**Freescale Semiconductor**
Application Note

AN3085
Rev. 0, 06/2006

# SEC 2.x Descriptor Programmer's Guide Supplement

## Implementing AES-CMAC Using the SEC 2.x

*by* *Systems Engineering*
*Freescale Semiconductor, Inc.*

The advanced encryption standard-cipher message authentication code (AES-CMAC) is a mode of AES encryption used in IEEE® Std. 802.16e™ (WiMax) for subscriber station authentication with base stations. CMAC is a message authentication scheme built on two more basic modes of AES:

- Electronic code book (ECB)
- Cipher block chaining (CBC)

AES-CMAC is relatively new, and most commercially available encryption accelerators do not support it directly. The CMAC specification published by NIST can be accelerated with hardware that performs AES-ECB and AES-CBC. However, some processing that would likely be executed on a general-purpose CPU is required at each stage.

The SEC 2.x is a descriptor-based security engine integrated into several Freescale PowerQUICC™ products. It can accelerate AES-ECB and CBC, and, by extension, AES-CMAC. This application notes describes the hardware and software procedures for generating a CMAC using examples based on the MPC8555E SEC 2.0 hardware and device drivers. The techniques and code examples generically apply to all PowerQUICC devices with an SEC 2.x security engine. This document supplements the *SEC 2.0 Descriptor Programmer's Guide* (AN2755).

**Contents**





freescale™ semiconductor

All application software provided with this application note applies to the Freescale Linux-based security driver. The CPU executes the application software, prepares the crypto request data structure, and launches the crypto operation through an **ioctl** call.

# 1 AES-CMAC Basics

AES-CMAC is a new mode of AES for message authentication only. The IEEE 802.16e standard uses this AES mode to authenticate control messages. The PowerQUICC SEC 2.x hardware cannot perform full AES-CMAC in a single descriptor as it can for many of the more established AES modes. However, with some modification of the 802.16e management software, the SEC can complete a CMAC operation in a single descriptor.

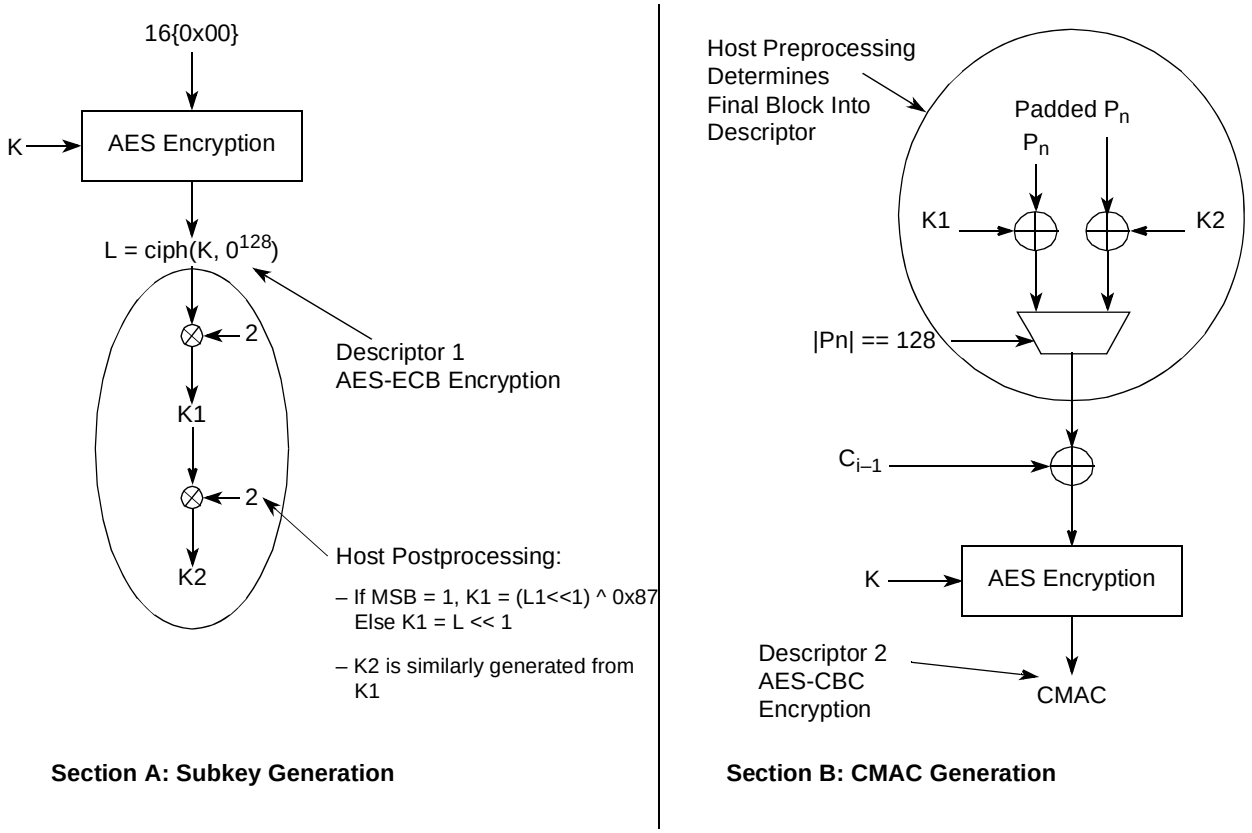


**Figure 1. SEC-Accelerated CMAC**

The procedure for an SEC-accelerated CMAC is as follows:

1. When a secure 802.16e connection is established, generate an AES symmetric key for use in AES-CMAC message authentication.
2. With this key, encrypt 16 bytes of zeros, shown in Section A of Figure 1 as 16{0x00}, with the SEC in AES-ECB mode.
3. The CPU software processes the output of this operation to generate two subkeys, K1 and K2.
4. Save K1 and K2 in memory.

   Each time the connection performs a CMAC, these two subkeys are used. Subkey generation is performed only one time.

5. The CMAC authentication value is generated using AES-CBC encryption as shown in Section B of Figure 1.
6. The message/PDU is preprocessed to generate Pn or Padded Pn, which is XORed with K1 or K2, respectively.

   If the last block of the message is 16 bytes long, Pn is generated, otherwise Padded Pn.
7. The Pn block or the Padded Pn block is concatenated with the rest of the message Ci-1.
8. The AES-CBC descriptor is prepared and launched.

   The preprocessing includes checking whether the input message size is an integer multiple of 16 bytes. If not, CPU software adds padding and then XORs the input message with either K1 or K2.
9. The result of this operation is fed into the AES-CBC encryption process and the descriptor is prepared and launched.

   The output context of this AES-CBC operation is CMAC.

For information on the security principles and the details of the CMAC algorithm, refer to Section 4, "References."

# 2 CMAC Software

The CMAC software discussed here is an add-on to the SEC2 kernel (Linux 2.6.11) downloadable module. For details on this driver software, refer to the *SEC 2.0 Reference Device Driver User's Guide* (SEC2SWUG) and the corresponding driver software, both of which are available from the MPC8555E page at the Freescale web site listed on the back cover of this document.

To run the software as discussed in this application note, t have a MPC8555/41 CDS platform and a board-support package (BSP) for that target platform. Because it is so easily available, the Freescale LTIB-based BSP is used for building the bootloader, kernel, and SEC2 driver and modules. It is beyond the scope of this document to describe the BSP procedure for the MPC8555CDS platform. If you are interested in a LTIB-based BSP for running the SEC2 driver and modules on the MPC8555CDS platform, refer to the Freescale application note entitled *MPC8555E Security Quick Start Guide* (AN3075).

## 2.1 Configuring and Building the CMAC Module

The downloadable CMAC kernel has several tunable parameters that are explained in this section. The following line prints a debug message, if it is defined:

```
//#define DEBUG_CMAC
```

A CMAC can be generated for AES ciphers with key lengths of either 128, 192, or 256 bits. Define one of these macros before compiling and running the module on the target MPC8555CDS platform. There is no guard against the user error of defining more than one of these macros.

```
//#define AES_128
//#define AES_192
#define AES_256
```

For each selection of cipher block (AES_128, AES_192, or AES_256) four tests (TEST1, TEST2, TEST3 and TEST4) can be run. Define one of the four TEST*n* macros. For each of these macros, a NIST published known input message is taken. Then the SEC2 performs the cascaded AES crypto operations, generates

the CMAC, and compares it with the NIST published values. Refer to the `Updated_CMAC_Examples.pdf` document published by NIST. There is no guard against the user error of defining more than one of these marcos.

```
//#define TEST1
//#define TEST2
//#define TEST3
#define TEST4
```

To compile the CMAC software successfully, have a working kernel source tree (preferably 2.6.11) and the SEC2 driver code to place on the kernel tree. For an example, see the Freescale application note entitled *MPC8555E Security Quick Start Guide* (AN3075), which describes the LTIB approach of building the kernel and SEC2 drivers/modules.

Assuming that the SEC2 driver is up and running in a Linux environment, you must add the CMAC features to the SEC2 module by adding the following files.

- `testCmac.c : $KERNEL_SRC/drivers/sec2x-test/`
- `testAll.c : $KERNEL_SRC/drivers/sec2x-test/`
- `Makefile: $KERNEL_SRC/drivers/sec2x-test/`

Next, compile the kernel and build the modules. In the `sec2x-test` directory, an sec2drvTest.ko module is created. For simplicity, all other traditional SEC2 testing features are commented out in the `testAll.c` file and only the CMAC test procedure is enabled. When the module runs, notice that the CMAC results are displayed on the console. An example test result is shown in next section.

## 2.2 Example Run

LTIB, which is based on the Linux 2.6 kernel and rootfilesystem, is deployed to the MPC8555CDS target platform as described in the *MPC8555E Security Quick Start Guide* (AN3075). From the Linux prompt, install the module. Example 1 shows how the console display lists the key, message, subkeys cipher block chosen, and the CMAC derived.

**Example 1. Running the CMAC Module**

```
/lib/modules/2.6.11/kernel/drivers/sec2x-test # insmod sec2drvTest.ko

*** Test CMAC subkey generation ***

==========================================

Key:
d107d924   60 3d eb 10 15 ca 71 be   2b 73 ae f0 85 7d 77 81   `=....q.+s...}w.
d107d934   1f 35 2c 07 3b 61 08 d7   2d 98 10 a3 09 14 df f4   .5,.;a..-.......

Message for subkey generation:
d107d984   00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ................

K1:
cec17c90   ca d1 ed 03 29 9e ed ac   2e 9a 99 80 86 21 50 2f   ....).........!P/

K2:
cec17ca0   95 a3 da 06 53 3d db 58   5d 35 33 01 0c 42 a0 d9   ....S=.X]53..B..

*** Test CMAC generation  ***
```

```
====================================
Block Cipher chosen : AES 256
TEST4 : Message Length 512 bits (64 Bytes)

Message for CMAC generation:
d107d944   6b c1 be e2 2e 40 9f 96   e9 3d 7e 11 73 93 17 2a   k....@...=~.s..*
d107d954   ae 2d 8a 57 1e 03 ac 9c   9e b7 6f ac 45 af 8e 51   .-.W......o.E..Q
d107d964   30 c8 1c 46 a3 5c e4 11   e5 fb c1 19 1a 0a 52 ef   0..F.\........R.
d107d974   f6 9f 24 45 df 4f 9b 17   ad 2b 41 7b e6 6c 37 10   ..$E.O...+A{.l7.

CMAC:
cec17cf0   e1 99 21 90 54 9f 6e d5   69 6a 2c 05 6c 31 54 10   ..!.T.n.ij,.l1T.
testAll(): All 2 Tests Passed
/lib/modules/2.6.11/kernel/drivers/sec2x-test #
```

## 2.3 Customizing the Software

The message for which the CMAC module must be generated is defined as a static array. Both the input and output arrays are padded with 0s to ensure that their size is a multiple of 16 bytes. This padding is just a place holder for actual padded data to be constructed by software, thus preventing the need for the SEC scatter gather mechanism. A message of arbitrary length could be passed to the CMAC software instead of a static array. If this message length is not a multiple of 16 bytes, the software must create the padding bits in a memory region that is probably not contiguous to the message supplied to the CMAC module. In this case, the SEC driver scatter gather capability can be used to fetch the non-contiguous padding.

Choosing static arrays simplifies data handling. The underlying cryptographic operation remains the same, thus validating that the CMAC software implements the algorithm accurately, which is the sole objective of this software. With minimal effort, you can add custom features to this CMAC software.

# 3 Conclusions

This document enables developers to accelerate CMAC generation using PowerQUICC devices with SEC 2.x functionality. Some customization may be needed to tailor the CMAC module to a specific environment. These changes should be minimal if the Freescale Linux-based security driver is used. The CMAC module is implemented using the SEC2 driver API.

# 4 References

The following reference documents for this application note are available either at the Freescale web site listed on the back cover of this document or at the NIST web site:

- *MPC8555E Security Quick Start Guide* (AN3075)
- *SEC 2.0 Reference Device Driver User's Guide* (SEC2SWUG)
- Updated CMAC examples:
  `http://csrc.nist.gov/publications/nistpubs/800-38B/Updated_CMAC_Examples.pdf`
- *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*
  `http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf`
- *SEC 2.0 Descriptor Programmer's Guide* (AN2755)

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

### *How to Reach Us:*

**Home Page:**
www.freescale.com

**email:**
support@freescale.com

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
1-800-521-6274
480-768-2130
support@freescale.com

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064, Japan
0120 191014
+81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

**For Literature Requests Only:**
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com



Document Number: AN3085
Rev. 0
06/2006

freescale™
semiconductor

**Freescale Semiconductor**
Application Note

AN3085
Rev. 0, 06/2006

# SEC 2.x Descriptor Programmer's Guide Supplement

## Implementing AES-CMAC Using the SEC 2.x

*by* *Systems Engineering*
*Freescale Semiconductor, Inc.*

The advanced encryption standard-cipher message authentication code (AES-CMAC) is a mode of AES encryption used in IEEE® Std. 802.16e™ (WiMax) for subscriber station authentication with base stations. CMAC is a message authentication scheme built on two more basic modes of AES:

- Electronic code book (ECB)
- Cipher block chaining (CBC)

AES-CMAC is relatively new, and most commercially available encryption accelerators do not support it directly. The CMAC specification published by NIST can be accelerated with hardware that performs AES-ECB and AES-CBC. However, some processing that would likely be executed on a general-purpose CPU is required at each stage.

The SEC 2.x is a descriptor-based security engine integrated into several Freescale PowerQUICC™ products. It can accelerate AES-ECB and CBC, and, by extension, AES-CMAC. This application notes describes the hardware and software procedures for generating a CMAC using examples based on the MPC8555E SEC 2.0 hardware and device drivers. The techniques and code examples generically apply to all PowerQUICC devices with an SEC 2.x security engine. This document supplements the *SEC 2.0 Descriptor Programmer's Guide* (AN2755).

**Contents**





freescale™ semiconductor

All application software provided with this application note applies to the Freescale Linux-based security driver. The CPU executes the application software, prepares the crypto request data structure, and launches the crypto operation through an **ioctl** call.

# 1 AES-CMAC Basics

AES-CMAC is a new mode of AES for message authentication only. The IEEE 802.16e standard uses this AES mode to authenticate control messages. The PowerQUICC SEC 2.x hardware cannot perform full AES-CMAC in a single descriptor as it can for many of the more established AES modes. However, with some modification of the 802.16e management software, the SEC can complete a CMAC operation in a single descriptor.

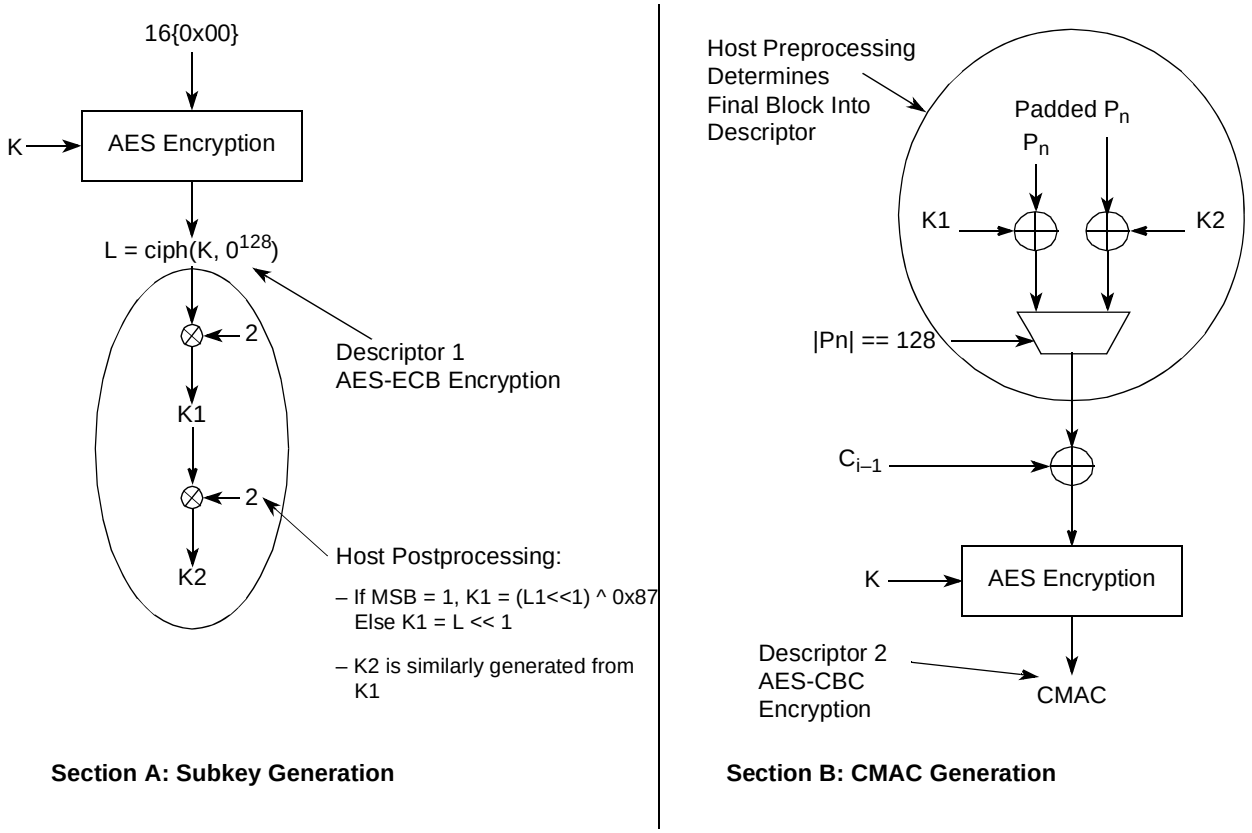


**Figure 1. SEC-Accelerated CMAC**

The procedure for an SEC-accelerated CMAC is as follows:

1. When a secure 802.16e connection is established, generate an AES symmetric key for use in AES-CMAC message authentication.
2. With this key, encrypt 16 bytes of zeros, shown in Section A of Figure 1 as 16{0x00}, with the SEC in AES-ECB mode.
3. The CPU software processes the output of this operation to generate two subkeys, K1 and K2.
4. Save K1 and K2 in memory.

   Each time the connection performs a CMAC, these two subkeys are used. Subkey generation is performed only one time.

5. The CMAC authentication value is generated using AES-CBC encryption as shown in Section B of Figure 1.
6. The message/PDU is preprocessed to generate Pn or Padded Pn, which is XORed with K1 or K2, respectively.

   If the last block of the message is 16 bytes long, Pn is generated, otherwise Padded Pn.
7. The Pn block or the Padded Pn block is concatenated with the rest of the message Ci-1.
8. The AES-CBC descriptor is prepared and launched.

   The preprocessing includes checking whether the input message size is an integer multiple of 16 bytes. If not, CPU software adds padding and then XORs the input message with either K1 or K2.
9. The result of this operation is fed into the AES-CBC encryption process and the descriptor is prepared and launched.

   The output context of this AES-CBC operation is CMAC.

For information on the security principles and the details of the CMAC algorithm, refer to Section 4, "References."

# 2 CMAC Software

The CMAC software discussed here is an add-on to the SEC2 kernel (Linux 2.6.11) downloadable module. For details on this driver software, refer to the *SEC 2.0 Reference Device Driver User's Guide* (SEC2SWUG) and the corresponding driver software, both of which are available from the MPC8555E page at the Freescale web site listed on the back cover of this document.

To run the software as discussed in this application note, t have a MPC8555/41 CDS platform and a board-support package (BSP) for that target platform. Because it is so easily available, the Freescale LTIB-based BSP is used for building the bootloader, kernel, and SEC2 driver and modules. It is beyond the scope of this document to describe the BSP procedure for the MPC8555CDS platform. If you are interested in a LTIB-based BSP for running the SEC2 driver and modules on the MPC8555CDS platform, refer to the Freescale application note entitled *MPC8555E Security Quick Start Guide* (AN3075).

## 2.1 Configuring and Building the CMAC Module

The downloadable CMAC kernel has several tunable parameters that are explained in this section. The following line prints a debug message, if it is defined:

```
//#define DEBUG_CMAC
```

A CMAC can be generated for AES ciphers with key lengths of either 128, 192, or 256 bits. Define one of these macros before compiling and running the module on the target MPC8555CDS platform. There is no guard against the user error of defining more than one of these macros.

```
//#define AES_128
//#define AES_192
#define AES_256
```

For each selection of cipher block (AES_128, AES_192, or AES_256) four tests (TEST1, TEST2, TEST3 and TEST4) can be run. Define one of the four TEST*n* macros. For each of these macros, a NIST published known input message is taken. Then the SEC2 performs the cascaded AES crypto operations, generates

the CMAC, and compares it with the NIST published values. Refer to the `Updated_CMAC_Examples.pdf` document published by NIST. There is no guard against the user error of defining more than one of these marcos.

```
//#define TEST1
//#define TEST2
//#define TEST3
#define TEST4
```

To compile the CMAC software successfully, have a working kernel source tree (preferably 2.6.11) and the SEC2 driver code to place on the kernel tree. For an example, see the Freescale application note entitled *MPC8555E Security Quick Start Guide* (AN3075), which describes the LTIB approach of building the kernel and SEC2 drivers/modules.

Assuming that the SEC2 driver is up and running in a Linux environment, you must add the CMAC features to the SEC2 module by adding the following files.

- `testCmac.c : $KERNEL_SRC/drivers/sec2x-test/`
- `testAll.c : $KERNEL_SRC/drivers/sec2x-test/`
- `Makefile: $KERNEL_SRC/drivers/sec2x-test/`

Next, compile the kernel and build the modules. In the `sec2x-test` directory, an sec2drvTest.ko module is created. For simplicity, all other traditional SEC2 testing features are commented out in the `testAll.c` file and only the CMAC test procedure is enabled. When the module runs, notice that the CMAC results are displayed on the console. An example test result is shown in next section.

## 2.2 Example Run

LTIB, which is based on the Linux 2.6 kernel and rootfilesystem, is deployed to the MPC8555CDS target platform as described in the *MPC8555E Security Quick Start Guide* (AN3075). From the Linux prompt, install the module. Example 1 shows how the console display lists the key, message, subkeys cipher block chosen, and the CMAC derived.

**Example 1. Running the CMAC Module**

```
/lib/modules/2.6.11/kernel/drivers/sec2x-test # insmod sec2drvTest.ko

*** Test CMAC subkey generation ***

==========================================

Key:
d107d924   60 3d eb 10 15 ca 71 be   2b 73 ae f0 85 7d 77 81   `=....q.+s...}w.
d107d934   1f 35 2c 07 3b 61 08 d7   2d 98 10 a3 09 14 df f4   .5,.;a..-.......

Message for subkey generation:
d107d984   00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ................

K1:
cec17c90   ca d1 ed 03 29 9e ed ac   2e 9a 99 80 86 21 50 2f   ....).........!P/

K2:
cec17ca0   95 a3 da 06 53 3d db 58   5d 35 33 01 0c 42 a0 d9   ....S=.X]53..B..

*** Test CMAC generation  ***
```

```
====================================
Block Cipher chosen : AES 256
TEST4 : Message Length 512 bits (64 Bytes)

Message for CMAC generation:
d107d944   6b c1 be e2 2e 40 9f 96   e9 3d 7e 11 73 93 17 2a   k....@...=~.s..*
d107d954   ae 2d 8a 57 1e 03 ac 9c   9e b7 6f ac 45 af 8e 51   .-.W......o.E..Q
d107d964   30 c8 1c 46 a3 5c e4 11   e5 fb c1 19 1a 0a 52 ef   0..F.\........R.
d107d974   f6 9f 24 45 df 4f 9b 17   ad 2b 41 7b e6 6c 37 10   ..$E.O...+A{.l7.

CMAC:
cec17cf0   e1 99 21 90 54 9f 6e d5   69 6a 2c 05 6c 31 54 10   ..!.T.n.ij,.l1T.
testAll(): All 2 Tests Passed
/lib/modules/2.6.11/kernel/drivers/sec2x-test #
```

## 2.3 Customizing the Software

The message for which the CMAC module must be generated is defined as a static array. Both the input and output arrays are padded with 0s to ensure that their size is a multiple of 16 bytes. This padding is just a place holder for actual padded data to be constructed by software, thus preventing the need for the SEC scatter gather mechanism. A message of arbitrary length could be passed to the CMAC software instead of a static array. If this message length is not a multiple of 16 bytes, the software must create the padding bits in a memory region that is probably not contiguous to the message supplied to the CMAC module. In this case, the SEC driver scatter gather capability can be used to fetch the non-contiguous padding.

Choosing static arrays simplifies data handling. The underlying cryptographic operation remains the same, thus validating that the CMAC software implements the algorithm accurately, which is the sole objective of this software. With minimal effort, you can add custom features to this CMAC software.

# 3 Conclusions

This document enables developers to accelerate CMAC generation using PowerQUICC devices with SEC 2.x functionality. Some customization may be needed to tailor the CMAC module to a specific environment. These changes should be minimal if the Freescale Linux-based security driver is used. The CMAC module is implemented using the SEC2 driver API.

# 4 References

The following reference documents for this application note are available either at the Freescale web site listed on the back cover of this document or at the NIST web site:

- *MPC8555E Security Quick Start Guide* (AN3075)
- *SEC 2.0 Reference Device Driver User's Guide* (SEC2SWUG)
- Updated CMAC examples:
  `http://csrc.nist.gov/publications/nistpubs/800-38B/Updated_CMAC_Examples.pdf`
- *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*
  `http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf`
- *SEC 2.0 Descriptor Programmer's Guide* (AN2755)

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

### *How to Reach Us:*

**Home Page:**
www.freescale.com

**email:**
support@freescale.com

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
1-800-521-6274
480-768-2130
support@freescale.com

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064, Japan
0120 191014
+81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

**For Literature Requests Only:**
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com



Document Number: AN3085
Rev. 0
06/2006

freescale™
semiconductor