AN708

SETTING MANUFACTURING CERTIFICATES AND INSTALLATION CODES FOR THE EM35X SOC AND COPROCESSOR PLATFORMS

(Formerly document 120-5058-000)

This application note gives an overview of the use of install codes and certificates in a Smart Energy network. Besides, it describes the components of a Smart Energy certificate and the differences between test certificates and production certificates. It also explains (with the help of examples) how one can use Ember tools (Debug Adapter utilities) to program, verify and erase install codes and certificates from Ember chips.

New in This Revision

Smart Energy 1.2 Certificate support.

Contents

1 Smart Energy Certificate Overview

The ZigBee Smart Energy specification uses public/private key technology to authenticate a device joining a Smart Energy network and provides a means to securely establish encryption keys for future transactions. The Smart Energy specification uses Elliptical Curve Cryptography (ECC) for cryptographic authentication and key generation.

The Smart Energy profile uses ECC with the key establishment cluster to derive a link key. It also uses ECC for creating digital signatures for software upgrade images sent through the ZigBee Over-the-air bootload cluster (OTA).

Currently Certicom (www.certicom.com) provides both the certificates and the ECC technology for use in ZigBee Smart Energy networks.

Smart Energy 1.0 used an ECC curve 163k1 with a 48-byte certificate. Smart Energy 1.2 introduced use of the ECC curve 283k1 with a 74-byte certificate. The certificates are separate and unique and are not interoperable. However when both are present on the same device they must contain the same IEEE address. Devices may have one or both sets of security data installed. The rules for what devices must have both certificates or what devices only need one certificate is governed by the Smart Energy specification.

2 Security Components

When referring to Smart Energy Certificates, there are actually three components that make up the ECC security data contained within a node:

- Certificate
- Static Private Key
- CA Public Key

2.1 Certificate

The first component of the ECC security data is the certificate. This is the device's IEEE address and static public key, which are signed using the CA's private key. The certificate is associated with a particular IEEE address. **As a result, the IEEE address of the device must match the value contained in the certificate**.

Although this is part of the security data, it is not secret, and does not need to be handled in the same fashion as the private key. It is transmitted over the air during key establishment.

2.1.1 Smart Energy 1.0 Certificate

Smart Energy 1.0 uses a 48-byte certificate. Table 1 summarizes the four different fields contained within the certificate body.

Table 1: Smart Energy 1.0 Certificate Body Fields

Field name	Length (bytes)	Description
Public Key Reconstruction Data	22	Device's public key signed by the CA's private key.
Subject	8	Contains the IEEE address associated with the certificate, in big endian format.
Issuer	8	Identity of the CA that issued the certificate.
Attributes	10	An extra set of data associated with the device whose authenticity is guaranteed by the CA. It is not currently used by ZigBee Smart Energy.

2.1.2 Smart Energy 1.2 Certificate

Smart Energy 1.2 uses a 74-byte certificate. Table 2 summarizes the four different fields contained within the certificate body.

Table 2: Smart Energy 1.2 Certificate Body Fields

Name	Bytes	Description
Type	1	Type of certificate = 0, implicit no extensions
SerialNo	8	Serial Number of the certificate
Curve	1	Curve identifier (sect283k1 is 13 or byte value 0x0D)
Hash	1	Hash identifier (AES-MMO is byte value 0x08)
Issuer	8	8 byte identifier, 64-bit IEEE 802.15.4 address
ValidFrom	5	40-bit Unix time from which the certificate is valid
ValidTo	4	32-bit # of seconds from the ValidFrom time for which the certificate is considered valid (0xFFFFFFFF = infinite)
SubjectID	8	8 byte identifier, 64-bit IEEE 802.15.4 address
KeyUsage	1	Bit flag indicating key usage (0x88 = digital signature or key agreement allowed)
PublicKey	37	37-byte compressed public key value from which the public key of the Subject is reconstructed.

2.2 Static Private Key

The second component of the ECC security data is the static private key. This is a secret piece of data that should be carefully protected. Silicon Labs recommends that during the manufacturing process only those computers that need to know this data to program the chip have access to it.

A Smart Energy 1.0 private key is 21-bytes in length.

A Smart Energy 1.2 private key is 36-bytes in length.

2.2.1 CA public key

This is the CA's public key that corresponds to the secret private key held by Certicom. It is used to authenticate certificates received by remote devices and validate the keys derived using the Smart Energy Key Establishment Cluster. While it is not transmitted over the air, it is also public information and does not need to be kept secret.

A Smart Energy 1.0 CA Public Key is 22-bytes in length.

A Smart Energy 1.2 CA Public Key is 37-bytes in length.

3 Certificate Authorities

Two Certificate Authorities (CA) are provided by Certicom for use in Smart Energy networks:

- A Test CA that is intended only for use in internal development environments.
- The Production CA that is intended for use in real Smart Energy deployments.

Note: Certificates from the Test CA and the Production CA are **not compatible**.

There are two CA's because developers and installers have different security needs. Developers need to be able to test and debug devices where they will have access to the private keys. Their interests lie in having access to internal data in order to accomplish their goal of writing software. On the other hand, installers want reasonable assurances that a device will not have compromised security data. Their goal is to have a secure installation. Certicom can meet both needs by providing different security for each CA.

3.1 Test CA

Certicom provides a Test CA for quickly and easily generating certificates for use in development units wishing to use ECC. The following link can be used to obtain certificates from the Test CA: http://www.certicom.com/index.php/gencertregister.

Silicon Labs provides two test certificates in their Smart Energy sample application, bound to IEEE addresses 0022080000000001 and 0022080000000002. These may be used in combination with the Test CA to perform key establishment. They can be found in the Ember ECC distribution ZIP file as certicom-test-cert-1.txt and certicom-test-cert-2.txt. The following is the CA Public Key for the Smart Energy 1.0 Test CA:

```
0200FDE8A7F3D1084224962A4E7C54E69AC3F04DA6B8
```

3.2 Production CA

The Production CA is the official CA that is used by all deployed devices. All manufactured devices should contain the Production CA's public key, and a certificate and private key associated with the Production CA.

The following is the Smart Energy 1.0 Production CA's Public Key, which should be installed on production devices:

```
0202264C5E4CBFA186A6B925B966B5B3A4D7A390344E
```

Production CAs are obtained through a different mechanism from Test CAs. Since manufacturing will involves hundreds or potentially thousands of devices, a different process is used to obtain bulk certificates for a block of IEEE addresses. In addition, security of the transmission of the private keys is much more important and thus is handled differently from the Test CA.

It is recommended that a manufacturer obtain their own block of IEEE addresses for production units instead of using the internal Ember one programmed on the chip. Certificates identify a Smart Energy device and its vendor and an Ember chip may be used in many Smart Energy deployments. In addition, it is easier to issue certificates for blocks of IEEE addresses. Silicon Labs cannot guarantee a set of known IEEE addresses with a particular set of chips, so it is recommended that the manufacturer install their own to make it easier to manage them.

Contact Certicom for more information about obtaining production certificates.

4 Certificate Validation

It is important to note that ECC uses a technology called implicit certificates. Traditional SSL certificates used on the Internet contain the unencrypted public key of the device along with a separate signature field that ensures the authenticity of the data. However, with implicit certificates, the two pieces of data are combined together to shrink the size of the certificate for use in embedded devices. This is known as *Public Key Reconstruction data*.

As a result of the combination, it is not possible to verify an implicit certificate without using the certificate as part of the Key Establishment Cluster. Only by performing Key Establishment to derive a link key can a device determine if its certificate is valid.

5 Matching the Components

It is important to note that the four pieces of security data installed for Smart Energy **must all match up correctly**. The installed certificate must be the correct one for the private key that is also installed. The IEEE address of the device must match the certificate that is installed. Lastly, the certificate must have been issued by the same CA whose public key is installed along with it.

If any of the components is mismatched, the device will never be able to successfully perform key establishment to authenticate itself on a Smart Energy network.

5.1 Manufacturing Test

Silicon Labs recommends that manufacturers validate their process by testing the installed certificate in their first batch of devices. A full test involves successfully joining a Smart Energy network and using the key establishment cluster to derive a link key. The devices in the test should be using Production Certificates issued by the Production CA.

The EM35x SoC and coprocessor platforms can all store application-specific manufacturing data into an area of flash that can be used to store unique device information. This information is read-only at run-time and must be programmed via the Debug Adapter.

Note: Hard-coding a device's unique information in source code does not work when using the same application image on multiple devices. The preferred alternative is to have the software read in the device's unique information from the manufacturing area of flash.

6 Working with Certificates on the EM35x

6.1 Smart Energy 1.0 and 1.2 Certificates

Smart Energy 1.0 used an ECC 163k1 curve with a 48-byte certificate. Smart Energy 1.2 added support for ECC 283k1 curve with a 74-byte certificate. The certificates and the libraries are not interoperable. The ZigBee specification requirements describe what devices must support both certificates and both sets of libraries.

6.2 Overriding the Default EUI64

By default, an Ember EUI64 address is pre-programmed into all EM3xx chips. Customers may override this by writing their own EUI64 into the Custom EUI64 area of the manufacturing flash.

When programming certificates and installation codes the **em3xx_load.exe** tool allows you to override the Ember EUI64 (see document UG107, *EM35x Utilities Guide*, for more information). When a custom EUI64 is already present, the certificate or installation code **must use an EUI64 that matches the custom one**.

If you do not want to override the EUI64, then the certificate and installation codes must use Ember EUI64s, and those **EUI64s must match what is already programmed on the chip**. You cannot override the EUI64 of the device with another Ember EUI64.

The reason these requirements are in place is to prevent certificates or installation codes from being programmed on the wrong device. Both of these items are bound to the EUI64 of the device and therefore a mismatch will cause problems when deploying an end product.

To program a device with a certificate or installation code, the device must be connected via the debug cable to an Ember Debug Adapter. The tools provided with the Debug Adapter (ISA3) Utilities installer provide the basic elements needed to do this. The tool used is em3xx_load.exe.

Verifying the Software Version

To verify that you have the correct version of the tools, open a command prompt, and change to the directory where the tools are installed. The default for the Ember installation is **C:\Program Files\Ember\ISA3 Utilities\bin\**.

To verify the software version, run the tool without any arguments using this command:

```
$ em3xx_load.exe
```

In order to support Smart Energy 1.0 certificates, you need to be running at least version **1.0b13**:

```
$ em3xx_load.exe
em3xx_load (Version 1.0b13.1256666220)
```

In order to support Smart Energy 1.2 certificates, you need to be running at least version **4.1b04**:

```
$ em3xx_load.exe

em3xx_load version 4.1b04
NAME
    em3xx_load - Programming utilities for EM3XX series chips

SYNOPSIS
    em3xx_load [TARGET OPTION] [MODIFIERS] COMMAND [COMMAND ARGUMENTS]
```

7 Programming the Certificates into a Device

7.1 Checking the Node Information

Prior to modifying the certificate it is best to verify there is connectivity with the device to be programmed, and what information is currently stored on the node. To do this, execute the following command:

```
$ em3xx_load.exe  --cibtokensprint
```

You should see the following output:

```
$ em3xx_load.exe --cibtokensprint

em3xx_load version 4.1b04
Connecting to ISA via IP address 10.4.176.51
DLL version 1.1.28, compiled Sep 25 2013 13:55:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting EM3588


'General' token group
TOKEN_MFG_CIB_OBS              [16 byte array ] : A55AFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_CUSTOM_VERSION       [16-bit integer] : 0xFFFF
TOKEN_MFG_CUSTOM_EUI_64        [ 8 byte array ] : FFFFFFFFFFFFFFFF
TOKEN_MFG_STRING               [16 byte string] : "" (0 of 16 chars)
                                                  FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_BOARD_NAME           [16 byte string] : "" (0 of 16 chars)
                                                  FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_MANUF_ID             [16-bit integer] : 0xFFFF
TOKEN_MFG_PHY_CONFIG           [16-bit integer] : 0xFF26
TOKEN_MFG_BOOTLOAD_AES_KEY     [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
```

```
TOKEN_MFG_EZSP_STORAGE          [ 8 byte array ] : FFFFFFFFFFFFFFFF
TOKEN_MFG_OSC24M_BIAS_TRIM      [16-bit integer] : 0xFFFF
TOKEN_MFG_SYNTH_FREQ_OFFSET     [16-bit integer] : 0xFFFF
TOKEN_MFG_OSC24M_SETTLE_DELAY   [16-bit integer] : 0xFFFF
TOKEN_MFG_SECURITY_CONFIG       [16-bit integer] : 0xFFFF
TOKEN_MFG_CCA_THRESHOLD         [16-bit integer] : 0xFFFF
TOKEN_MFG_SECURE_BOOTLOADER_KEY [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF

'Smart Energy CBKE (TOKEN_MFG_CBKE_DATA)' token group
Device Implicit Cert [48 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
CA Public Key        [22 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFF
Device Private Key   [21 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFF
CBKE Flags           [ 1 byte array ] : FF

'Smart Energy Install Code (TOKEN_MFG_INSTALLATION_CODE)' token group
Install Code Flags [ 2 byte array ] : FFFF
Install Code       [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
CRC                [16-bit integer] : 0xFFFF

'Smart Energy 1.2 CBKE (TOKEN_MFG_CBKE_283K1_DATA)' token group
Device Implicit Cert (283k1) [74 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFF
CA Public Key (283k1)        [37 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFF
Device Private Key (283k1)   [36 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFF
CBKE FLAGS (283k1)           [ 1 byte array ] : FF

DONE
```

Note: The pre-programmed Ember EUI64 is not shown in this output as it is part of the Silicon Labs-programmed manufacturing data rather than the customer-programmed manufacturing data. To obtain the Ember EUI64 of an EM3xx device, use the following command (shown for a device whose Debug Adapter (ISA3) is attached to the PC via USB at Device ID 0):

```
em3xx_load --read @080407A2-080407A9
```

This reads out the Ember EUI64 in least-significant byte (LSB) notation. For example, the command above produces the following output (indicating an Ember EUI64 value of 0x000D6F000092E047):

```
em3xx_load (Version 1.0b13.1256666220)

Connecting to ISA via USB Device 0
DLL version 1.1.2, compiled Oct 09 2009 14:09:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting EM357
```

```
Reset Chip
Getting memory from 0x080407A2 to 0x080407A9

{address:  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F}
080407A0:  .  . 47 E0 92 00 00 6F 0D 00  .  .  .  .  .  .

Run (by toggling nRESET)
DONE
```

If the custom EUI64 (IEEE address) is set to all F’s, then the Ember EUI64 address is used. If this value is not all F’s, then it used instead of the Ember EUI64.

If the data within the Smart Energy token group (certificate, private key, root CA public key, metadata) is set to all F’s, then the data is not valid and will not be used by the software.

7.2 Format of the Certificate File

Note: If programming both a Smart Energy 1.0 and a Smart Energy 1.2 certificate the two files may be combined into one.

7.2.1 Smart Energy 1.0

To program a Smart Energy 1.0 certificate into the tokens, it is necessary to create a simple text file with the security information in it. The following is the format of that file expected by the **em3xx_load.exe** tool for Smart Energy 1.0 certificates.

```
CA Public Key: <hex-string>
Device Implicit Cert: <hex-string>
Device Private Key: <hex-string>
```

The following is a sample certificate file issued by the **Test CA**, for a device with an IEEE of 0000000000000001 (big endian).

```
CA Public Key: 0200fde8a7f3d1084224962a4e7c54e69ac3f04da6b8
Device Implicit Cert:
03045fdfc8d85ffb8b3993cb72ddcaa55f00b3e87d6d000000000000000154455354534543410109000600
00000000000
Device Private Key: 00b8a900fcadebabbfa383b540fce9ed438395eaa7
```

Note: The code line starting with ‘Device Implicit Cert:’ and its value appears in the CA Public Key on a single line, not three lines as represented above.

7.2.2 Smart Energy 1.2

To program a Smart Energy 1.2 certificate into the tokens, it is necessary to create a simple text file with the security information in it. The following is the format of that file expected by the **em3xx_load.exe** tool for Smart Energy 1.2 certificates.

```
CA Public Key (283k1): <hex-string>
Device Implicit Cert (283k1): <hex-string>
Device Private Key (283k1): <hex-string>
```

The following is a sample certificate file issued by the **Test CA**, for a device with an IEEE of 0000000000000001 (big endian).

```
CA Public Key (283k1):
      0207a445022d9f39f49bdc38380026a27a9e0a1799313ab28c5c1a1c6b605154db1dff6752
```

```
Device Implicit Cert (283k1):
      00602a3c32e55a8acf0d0811121314151617180053 20c3a6ffffffff0000000000000001080201d
      4d1887cd727e195300c9a11c4b6cd82d37f36381bc3707a05b8f01b73d236d3a59ee9
```

```
Device Private Key (283k1):
      000071ab965eeedfa9ad08370fed7dc7b1e7d0940c9f3917a23b2cdb16afd2402ade31a5
```

Note: The lines with hex data on them are on the same line as the text strings and not on separate lines as shown above.

7.3 Writing the Certificate into the Manufacturing Area

To write the certificate into the manufacturing area, execute the following command:

```
$ em3xx_load.exe --cibtokenspatch sample_cert.txt
```

You should see the following output (Note: This assumes you are programming two certificates):

```
$ em3xx_load.exe --cibtokenspatch sample_cert.txt
em3xx_load version 4.1b04
Connecting to ISA via IP address 10.4.176.51
DLL version 1.1.28, compiled Sep 25 2013 13:55:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting EM3588
Reset Chip

Writing to address 0x0808087E for token 'Device Implicit Cert'
Writing to address 0x080808AE for token 'CA Public Key'
Writing to address 0x080808C4 for token 'Device Private Key'
Writing to address 0x080808D9 for token 'CBKE Flags'
Writing to address 0x0808090E for token 'Device Implicit Cert (283k1)'
Writing to address 0x08080958 for token 'CA Public Key (283k1)'
Writing to address 0x0808097D for token 'Device Private Key (283k1)'
Writing to address 0x080809A1 for token 'CBKE FLAGS (283k1)'

NOTE: Writing Custom EUI64 '0100000000000000' to match certificate.  Address
0x08080812.

Create image file
Install RAM image
Verify RAM image
Install Flash image
Verify Flash image
Run (by toggling nRESET)
DONE
```

In addition to writing the certificate, this will update the custom IEEE address so that it matches the data in the certificate. This ensures that there is parity between the data in the certificate and the IEEE address used by the local device for its identity on the network.

7.4 Verifying the Stored Certificate

After writing the certificate, it is best to verify the information by executing em3xx_load again.

```
$ ./em3xx_load.exe  --cibtokensprint
```

You should see the following output:

```
$ ./em3xx_load.exe  --cibtokensprint
em3xx_load version 4.1b04
Connecting to ISA via IP address 10.4.176.51
DLL version 1.1.28, compiled Sep 25 2013 13:55:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting EM3588


'General' token group
TOKEN_MFG_CIB_OBS                [16 byte array ] : A55AFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_CUSTOM_VERSION         [16-bit integer] : 0xFFFF
TOKEN_MFG_CUSTOM_EUI_64          [ 8 byte array ] : 0100000000000000
TOKEN_MFG_STRING                 [16 byte string] : "" (0 of 16 chars)
                                                    FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_BOARD_NAME             [16 byte string] : "" (0 of 16 chars)
                                                    FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_MANUF_ID               [16-bit integer] : 0xFFFF
TOKEN_MFG_PHY_CONFIG             [16-bit integer] : 0xFF26
TOKEN_MFG_BOOTLOAD_AES_KEY       [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_EZSP_STORAGE           [ 8 byte array ] : FFFFFFFFFFFFFFFF
TOKEN_MFG_OSC24M_BIAS_TRIM       [16-bit integer] : 0xFFFF
TOKEN_MFG_SYNTH_FREQ_OFFSET      [16-bit integer] : 0xFFFF
TOKEN_MFG_OSC24M_SETTLE_DELAY    [16-bit integer] : 0xFFFF
TOKEN_MFG_SECURITY_CONFIG        [16-bit integer] : 0xFFFF
TOKEN_MFG_CCA_THRESHOLD          [16-bit integer] : 0xFFFF
TOKEN_MFG_SECURE_BOOTLOADER_KEY  [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF

'Smart Energy CBKE (TOKEN_MFG_CBKE_DATA)' token group
Device Implicit Cert [48 byte array ] : 03045FDFC8D85FFB 8B3993CB72DDCAA5
                                        5F00B3E87D6D0000 0000000000015445
                                        5354534543410109 0006000000000000
CA Public Key        [22 byte array ] : 0200FDE8A7F3D108 4224962A4E7C54E6
                                        9AC3F04DA6B8
Device Private Key   [21 byte array ] : 00B8A900FCADEBAB BFA383B540FCE9ED
                                        438395EAA7
CBKE Flags           [ 1 byte array ] : 00

'Smart Energy Install Code (TOKEN_MFG_INSTALLATION_CODE)' token group
Install Code Flags [ 2 byte array ] : FFFF
Install Code       [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
CRC                [16-bit integer] : 0xFFFF

'Smart Energy 1.2 CBKE (TOKEN_MFG_CBKE_283K1_DATA)' token group
Device Implicit Cert (283k1) [74 byte array ] : 00602A3C32E55A8A CF0D081112131415
                                                161718005320C3A6 FFFFFFFF00000000
                                                00000001080201D4 D1887CD727E19530
                                                0C9A11C4B6CD82D3 7F36381BC3707A05
                                                B8F01B73D236D3A5 9EE9
CA Public Key (283k1)        [37 byte array ] : 0207A445022D9F39 F49BDC38380026A2
```

```
                                                7A9E0A1799313AB2 8C5C1A1C6B605154
                                                DB1DFF6752
Device Private Key (283k1)   [36 byte array ] : 000071AB965EEEDF A9AD08370FED7DC7
                                                B1E7D0940C9F3917 A23B2CDB16AFD240
                                                2ADE31A5
CBKE FLAGS (283k1)           [ 1 byte array ] : 00


DONE
```

8 Erasing the Certificate

If the wrong device is programmed with a certificate, or it is necessary to remove this security data from the device, complete the following procedures.

To erase token data create an erase file listing the token name using **!ERASE!** as the token data. For example to erase an ECC 163k1 certificate:

```
CA Public Key: !ERASE!
Device Implicit Cert: !ERASE!
Device Private Key: !ERASE!
```

To erase an ECC 283k1 certificate:

```
CA Public Key (283k1): !ERASE!
Device Implicit Cert (283k1): !ERASE!
Device Private Key (283k1): !ERASE!
```

Note: You can erase both certificates simultaneously by putting the above lines in the same file.

To erase a sample certificate, execute this command:

```
$ em3xx_load.exe --cibtokenspatch sample_erase_cert.txt
```

You should see the following output:

```
$ em3xx_load.exe --cibtokenspatch sample_erase_cert.txt
em3xx_load (Version 1.0b13.1256666220)

Connecting to ISA via USB Device 0
DLL version 1.1.2, compiled Oct 09 2009 14:09:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting EM3577
Reset Chip

Writing to address 0x0804087E for token 'Device Implicit Cert'
Writing to address 0x080408AE for token 'CA Public Key'
Writing to address 0x080408C4 for token 'Device Private Key'
Writing to address 0x080408D9 for token 'CBKE Flags'

NOTE: Writing Custom EUI64 'FFFFFFFFFFFFFFFF' to match certificate.  Address
0x08040812.

Create image file
Install RAM image
Verify RAM image
```

```
Install Flash image
Verify Flash image
Run (by toggling nRESET)
DONE
```

9 Smart Energy Installation Code Overview

Smart Energy Installation codes are provided as a means for a device to join a ZigBee network in a reasonably secure fashion. The installation code itself is a random value printed on the joining device, which is used to encrypt the initial message exchange between it and the Energy Services Interface (ESI).

The installation code can be thought of as similar to the PIN code on Bluetooth devices when two devices are paired. The PIN code is provided as an authorization code for the parent device so that the joining device knows it is receiving information securely, such as when a handsfree headset is paired to a cellular phone.

The installation code for a Smart Energy device is printed on the outside of the device, and provided via an out-of-band mechanism to the utility along with the IEEE of the device. The utility then securely transports that information to the ESI.

The ESI and the joining device use the installation code as a shared key to establish an initial bond of trust allowing the new device to join the Smart Energy network.

10 Security Use

An installation code is used to create a preconfigured, link key. The installation code is transformed into a link key by use on an AES hash. For more information and sample code, consult the ZigBee Smart Energy Profile Specification (ZigBee Document Number 07-5356). Public access for ZigBee Technical Documents can be found at: http://www.zigbee.org/Specifications/ZigBee/download.aspx.

The derived ZigBee link will be known only by the ESI and the joining device. The ESI uses that key to securely transport the ZigBee network key to the device. Once the device has the network key, it can communicate at the network layer to the Smart Energy network. It has the ability to perform service discovery and begin the application’s initialization process.

The installation code, while not exactly a secret, cannot be easily guessed by a malicious device that hears the initial exchange between the joining device and ESI. Without knowledge of the installation code and thus the key, it cannot decrypt the messages.

The initial link key derived from the installation code does not have full access privileges on a Smart Energy network. Attempts to use it for Smart Energy messaging are not allowed and will be ignored by other Smart Energy devices. Shortly after joining a network, a device must use the Key Establishment cluster to establish a new link key with the ESI. Only when key establishment completes successfully will a device have full privileges on the network and be able send and receive certain Smart Energy messages.

11 Installation Code Format

The installation code is made up of a 6-, 8-, 12-, or 16-byte random, hexadecimal number with a 2-byte CRC appended to the end. As far as the user is concerned, the CRC is part of the installation code and they do not need to know that it is there or why. Therefore, from the user’s point of view, the length of the install code is 8, 10, 14, or 18 bytes.

The choice of 6-, 8-, 12-, or 16-byte length for the installation code is up to the device vendor. Manufacturing and managing the list of installation codes will play a part in choosing the size, security, and user experience in installing the device. A larger installation code size will mean less of a chance of an attacker “guessing” the installation code and eavesdropping on the initial join. However smaller installation code is much easier for a user to read off the device during installation.

NOTE: It is recommended to only use a16-byte installation code. While this may be more difficult to enter, it provides sufficient strength against an attacker from guessing the install code and gaining unauthorized access to network or device.

12 Installation Code CRC

The installation code CRC is mechanism used to verify the integrity of an installation code when it is transmitted via an out-of-band mechanism to the utility. This transport mechanism involves human interaction in some way. As a result, the CRC was designed as a way to verify that an installation code is valid and was not mistakenly changed during transport.

The Smart Energy installation model enables users to install a device themselves. Users simply read the installation code on the back of the device and enter it into a webpage or provide it over the phone to a utility service. Because the number is a hexadecimal value, it is easy to transpose digits or read the wrong value.

12.1 Validation

The ZigBee Smart Energy Profile Specification expects that the utility will perform basic checking of the installation code for validity. The utility calculates the CRC over all bytes in the installation code except the final two. It then compares the final two bytes of the installation code with the calculated CRC to see if they match. If they do not match, the user entering the installation code can be informed immediately that it does not look valid. The user should then double check the value.

The ZigBee Smart Energy Profile Specification does not require the ESI to validate the installation code. The ESI expects to receive a pre-configured link key along with the IEEE address of the new joining device. It does not need to have any knowledge about how that key was derived. It is up to the particular utility how they wish to manage and transport the link key to the ESI.

For details on how the CRC is calculated, including sample code, consult the ZigBee Smart Energy Profile Specification (ZigBee Document Number 07-5356). Public document access can be found here: http://www.zigbee.org/Specifications/ZigBee/download.aspx.

12.2 Generation

Silicon Labs recommends that the installation code be a random number. This reduces the chances of an attacker guessing the installation code and compromising the initial join procedure. The installation code should not be based on the manufacturing process, such as tied to the IEEE address or sequential numbering based on the manufacturing lot.

If that were the case, an attacker with knowledge about the type of device being joined would have a known range of installation codes it could try to compromise the network and clone the device's identity. An installation code does not have to be unique across all Smart Energy devices for all manufacturers.

12.3 Labels

The device's installation code should be printed on a label on the outside of the device along with its IEEE address. Both elements should be identified with text indicating what they are. The installation code should not be printed on the outside of the box because that makes it easier for an attacker to gain knowledge of the installation code and potentially compromise the device. It is recommended that the installation code be printed in 2-byte blocks (for example, 83FE D340 7A93 9723 A5C6 39B2 6916 D505 C3B5).

Note: The CRC should be appended to the installation code in little endian format on the label.

12.3.1 Example

The following is a 18-byte installation code label (16-byte random code with a 2-byte CRC):

```
83FE D340 7A93 9723 A5C6 39B2 6916 D505 C3B5 C3B5
```

The random number portion of the code is the first 16 sequential bytes. The calculated CRC value is 0xB5C3, but it is appended in little-endian format.

13 Programming the Installation Code on an EM35x

13.1 Checking the Installation Code on a Device

Prior to modifying the certificate, it is best to verify there is connectivity with the device to be programmed, and what information is currently stored on the node. To do this, execute the following command:

```
$ ./em3xx_load.exe  --cibtokensprint
```

You should see the following output:

```
$ ./em3xx_load.exe  --cibtokensprint
em3xx_load version 4.1b04
Connecting to ISA via IP address 10.4.176.51
DLL version 1.1.28, compiled Sep 25 2013 13:55:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting EM3588


'General' token group
TOKEN_MFG_CIB_OBS                [16 byte array ] : A55AFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_CUSTOM_VERSION         [16-bit integer] : 0xFFFF
TOKEN_MFG_CUSTOM_EUI_64          [ 8 byte array ] : FFFFFFFFFFFFFFFF
TOKEN_MFG_STRING                 [16 byte string] : "" (0 of 16 chars)
                                                    FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_BOARD_NAME             [16 byte string] : "" (0 of 16 chars)
                                                    FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_MANUF_ID               [16-bit integer] : 0xFFFF
TOKEN_MFG_PHY_CONFIG             [16-bit integer] : 0xFF26
TOKEN_MFG_BOOTLOAD_AES_KEY       [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_EZSP_STORAGE           [ 8 byte array ] : FFFFFFFFFFFFFFFF
TOKEN_MFG_OSC24M_BIAS_TRIM       [16-bit integer] : 0xFFFF
TOKEN_MFG_SYNTH_FREQ_OFFSET      [16-bit integer] : 0xFFFF
TOKEN_MFG_OSC24M_SETTLE_DELAY    [16-bit integer] : 0xFFFF
TOKEN_MFG_SECURITY_CONFIG        [16-bit integer] : 0xFFFF
TOKEN_MFG_CCA_THRESHOLD          [16-bit integer] : 0xFFFF
TOKEN_MFG_SECURE_BOOTLOADER_KEY [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF

'Smart Energy CBKE (TOKEN_MFG_CBKE_DATA)' token group
Device Implicit Cert [48 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
CA Public Key        [22 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFF
Device Private Key   [21 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFF
CBKE Flags           [ 1 byte array ] : FF
```

```
'Smart Energy Install Code (TOKEN_MFG_INSTALLATION_CODE)' token group
Install Code Flags [ 2 byte array ] : FFFF
Install Code       [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
CRC                [16-bit integer] : 0xFFFF

'Smart Energy 1.2 CBKE (TOKEN_MFG_CBKE_283K1_DATA)' token group
Device Implicit Cert (283k1) [74 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFF
CA Public Key (283k1)        [37 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFF
Device Private Key (283k1)   [36 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFF
CBKE FLAGS (283k1)           [ 1 byte array ] : FF


DONE
```

The data stored at TOKEN_MFG_CUSTOM_EUI_64 is the custom value for the IEEE address in little-endian format. If this value is set to all F's, then the value for the internal IEEE address is used. If this value is not all F's, then it used instead of the internal IEEE address.

The data in the last block is the installation code metadata, the installation code, and the CRC. The installation code metadata (Flags and CRC) is automatically programmed by the em3xx_load tool.

13.2 Format of the Installation Code File

To program the installation code, create a simple text file with the value of the installation code (without the CRC). This file is passed into the em3xx_load.exe tool.

The format of the file is as follows:

```
Install Code: <ascii-hex>
```

Here is a sample installation code file. The CRC for that code is 0xB5C3 and does not need to be present in the file.

```
Install Code: 83FED3407A939723A5C639B26916D505
```

Per the Smart Energy specification, the installation code may be 6, 8, 12, or 16 bytes in length (not including the two-byte CRC).

13.3 Writing the Installation Code into the Manufacturing Area

To write the installation code into the manufacturing area, execute the following command:

```
$ em3xx_load.exe --cibtokenspatch install-code-file.txt
```

You should see the following output:

```
$ em3xx_load.exe --cibtokenspatch install-code-file.txt

em3xx_load version 4.1b04
Connecting to ISA via IP address 10.4.176.51
DLL version 1.1.28, compiled Sep 25 2013 13:55:00
SerialWire interface selected
```

```
SWJCLK speed is 500kHz
Targeting EM3588
Reset Chip

NOTE:  Calculated install code CRC is 0xB5C3.
Writing to address 0x080808DA for token 'Install Code Flags'
Writing to address 0x080808DC for token 'Install Code'
Writing to address 0x080808EC for token 'CRC'

Create image file
Install RAM image
Verify RAM image
Install Flash image
Verify Flash image
Run (by toggling nRESET)
DONE
```

13.4 Verifying the Stored Installation Code

After writing the installation code, it is best to verify the information by executing **em3xx_load** again. Note the bold values.

```
$ ./em3xx_load.exe  --cibtokensprint
```

You should see the following output:

```
$ ./em3xx_load.exe  --cibtokensprint

em3xx_load version 4.1b04
Connecting to ISA via IP address 10.4.176.51
DLL version 1.1.28, compiled Sep 25 2013 13:55:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting EM3588


'General' token group
TOKEN_MFG_CIB_OBS                [16 byte array ] : A55AFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_CUSTOM_VERSION         [16-bit integer] : 0xFFFF
TOKEN_MFG_CUSTOM_EUI_64          [ 8 byte array ] : FFFFFFFFFFFFFFFF
TOKEN_MFG_STRING                 [16 byte string] : "" (0 of 16 chars)
                                                    FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_BOARD_NAME             [16 byte string] : "" (0 of 16 chars)
                                                    FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_MANUF_ID               [16-bit integer] : 0xFFFF
TOKEN_MFG_PHY_CONFIG             [16-bit integer] : 0xFF26
TOKEN_MFG_BOOTLOAD_AES_KEY       [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_EZSP_STORAGE           [ 8 byte array ] : FFFFFFFFFFFFFFFF
TOKEN_MFG_OSC24M_BIAS_TRIM       [16-bit integer] : 0xFFFF
TOKEN_MFG_SYNTH_FREQ_OFFSET      [16-bit integer] : 0xFFFF
TOKEN_MFG_OSC24M_SETTLE_DELAY    [16-bit integer] : 0xFFFF
TOKEN_MFG_SECURITY_CONFIG        [16-bit integer] : 0xFFFF
TOKEN_MFG_CCA_THRESHOLD          [16-bit integer] : 0xFFFF
TOKEN_MFG_SECURE_BOOTLOADER_KEY [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF

'Smart Energy CBKE (TOKEN_MFG_CBKE_DATA)' token group
Device Implicit Cert [48 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
```

```
                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
CA Public Key        [22 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                FFFFFFFFFFFF
Device Private Key   [21 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                FFFFFFFFFF
CBKE Flags           [ 1 byte array ] : FF

'Smart Energy Install Code (TOKEN_MFG_INSTALLATION_CODE)' token group
Install Code Flags [ 2 byte array ] : 0600
Install Code       [16 byte array ] : 83FED3407A939723 A5C639B26916D505
CRC                [16-bit integer] : 0xB5C3

'Smart Energy 1.2 CBKE (TOKEN_MFG_CBKE_283K1_DATA)' token group
Device Implicit Cert (283k1) [74 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFF
CA Public Key (283k1)        [37 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFF
Device Private Key (283k1)   [36 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                                FFFFFFFF
CBKE FLAGS (283k1)           [ 1 byte array ] : FF


DONE
```

When the installation code is less than 16 bytes (not including the CRC), the rest of the words will be all F’s. The final two bytes are the CRC *displayed* in **big endian** format. On the chip, it will be stored in the processor’s native endianness (little-endian on the 35x).

14 Erasing the Installation Code

If the wrong device is programmed with a certificate, or you want to remove this security data from the device, complete the following steps.

14.1 Erase process

To erase token data create an erase file listing the token name using **!ERASE!** as the token data. For example:

```
Install Code: !ERASE!
TOKEN_MFG_CUSTOM_EUI_64: !ERASE!
```

This is the command-line:

```
$ em3xx_load.exe --cibtokenspatch install-code-file-erase.txt
```

You should see the following output:

```
$ em3xx_load.exe --cibtokenspatch install-code-file-erase.txt

em3xx_load version 4.1b04
Connecting to ISA via IP address 10.4.176.51
DLL version 1.1.28, compiled Sep 25 2013 13:55:00
SerialWire interface selected
```

```
SWJCLK speed is 500kHz
Targeting EM3588
Reset Chip

Writing to address 0x080808DA for token 'Install Code Flags'
Writing to address 0x080808DC for token 'Install Code'
Writing to address 0x080808EC for token 'CRC'

NOTE: Writing Custom EUI64 'FFFFFFFFFFFFFFFF'.  Address 0x08080812.

Create image file
Install RAM image
Verify RAM image
Install Flash image
Verify Flash image
Run (by toggling nRESET)
DONE
```

Contact Information

Silicon Laboratories Inc.

400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032

Please visit the Silicon Labs Technical Support web page for ZigBee products:
www.silabs.com/zigbee-support and register to submit a technical support request