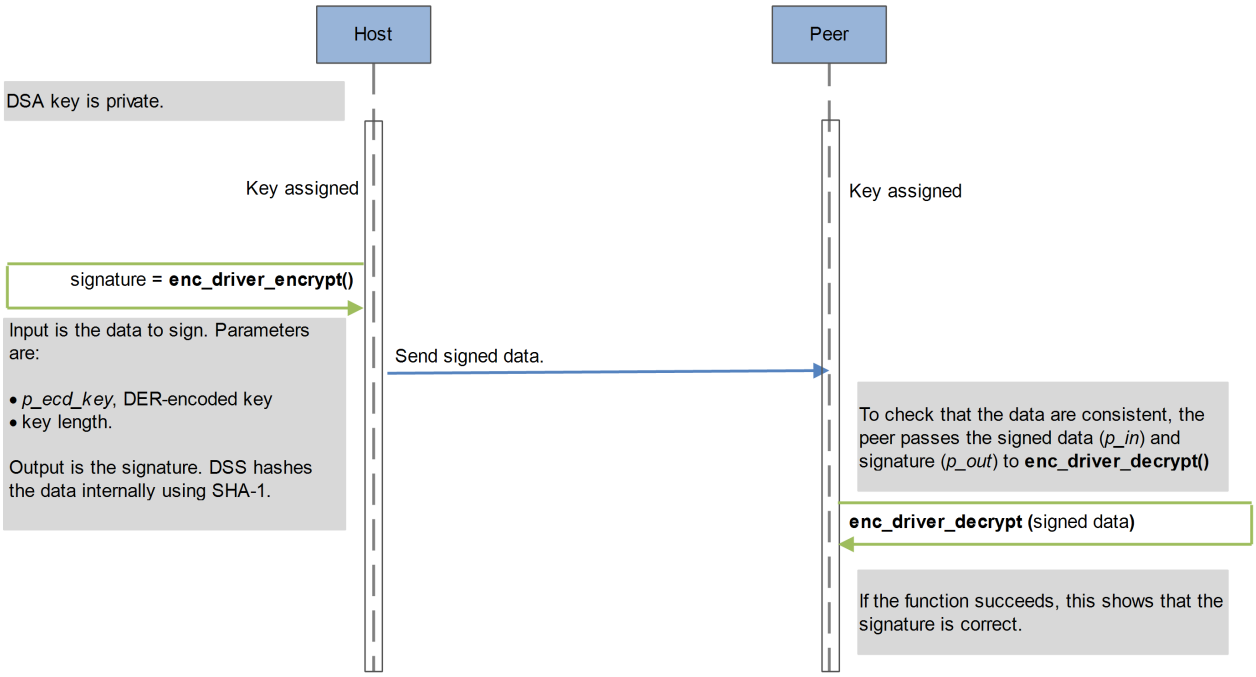
Other fields are discarded but should be set to NULL.

Set the output data from **enc_driver_decrypt()** to the signature data to be checked, stored in *p_out[]*.

You must set the output length, *p_out_len*, to the length of the signature.

Sequence Diagram

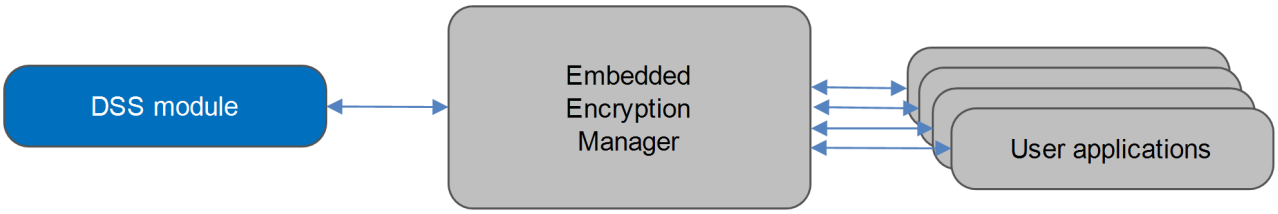
The following sequence diagram shows the process:



Using the Module

You register the DSS module with HCC's Embedded Encryption Manager (EEM), making it usable by other applications (for example, HCC's TLS/DTLS) through a standard interface. The EEM is the core component of HCC's encryption system.

The system structure is shown below:



A complete test suite is available for validating the algorithms.

Note:

- Although every attempt has been made to simplify the system's use, to get the best results you must understand clearly the requirements of the systems you design.
- HCC Embedded offers hardware and firmware development consultancy to help you implement your system; contact sales@hcc-embedded.com.

1.2 Feature Check

The main features of the DSS module are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Conforms to the HCC Coding Standard including full MISRA compliance.
- Designed for integration with both RTOS and non-RTOS based systems.
- Conforms to the HCC Embedded Encryption Manager (EEM) standard and is compatible with the EEM.
- Integral test suite gives complete logical coverage test of the 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_dss	The DSS package described in this document.

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 Encryption Test Suite User Guide

This document describes how to run tests to validate the algorithms.

HCC Digital Signature Standard User Guide

This is this document.

1.4 Change History

This section describes past changes to this manual.

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

The current version of this manual is 1.50 BETA. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.50B	2018-02-22	1.09	Extended *Introduction*, added new test options and function.
1.40B	2017-06-15	1.08	New *Change History* format.
1.30B	2017-04-12	1.07	Added DSS_BUF_LEN option.
1.20B	2017-01-10	1.05	Added lists of functions to API headers.
1.10B	2016-03-09	1.04	Added Change History.
1.00B	2015-02-11	1.03	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_sw_dss.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_sw_dss.h** contains the configurable parameters of the system. Configure these as required. This is the only file in the module that you should modify.

2.3 System File

The file **src/enc/software/dss/dss.c** contains the source code. **This file should only be modified by HCC** .

2.4 Test File

The file **src/enc/test/test_dss.c** contains the test source code. **This file should only be modified by HCC** .

2.5 Version File

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

3 Configuration Options

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

DSS_INSTANCE_NR

The maximum number of DSS instances. The default is 1.

DSS_SHA1_OUT_LEN

The SHA-1 hash output length. The default is SHA1_OUT_LEN.

DSS_BUF_LEN

The DSS buffer length. The default is 256.

DSS_TEST_ENABLE

Keep the default of 1 to enable the DSS test suite. Otherwise, set this to 0.

DSS_TEST_DSS_INITFN

The DSS encryption driver initialization function. The default is *&dss_init_fn*. Redefine this if you want to use another driver for a compatibility check.

4 Application Programming Interface

This section describes the Application Programming Interface (API) functions and the error codes.

4.1 Functions

Call the initialization function from the EEM to register the algorithm with it. Call the test function to register the RSA tests with the EEM test module.

The functions are the following:

Function	Description
dss_init_fn()	Called from the EEM, this registers the DSS algorithm with it.
dss_register_sha1()	Registers a Secure Hash Algorithm 1 (SHA-1) interface handle with the DSS module.
dss_register_tests()	Registers the DSS tests with the EEM test module.

dss_init_fn

Call this initialization function from the EEM to register the DSS algorithm with it.

This forwards the *t_enc_driver_fn* structure containing DSS functions to the EEM. This structure is described in the the *HCC Embedded Encryption Manager User Guide*.

Format

```
t_enc_ret dss_init_fn ( t_enc_driver_fn const * * const pp_encdriver )
```

Arguments

Parameter	Description	Type
pp_encdriver	A pointer to a *t_enc_driver_fn* structure containing DSS functions.	t_enc_driver_fn * *

Return Values

Return value	Description
ENC_SUCCESS	Successful execution.
ENC_INVALID_ERR	The module has already been initialized.

dss_register_sha1

Use this function to register a Secure Hash Algorithm 1 (SHA-1) interface handle with the DSS module.

> **Note:** Call this function just once during system initialization.

Format

```
t_enc_ret dss_register_sha1( t_enc_ifc_hdl sha1_hdl )
```

Arguments

Parameter	Description	Type
sha1_hdl	The SHA-1 interface handle.	t_enc_ifc_hdl

Return Values

Return value	Description
ENC_SUCCESS	Successful execution.

dss_register_tests

Call this function to register the DSS tests with the EEM test module.

Once you have registered the tests, you can execute the test suite as directed in the *HCC Encryption Test Suite User Guide*.

> **Note:** The DSS_TEST_ENABLE configuration option must be set to 1 to enable this function.

Format

```
t_enc_ret dss_register_tests ( void )
```

Arguments

None.

Return Values

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

4.2 Error Codes

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

Error code	Value	Meaning
ENC_SUCCESS	0	Successful execution.
ENC_INVALID_ERR	1	The module has already been initialized.

5 Integration

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

The module 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. For full details of these elements, see the *HCC Base Platform Support Package User Guide*.

The module makes use of the following standard PSP function:

Function	Package	Element	Description
PSP_WR_8BITARRAY_OFFSET()	psp_base	psp_array	Writes the offset in an 8 bit array.

The module makes use of the following standard PSP macro:

Macro	Package	Element	Description
PSP_WR_BE16	psp_base	psp_endianness	Writes a 16 bit value to be stored as big-endian to a memory location.

The module uses the following big number arithmetic functions from the EEM's Big Number Arithmetic API. These are described in the *HCC Embedded Encryption Manager User Guide*.

Function	Description
bn_add()	Adds two numbers.
bn_assign_be_buf()	Assigns a little-endian buffer to a big number, based on a big-endian buffer.
bn_assign_le_buf()	Assigns a buffer to a big number, based on a little-endian buffer.
bn_compare()	Compares two big numbers.
bn_get_be_buf()	Exports a big number to a big-endian buffer.
bn_get_le_buf()	Exports a big number to a little-endian buffer.
bn_modulo()	Calculates the remainder of *p_a* divided by *p_mod*.
bn_get_power_modulo()	Calculates *p_a* raised to the power of *p_e*, modulo *p_m*, and stores the result in *p_r*.
bn_inverse_modulo()	Calculates the modular multiplicative inverse of *p_a* with modulus *p_modulo*.
bn_modular_multiplication()	Counts *a*b* mod *modulo* using the Montgomery algorithm.

Note: To improve performance, you can replace these functions with optimized or hardware-supported versions.