GM8136

AES-DES Coprocessor

User Guide

Rev.: 1.0

Issue Date: August 2014

REVISION HISTORY

GM8136 AES-DES User Guide

Date	Rev.	From	To
Aug. 2014	1.0	-	Original

Printed in Taiwan 2014

All information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in implantation or other life support application where malfunction may result in injury or death to persons. The information contained in this document does not affect or change Grain Media's product specification or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Grain Media or third parties. All information contained in this document was obtained in specific environments, and is presented as an illustration. The results obtained in other operating environments may vary.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS. In no event will Grain Media be liable for damages arising directly or indirectly from any use of the information contained in this document.

Grain Media, Inc.
5F, No. 5, Li-Hsin Road III, Hsinchu Science Park, Hsinchu City, Taiwan 300, R.O.C.

Grain Media's home page can be found at:
http://www.grain-media.com

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

AES Configuration

This chapter contains the following sections:

1.1 AES-DES Overview

The Data Encryption Standard (DES) is a previously predominant algorithm for encryption of the electronic data. The Advanced Encryption Standard (AES) is a specification for encryption of the electronic data.

The AES-DES cipher coprocessor provides an efficient hardware implementation of the DES/Triple-DES/AES algorithm for the high-performance encryption and decryption, which can be applied to various applications.

In the DES/Triple-DES configuration, FTAES020S supports four block cipher modes, including the ECB mode, CBC mode, CFB mode, and OFB mode. In the AES configuration, it supports five block cipher modes, including the ECB mode, CBC mode, CTR mode, CFB mode, and OFB mode.

GM8136 has a built-in AES function and constructs a functional security driver, which is attached to the Linux kernel.

1.2 Linux Configuration for AES

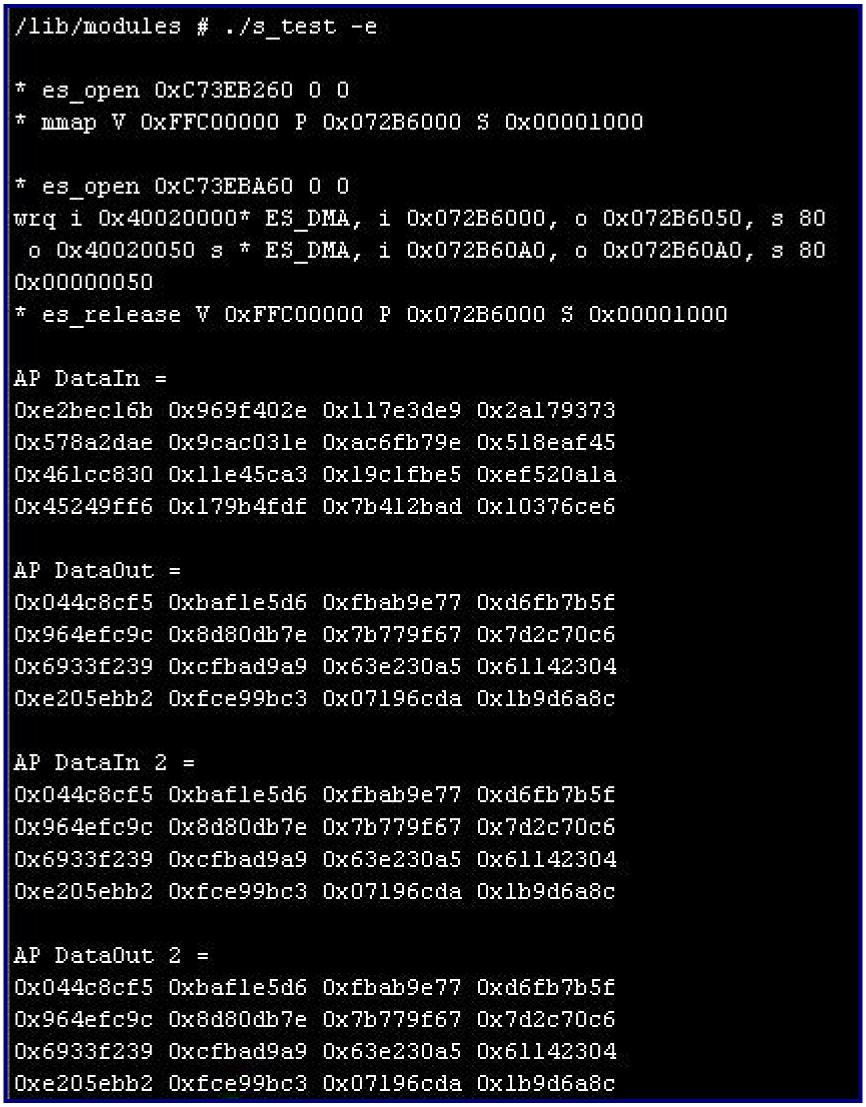
Please make sure that the following options are enabled in the Linux kernel before using the AES function of GM8136.

After configuration, please switch to the “arm-linux-3.3/module/AES_DES” directory and make a copy of the security kernel module, aes_des.ko. Please use **insmod** to insert this module to kernel and use **mdev –s** to generate a security device node as indicated in Figure 1-1.

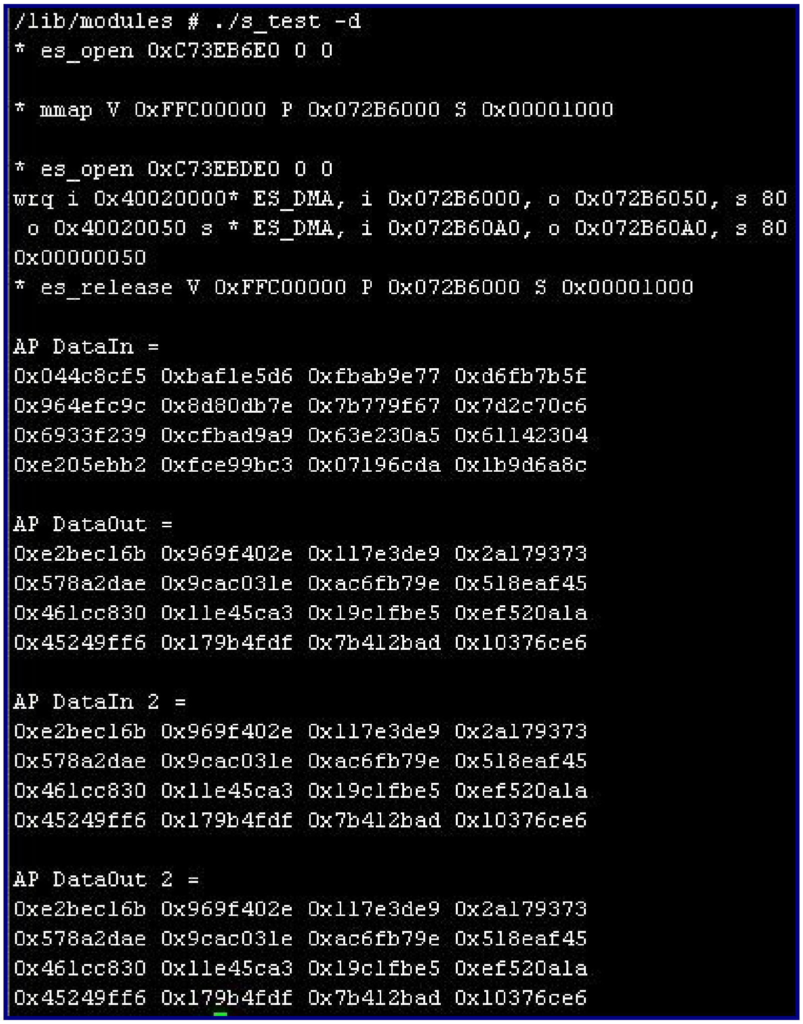
```
/ # insmod /lib/modules/aes_des.ko;mdev -s
# ls -al /dev/security
crw-rw----    1 root     root      10,  60 Jan  1 00:00 /dev/security
```

Figure 1-1. Insert Security Module

Users can use the test program after the GM8136 security is operating normally. Users can refer to arm-linux-3.3.\module\AES_DES\test for a sample.



```
/lib/modules # ./s_test -e

* es_open 0xC73EB260 0 0
* mmap V 0xFFC00000 P 0x072B6000 S 0x00001000

* es_open 0xC73EBA60 0 0
wrq i 0x40020000* ES_DMA, i 0x072B6000, o 0x072B6050, s 80
 o 0x40020050 s * ES_DMA, i 0x072B60A0, o 0x072B60A0, s 80
0x00000050
* es_release V 0xFFC00000 P 0x072B6000 S 0x00001000

AP DataIn =
0xe2bec16b 0x969f402e 0x117e3de9 0x2a179373
0x578a2dae 0x9cac031e 0xac6fb79e 0x518eaf45
0x461cc830 0x11e45ca3 0x19c1fbe5 0xef520a1a
0x45249ff6 0x179b4fdf 0x7b412bad 0x10376ce6

AP DataOut =
0x044c8cf5 0xbafle5d6 0xfbab9e77 0xd6fb7b5f
0x964efc9c 0x8d80db7e 0x7b779f67 0x7d2c70c6
0x6933f239 0xcfbad9a9 0x63e230a5 0x61142304
0xe205ebb2 0xfce99bc3 0x07196cda 0x1b9d6a8c

AP DataIn 2 =
0x044c8cf5 0xbafle5d6 0xfbab9e77 0xd6fb7b5f
0x964efc9c 0x8d80db7e 0x7b779f67 0x7d2c70c6
0x6933f239 0xcfbad9a9 0x63e230a5 0x61142304
0xe205ebb2 0xfce99bc3 0x07196cda 0x1b9d6a8c

AP DataOut 2 =
0x044c8cf5 0xbafle5d6 0xfbab9e77 0xd6fb7b5f
0x964efc9c 0x8d80db7e 0x7b779f67 0x7d2c70c6
0x6933f239 0xcfbad9a9 0x63e230a5 0x61142304
0xe205ebb2 0xfce99bc3 0x07196cda 0x1b9d6a8c
```



```
/lib/modules # ./s_test -d
* es_open 0xC73EB6E0 0 0

* mmap V 0xFFC00000 P 0x072B6000 S 0x00001000

* es_open 0xC73EBDE0 0 0
wrq i 0x40020000* ES_DMA, i 0x072B6000, o 0x072B6050, s 80
 o 0x40020050 s * ES_DMA, i 0x072B60A0, o 0x072B60A0, s 80
0x00000050
* es_release V 0xFFC00000 P 0x072B6000 S 0x00001000

AP DataIn =
0x044c8cf5 0xbaf1e5d6 0xfbab9e77 0xd6fb7b5f
0x964efc9c 0x8d80db7e 0x7b779f67 0x7d2c70c6
0x6933f239 0xcfbad9a9 0x63e230a5 0x61142304
0xe205ebb2 0xfce99bc3 0x07196cda 0x1b9d6a8c

AP DataOut =
0xe2bec16b 0x969f402e 0x117e3de9 0x2a179373
0x578a2dae 0x9cac031e 0xac6fb79e 0x518eaf45
0x461cc830 0x11e45ca3 0x19c1fbe5 0xef520a1a
0x45249ff6 0x179b4fdf 0x7b412bad 0x10376ce6

AP DataIn 2 =
0xe2bec16b 0x969f402e 0x117e3de9 0x2a179373
0x578a2dae 0x9cac031e 0xac6fb79e 0x518eaf45
0x461cc830 0x11e45ca3 0x19c1fbe5 0xef520a1a
0x45249ff6 0x179b4fdf 0x7b412bad 0x10376ce6

AP DataOut 2 =
0xe2bec16b 0x969f402e 0x117e3de9 0x2a179373
0x578a2dae 0x9cac031e 0xac6fb79e 0x518eaf45
0x461cc830 0x11e45ca3 0x19c1fbe5 0xef520a1a
0x45249ff6 0x179b4fdf 0x7b412bad 0x10376ce6
```

Figure 1-2. Usage of Test AP

1.3 AES Related Files

The AES related files are useful for the underneath operations. All paths are related to arm-linux-3.3/.

- module/include/security/security.h
  - This file includes the security driver and AP must include the head file to compile it.
- module/AES_DES/security.c
  - Implementation of the GM8136 AES function
- module/AES_DES/test/security_test.c
  - The sample code of using the AES/DES driver

Chapter 2

AES Driver Usage

This chapter contains the following section:

2.1 Programming Sequence

Please set the algorithm, mode, IV_addr[4], key_addr[8], data_length, DataIn_addr, and DataOut_addr of struct esreq_tag, which is defined in security.h before triggering encryption or decryption. struct esreq_tag will be the parameter of the IOCTL command.

```
typedef struct esreq_tag {
    int algorithm;
    int mode;
    unsigned int data_length;
    unsigned int key_length;
    unsigned int IV_length;
    unsigned int key_addr[8];
    unsigned int IV_addr[4];
    unsigned int DataIn_addr;
    unsigned int DataOut_addr;
    unsigned int Key_backup;
    unsigned int IV_backup;
} esreq;
```

Please fill the security algorithm and security mode of struct esreq_tag as shown below, which is defined in security.h. Please note that the CTR mode is only for AES, which means that setting the algorithm 0x0 or 0x2 with mode 0x20 will be invalid.

```
/* security algorithm */
#define Algorithm_DES                0x0
#define Algorithm_Triple_DES         0x2
#define Algorithm_AES_128            0x8
#define Algorithm_AES_192            0xA
#define Algorithm_AES_256            0xC
```

```
/* security mode */
#define ECB_mode          0x00
#define CBC_mode          0x10
#define CTR_mode          0x20
#define CFB_mode          0x40
#define OFB_mode          0x50
```

Please fill IV_addr[0]~[3], key_addr[0]~[7], and data_length of struct esreq_tag as shown in Table 2-1.

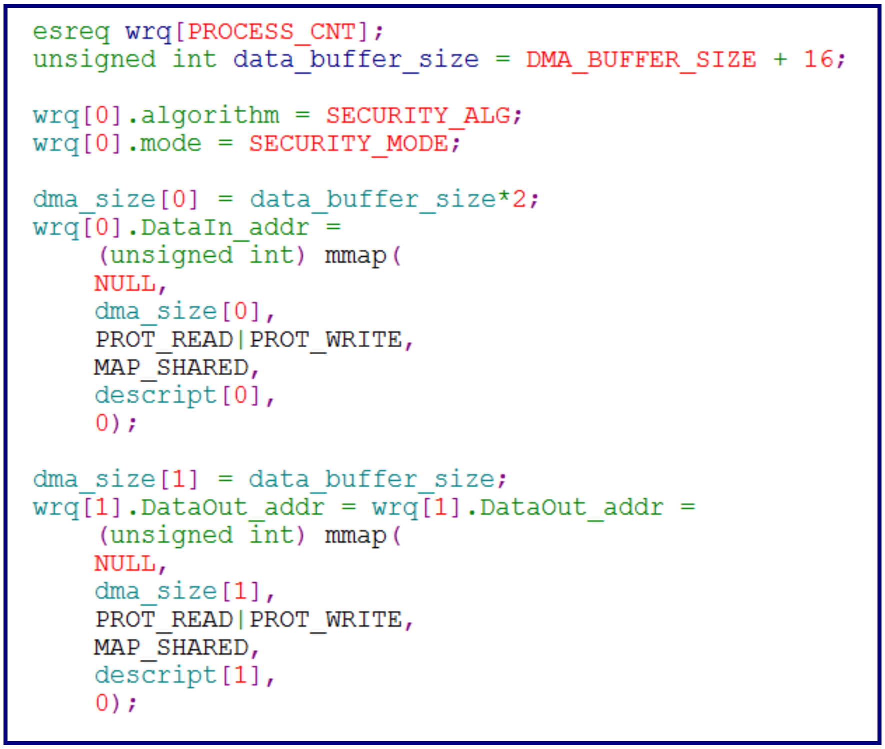
For example, when setting the algorithm, Algorithm_Triple_DES(0x2), please fill 64bit initial vector into IV_addr[0] and IV_addr[1] of struct esreq_tag and 192bit key into key_addr[0] ~ [5] of struct esreq_tag, then, set data_length of struct esreq_tag as the data size to be encrypted or decrypted plus extra 16bytes for storing the last initial vector.

To generate the initial vector and key, the IOCTL command, ES_GETKEY, is used. The information to generate the initial vector and key is listed in Table 2-1.

Table 2-1. Initial Vector and Key Generations

Algorithm	**Initial Vector**		**Key**		**Data Size**
	Bit Count	**Variable**	**Bit Count**	**Variable**	**Data_length (Byte Count)**
Algorithm_DES	64	IV_addr[0] ~ [1]	64	key_addr[0] ~ [1]	Data size (Should be multiples of 8) + 16
Algorithm_Triple_DES			192	key_addr[0] ~ [5]	
Algorithm_AES_128	128	IV_addr[0] ~ [3]	128	key_addr[0] ~ [3]	Data size (Should be multiples of 16) + 16
Algorithm_AES_192			192	key_addr[0] ~ [5]	
Algorithm_AES_256			256	key_addr[0] ~ [7]	

For better performance, the AES/DES driver provides the mmap function for data-in/data-out instead of copying data from the kernel space to user space. Please set DataIn_addr and DataOut_addr in the range of the mapped area. The recommended mapping size is the data size plus 16 if users expect data-out overwrites data-in, as wrq[1] in Figure 2-1, while the recommended mapping size is twice of the data size plus 16 if data-out does not overwrite data-in, as wrq[0] in Figure 2-1.



```
esreq wrq[PROCESS_CNT];
unsigned int data_buffer_size = DMA_BUFFER_SIZE + 16;

wrq[0].algorithm = SECURITY_ALG;
wrq[0].mode = SECURITY_MODE;

dma_size[0] = data_buffer_size*2;
wrq[0].DataIn_addr =
     (unsigned int) mmap(
     NULL,
     dma_size[0],
     PROT_READ|PROT_WRITE,
     MAP_SHARED,
     descript[0],
     0);

dma_size[1] = data_buffer_size;
wrq[1].DataOut_addr = wrq[1].DataOut_addr =
     (unsigned int) mmap(
     NULL,
     dma_size[1],
     PROT_READ|PROT_WRITE,
     MAP_SHARED,
     descript[1],
     0);
```

Figure 2-1. Recommended Mapping Size with DataIn_addr and DataOut_addr

After the algorithms, IV_addr, key_addr, data_length, DataIn_addr, and DataOut_addr are set, please send the IOCTL command, ES_DECRYPT, to decrypt the data or ES_ENCRYPT to encrypt the data in DataIn_addr with complete struct esreq_tag as the parameter of the IOCTL command to get the decrypted or encrypted data from DataOut_addr for later usage.

Please refer to module/AES_DES/test/security_test.c for better understanding of the programming sequence.