Triple Data Encryption Standard User Guide

Version 1.20

For use with Triple Data Encryption Standard (3DES) module versions 1.08 and above

Date: 15-Jun-2017 15:12

Table of Contents

1 System Overview

1.1 Introduction

This guide is for those who want to implement bulk encryption using the Triple Data Encryption Standard (3DES). The 3DES uses a symmetric key algorithm, with the same key used both to encrypt and decrypt the data. The 3DES module implements the Triple Data Encryption Standard bulk encryption algorithm.

This module provides two types of 3DES encryption/decryption:

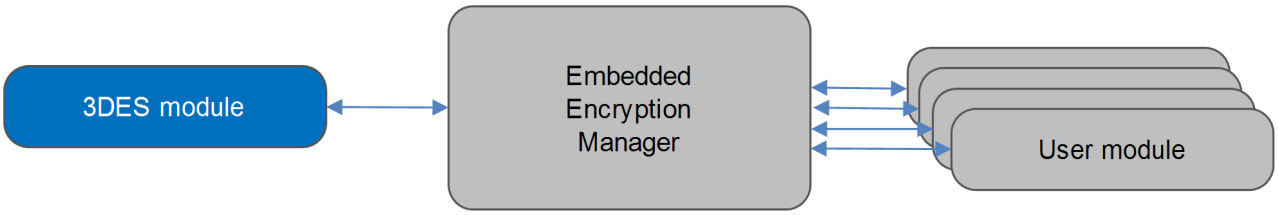
- Raw DES – this is the original form of DES and uses a single DES iteration. This is used by Network Time Protocol (NTP) authorization.
- Triple DES – this superseded DES and provides improved security.

Triple DES involves three iterations, as follows:

1. dataout = DES_encrypt(key1, datain)
2. dataout1 = DES_dencrypt(key2, dataout)
3. dataout2 = DES_encrypt(key3, dataout1)

You register the 3DES module with HCC's Embedded Encryption Manager (EEM), making it usable by other applications (for example, HCC's TLS/SSL) through a standard interface. The EEM is the core component of HCC's encryption system.

The system structure is shown below:



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 3DES 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 HCC's Embedded Encryption Manager (EEM) standard and is compatible with the EEM.
- Can be verified by using the HCC Encryption Test Suite.
- Provides Triple DES and raw DES options.

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_tdes	The 3DES 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 Triple Data Encryption Standard 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 Archive: 3DES User Guide.
- For the history of changes made to the package code itself, see History: enc_tdes.

The current version of this manual is 1.20. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.20	2017-06-15	1.08	New *Change History* format.
1.10	2016-03-11	1.07	Added TDES Raw.
1.10 BETA	2016-01-29	1.05	Added *Change History*.
1.00	2015-02-11	1.05	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_tdes.h** is the only file that should be included by an application using this module. It defines the Application Programming Interface.

2.2 Configuration File

The file **src/config/config_enc_sw_tdes.h** contains the single configurable parameter of the system. Configure this as required. This is the only file in the module that you should modify.

2.3 System File

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

2.4 Version File

The file **src/version/ver_enc_sw_tdes.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 Option

Set the single system configuration option in the file **src/config/config_enc_sw_tdes.h**.

TDES_TLS12_PADDING_METHOD

This controls padding generation. The values are:

- 0 (the default) – padding is generated consistent with PKCS #7. For details, see RFC 5652 section 6.3.
- 1 – use this for TLS 1.2 encryption. It generates padding in a manner consistent with RFC 5246 section 6.2.3.2.

4 Application Programming Interface

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

4.1 Functions

The functions are the following. Each of these may be called once by the EEM.

Function	Description
tdes_init_fn()	Called from the EEM, registers 3DES encryption/decryption. This forwards the structure containing 3DES functions to the EEM.
des_raw_init_fn()	Called from the EEM, registers RAW DES encryption/decryption. This forwards the structure containing RAW DES functions to the EEM.
tdes_raw_init_fn()	Called from the EEM, registers RAW 3DES encryption/decryption. This forwards the structure containing RAW 3DES functions to the EEM.

tdes_init_fn

Call this function once from the EEM to register 3DES encryption/decryption. This forwards the structure containing 3DES functions to the EEM.

Format

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

Arguments

Parameter	Description	Type
pp_encdriver	A pointer to a structure containing 3DES functions.	t_enc_driver_fn * *

Return Values

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

des_raw_init_fn

Call this function once from the EEM to register RAW DES encryption/decryption. This forwards the structure containing RAW DES functions to the EEM.

In RAW mode no padding is added. The size of the data to be encrypted must be a multiple of 8.

Format

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

Arguments

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

Return Values

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

tdes_raw_init_fn

Call this function once from the EEM to register RAW 3DES encryption/decryption. This forwards the structure containing RAW 3DES functions to the EEM.

In RAW mode no padding is added. The size of the data to be encrypted must be a multiple of 8.

Format

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

Arguments

Parameter	Description	Type
pp_encdriver	A pointer to a structure containing RAW 3DES functions.	t_enc_driver_fn * *

Return Values

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

4.2 Key Length

The key length is defined in the file **src/api/api_enc_sw_tdes.h**.

Name	Value	Description
TDES_INIVECT_SIZE	8	The 3DES initialization vector size.
TDES_KEY_LEN	24	The key length in bytes.

4.3 Error Codes

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

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

5 Integration

This section describes all aspects of the module that require integration with your target project. This includes porting and configuration of external resources.

5.1 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_memset()	psp_base	psp_string	Sets the specified area of memory to the defined value.