Advanced Encryption Standard User Guide

Version 1.10 BETA

For use with Advanced Encryption Standard (AES) module versions 1.09 and above

Date: 09-Mar-2016 15:06

Table of Contents

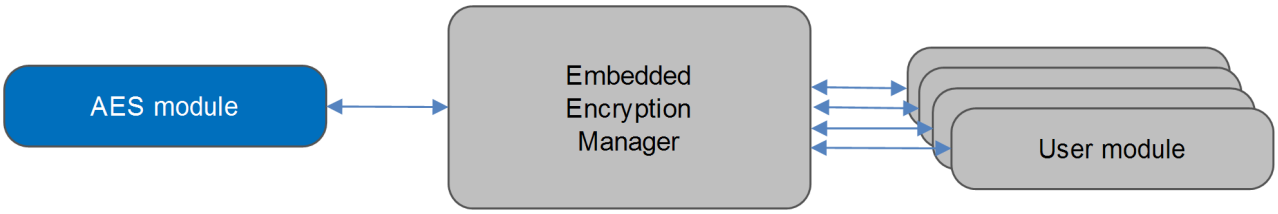
1 System Overview

1.1 Introduction

This guide is for those who want to implement bulk encryption using the Advanced Encryption Standard (AES). The AES uses a symmetric key algorithm, with the same key used to both encrypt and decrypt the data. The AES module implements the AES bulk encryption algorithm with Cipher Block Chaining (CBC). It supports AES RAW, AES CTR, AES-XCBC-MAC, and AES-XCBC-MAC-96.

You register the AES 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:



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.

This module supports the following AES variations:

- AES RAW – no padding is added to the input data.
- AES CTR – the counter (CTR) mode of AES. Again no padding is added to the input data. This is compatible with Encapsulating Security Payload (ESP), one of the Internet Protocol Security (IPsec) protocols.
- AES-XCBC-MAC and AES-XCBC-MAC-96 – here a Message Authentication Code (MAC) value is calculated.

1.2 Feature Check

The main features of the AES module are the following:

- It conforms to the HCC Advanced Embedded Framework.
- It conforms to the HCC Coding Standard including full MISRA compliance.
- It can be used with or without an RTOS.
- It conforms to HCC's Embedded Encryption Manager (EEM) standard and is compatible with the EEM.
- It supports AES RAW and AES CTR (RFC 3686).
- It supports AES-XCBC-MAC and AES-XCBC-MAC-96 (RFC 3566).
- It can be verified by using the HCC Encryption Test Suite.

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_aes	The AES 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 Advanced Encryption Standard User Guide

This is this document.

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_aes.txt** in the distribution package.

Version	Changes
1.09	Added support for AES-XCBC-MAC/AES-XCBC-MAC-96. Added support for AES RAW. Added support for AES CTR (AES counter mode).
1.08	Removed mutex clearing during initialization. This could cause an OS compile error.
1.07	Added overflow handling when calculating output data length.
1.06	Corrected generation of round key for 256 bit key.
1.05	Added macro to determine if padding is generated according to RFC 5652 or RFC 5246 (TLS 1.2).

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_aes.h** is the only file that should be included by an application using this module. It defines the Application Programming Interface (API) functions.

2.2 Configuration File

The file **src/config/config_enc_sw_aes.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/aes/aes.c** contains the source code.

This file should only be modified by HCC.

2.4 Version File

The file **src/version/ver_enc_sw_aes.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_aes.h**.

AES_INSTANCE_NR

The maximum number of AES algorithm instances. The default is 1.

AES_RAW_INSTANCE_NR

The maximum number of AES RAW instances. The default is 1.

AES_CTR_INSTANCE_NR

The maximum number of AES CTR instances. The default is 1.

AES_XCBC_MAC_INSTANCE_NR

The maximum number of AES XCBC instances. The default is 1.

AES_TLS12_PADDING_METHOD

This controls padding generation. The values are:

- 0 (the default) – padding is generated consistent with PKCS #7 (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 lengths, output buffer lengths, AES-CTR parameters and the error codes.

4.1 Functions

aes_init_fn

Call this function from the EEM to forward the structure containing AES functions to it.

AES adds padding bytes to the input data. Specify the padding method by using the configuration option AES_TLS12_PADDING_METHOD.

Format

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

Arguments

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

Return Values

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

aes_ctr_init_fn

Call this function from the EEM to forward the structure containing AES CTR functions to it.

This is the counter (CTR) version of the AES algorithm. This includes ESP compatibility, so it it accepts an 8 byte IV vector. The NONCE and counter value is passed with the key.

Format

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

Arguments

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

Return Values

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

aes_raw_init_fn

Call this function from the EEM to forward the structure containing RAW AES functions to it.

RAW AES means that no padding is added to the input data. The data length for encryption and decryption must be a multiple of the AES block size (16).

Format

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

Arguments

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

Return Values

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

aes_xcbc_mac_init_fn

Call this function from the EEM to forward the structure containing AES AES-XCBC-MAC or AES-XCBC-MAC-96 functions to it. This initializes AES-XCBC-MAC (96), based on RFC 3566.

The output data is the MAC value.

> **Note:** To use AES-XCBC-MAC-96 mode, set the Output Buffer Length to AES_XCBC_MAC_96_OUT.

Format

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

Arguments

Parameter	Description	Type
pp_encdriver	A pointer to a structure containing AES AES-XCBC-MAC or AES-XCBC-MAC-96 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 Types and Definitions

AES-CTR Parameters

Set these parameters for operating AES-CTR in compatibility with ESP.

Name	Value	Description
AES_CTR_ESP_IV_SIZE	8	Initialization vector.
AES_CTR_ESP_128_KEY_SIZE	24	128 AES key size + 8 bytes of hidden Initialization vector.
AES_CTR_ESP_256_KEY_SIZE	40	256 AES key size + 8 bytes of hidden Initialization vector.

AES-XCBC-MAC and AES-XCBC-MAC-96 Output Buffer Lengths

Set the encryption output buffer length to one of the following to realize AES-XCBC-MAC or AES-XCBC-MAC-96:

Name	Value	Description
AES_XCBC_MAC_96_OUT	12	Output size for AES-XCBC-MAC-96.
AES_XCBC_MAC_OUT	16	Output size for AES-XCBC-MAC.

Key Lengths

The key lengths are as follows:

Name	Value	Description
AES_128_KEY_LEN	16	128 bit AES key length in bytes.
AES_256_KEY_LEN	32	256 bit AES key length in bytes.

4.3 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

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 functions, see the *Platform Support Package (PSP) Base User Guide*.

The module makes use of the following standard PSP function:

Function	Package	Element	Description
psp_memcpy()	psp_base	psp_string	Copies a block of memory. The result is a binary copy of the data.

The module makes use of the following standard PSP macros:

Macro	Package	Element	Description
PSP_RD_BE32	psp_base	psp_endianness	Reads a 32 bit value stored as big-endian from a memory location.
PSP_RD_LE32	psp_base	psp_endianness	Reads a 32 bit value stored as little-endian from a memory location.
PSP_WR_BE32	psp_base	psp_endianness	Writes a 32 bit value to be stored as big-endian to a memory location.
PSP_WR_LE32	psp_base	psp_endianness	Writes a 32 bit value to be stored as little-endian to a memory location.