ember

Application Note 5070
19 March 2011
120-5070-000A

EM250

EM260

EM35x

Smart Energy ECC-Enabled Device Setup Process

This document describes how to set up a device with the security resources required to support Smart Energy (SE) security, which is based on certificate-based key establishment (CBKE) using Elliptic-Curve Cryptography (ECC). These steps involve obtaining a certificate and installation code, programming them onto the device, setting up unique security keys for incoming devices, and joining new devices with this unique security information. While these security resources (which also include Certicom's ECC library, available upon request from Ember's support team) are not necessary for testing SE networks, any devices wishing to participate in or host a ZigBee-compliant, production-grade (non-test) SE network must implement these features. Readers of this document should be familiar with the ZigBee Smart Energy Application profile (available from the http://www.zigbee.org website in the Standards section) and have basic familiarity with Ember's graphical and command-line-based tools. This document also assumes that the user already has access to the Certicom ECC library (or an ECC-enabled network coprocessor image, for EZSP-based platforms) for their target Ember platform; for access to this content, please contact Ember's support team.

Contents

Ember Corporation
25 Thompson Place
Boston, MA 02210
+1 (617) 951-0200
www.ember.com

Introduction

The processes described in this application note are all those required to set up Smart Energy (SE) elliptic curve cryptography (ECC)-enabled devices. The processes include:

- Test certificate generation
- Certificate file manipulation
- Certificate file programming (for either EM2xx or EM3xx platforms)
- Installation code generation
- Installation code programming (for either EM2xx or EM3xx platforms)
- Application setup (build time)
- Application setup (run time) if using unique link keys
- Joining a new device
- InSight Desktop (ISD) capture setup

Test Certificate Generation

1. Register for an account (for test certificate generation) at http://www.certicom.com/index.php/gencertregister. This registration is valid for a limited time.
2. Log in to the test certificate generation site with your account login at http://www.certicom.com/index.php/devicelogin.
3. Go to the certificate generation page: http://www.certicom.com/index.php/smartenergydevicecertificateservice.
4. In the **Subject ID** field, enter the MAC address (EUI64) of the target device where the certificate will reside (MSB order, such as 000D6F... for Ember EUI64s). This can be different than your target node's current EUI64, but if the address differs, the current one will generally be overwritten during programming.
5. In the **Profile Attribute Data** field, enter any custom hex data you want associated with this device. (The recommended format is profile ID [0109] + cert type [1 for test] + 16-bit ZigBee Manufacturer Code + optional customer data of any length.)
6. Leave the **Public Key** field empty!
7. Click the **Generate** button.
8. Highlight and copy the text in the HTML table on the resulting page.
9. Open a text editor and paste the contents of the table into the text file; save the file.

Certificate File Modification

Note: This portion of the process is subject to change if Certicom's certificate generation web page alters its output format. Ember is working with Certicom in the hope of achieving a more consistent output format in the future, one which would require less manual editing before programming.

At the beginning of this process, the certificate text file you created in the previous step should look something like this (with different hex numbers in the fields):

CA Pub Key: 0200fde8a7f3d1084224962a4e7c54e69ac3f04da6b8 Device Implicit Cert: 02040e92886475999701a626a75be9cfb9ccb0ee7481

000d6f000092e047544553545345434101091083d1bbd1bbd1bb

Device Private Key: 00ea5606dec9ec8f85a5f53405b67b2988b74bcd73 Device Public Key: 0202f25c9e4aaf910d6bbc16f444715d39962e3a5500

1. (optional) Remove "Device Public Key" line from the certificate text file, as Ember tools do not need it. If you do not remove this line, it is ignored.
2. Remove extra line breaks in the hex string following "Device Implicit Cert:" so that all characters are on one line.
3. Change "CA Pub Key" to "CA Public Key"
4. Make sure that only a single blank space is between the ":" and the hex string of each line (no tabs or extra white space).

At the end of the process, the certificate text file should look like the following, with the Device Implicit Cert line on a single line:

CA Public Key: 0200fde8a7f3d1084224962a4e7c54e69ac3f04da6b8

Device Implicit Cert: 02040e92886475999701a626a75be9cfb9ccb0ee7481000d6f000092e047544553545345434101091083d1bbd1bbd1bb

Device Private Key: 00ea5606dec9ec8f85a5f53405b67b2988b74bcd73

Certificate File Programming

EM2xx Platform

1. Ensure that EM2xx utilities (SIF Toolkit) are installed. These are installed as part of the xIDE For EM250 installer or the EmberZNet EM260 installer. If you want to verify, you can check your computer's Programs list for "xIDE for EM250" or "EmberZNet Stack <version> em260".
2. Change to the directory where the certificate text file is stored.
3. Determine the IP address of target node's InSight Adapter [ISA]. You can use the InSight Desktop (ISD) adapter properties window to view the IP address and other ISA/node details.
4. Run the following command to apply the certificate (shown for IP address 192.168.0.1 and certificate filename cert_001.txt):

```
em2xx_patch -ip 192.168.0.1 -Mfg -loadCert cert_001.txt
```

Note: If you are overriding previously written certificate data, you may need to add an "-Override" flag before the "-Mfg" flag. However, use this option with caution, as it can permanently corrupt your device if the flash-writing process is terminated unexpectedly.

5. Optionally, verify written certificate data with a command like the following:

```
em2xx_read -id 192.168.0.1 -print
```

EM3xx Platform

1. Ensure that ISA3 Utilities (version 1.0.2 or higher) are installed. If you want to verify, you can check your computer's Programs list for "Ember ISA3 Utilities".
2. Change to the directory where the certificate text file is stored.
3. Determine the IP address of the target node's ISA3. You can use the InSight Desktop (ISD) adapter properties window to view the IP address and other ISA3/node details.
4. Run the following command to apply the certificate (shown for IP address 192.168.0.1 and certificate filename cert_001.txt):

```
em3xx_load --ip 192.168.0.1 --cibtokenspatch cert_001.txt
```

> **Note:** If you are overriding previously written certificate data, you may need to add an "--Override" flag before the "--cibtokenspatch" flag. However, use this option with caution, as it can permanently corrupt your device if the flash-writing process is terminated unexpectedly.

5. Optionally, verify written certificate data with a command like the following:

```
em3xx_load --ip 192.168.0.1 --cibtokensprint
```

Installation Code Generation

1. Pick a random hex string of 6, 8, 12, or 16 bytes. You can generate this programmatically with a random number generate if you wish.
2. Open a text editor and enter this string (as ASCII) in a file by itself, preceded by the string "Install Code: ". Note the single whitespace character following the colon.
3. Save this file for later use.

Installation Code Programming

> **Note:** This is not necessary for devices that only **form** SE networks (like an Energy Services Interface (ESI), which is typically the Trust Center for the network) rather than **join** them.

EM2xx Platform

1. Change to directory where installation code file you created in the previous step is stored.
2. Determine the IP address of target node's InSight Adapter [ISA]. You can use ISD's adapter properties window to view the IP address and other ISA/node details.
3. Run the following command to apply the installation code (shown for IP address 192.168.0.1 and installation code filename inst_001.txt):

```
em2xx_patch -ip 192.168.0.1 -Mfg -loadInst inst_001.txt
```

> **Note:** If you are overriding previously written installation code data, you may need to add an "-Override" flag before the "-Mfg" flag. However, use this option with caution, as it can permanently corrupt your device if the flash-writing process is terminated unexpectedly.

4. Optionally, verify written installation code data with a command like the following:

```
em2xx_read -id 192.168.0.1 -print
```

EM3xx Platform

1. Change to the directory where the installation code file is stored.
2. Determine the IP address of the target node's ISA3. You can use ISD's adapter properties window to view the IP address and other ISA3/node details.
3. Run the following command to apply the installation code (shown for IP address 192.168.0.1 and installation code filename inst_001.txt):

```
em3xx_load --ip 192.168.0.1 --cibtokenspatch inst_001.txt
```

> **Note:** If you are overriding previously written installation code data, you may need to add an "--Override" flag before the "--cibtokenspatch" flag. However, use this option with caution, as it can permanently corrupt your device if the flash-writing process is terminated unexpectedly.

4. Optionally, verify written installation code data with command like the following:

```
em3xx_load --ip 192.168.0.1 --cibtokensprint
```

Application Setup (Build Time)

Ensure that you have the ECC library for your SoC platform. Access is granted by Ember Support upon request

1. Create a new AppBuilder configuration from ISD's File | New | Application Configuration menu.
2. In AppBuilder configuration, on the Stack Configuration tab, check the following settings:
    1. If you are using unique, per-device link keys (as should be the case for production deployments), set the **Security** option to "Smart Energy Security full (compliant)".

       If you are using a single, global link key for all devices (often used in development/testing scenarios to reduce complexity), set the **Security** option to "Smart Energy Security test".

    2. **Use Real ECC** is enabled (checked).

3. The **Library path** next to the **Use Real ECC** checkbox points to the location of your ECC library file (*.xap for EM250 or *.a for EM35x).

Note: Some versions of ISD don't support white space in the ECC library path, so you may need to relocate these files to satisfy this requirement.

3. Set up the remaining SE device configuration as appropriate, and generate the project.
4. Populate callbacks in the generated project as necessary, then build and load to the target device (after the certificate and installation codes have been programmed as described above).

Application Setup (Run Time) Prior to Joining a New Device to the HAN

Note: This process is only required if you are using installation codes to generate unique, per-device link keys, such as when "Smart Energy Security (full)" is selected as the security model for your application configuration in AppBuilder.) If you are using a global link key for joining, such as when using AppBuilder's "Smart Energy Security (test)" security model, skip this section.

This section describes the process of setting up the Trust Center device (the network coordinator) for the SE network to accept an incoming SE device into its home area network (HAN). It also describes the process of setting up a new HAN device prior to joining the HAN created by this Trust Center.

Procedure for Development Prototypes / Debugging

This process relies on the serial command line interface (CLI) to Ember's Application Framework. If the CLI is no longer supported or accessible on your network's Trust Center or incoming HAN device, please refer to the "Procedure for Production/Field Deployments" later in this section.

Note: Ember's hashing-CLI tool, which can assist with this process, is available on request through Ember Support. It uses source code provided in the ZigBee document sections referenced in the Procedure for Production/Field Deployments section. It requires Cygwin.

1. Obtain a device-specific link key. You can do this in two ways.

   The first requires two steps:

   - Before joining a HAN device to the SE network, determine its installation code string, including its 16-bit CRC (in LSB order), as well as its EUI64 (MAC address). You can do this using the em2xx_read/em3xx_load tools with the appropriate "print" option to verify the installation code data. See the "Installation Code Programming" section for more information.
   - Use the MMO hash algorithm to compute the device-specific link key from the installation code. If you are using Ember's hashing-CLI tool, you can run "./hashing-cli.exe -i <installCodeString>", where installCodeString is your 6/8/12/16-digit code without the CRC16 appended, to determine the hash result. Devices running

EmberZNet 4.3.0 and later can also perform an AES MMO hash operation using the emberAesHashSimple() method provided on SoC platforms or the ezspAesMmoHash() API provided on EZSP host platforms.

Alternatively you can have your HAN device attempt to join a network (using Button1 or the "network join ..." CLI command, for example). Then, after the joining fails, use the "keys print" CLI command to print the TC Link Key, which should now contain the hash result. You can then use the "info" command to print the device's EUI64 if you don't already have it. This alternate method allows you to skip the two steps above.

2. Once you have the EUI64 and device-specific link key, access the trust center for this HAN and register the key into the key table using the "option link ..." CLI command. This command takes a table index, an EUI64 specified in MSB order (or LSB if using EmberZNet versions older than 4.3), and a key (in MSB order, just as it is printed by "keys print") for that device. For example, for EUI64 0x000D6F0011223344 and key 0x00112233445566778899AABBCCDDEEFF (shown with EUI in MSB as expected in newer stack versions):

```
option link 0 {44 33 22 11 00 GF 0D 00} {00 11 22 33 44 55 66 77 88 99 AA
BB CC DD EE FF}
```

3. Confirm that the proper key table entry now exists and is displayed in the output of the "keys print" command at the trust center.

Procedure for Production/Field Deployments

1. Before joining a HAN device to the SE network, determine its installation code string, including its 16-bit CRC (in LSB order), as well as its EUI64 (MAC address); these byte values are meant to be published externally with the device for use during installation.
2. Using some out-of-band (non-ZigBee) method, such as verbally or through some proprietary communications interface, convey the HAN device's installation code to the ESI for the HAN, its trust center device (if different from the ESI), or some head-end device in the utility's backhaul network that has access to the HAN's ESI.
3. Have the ESI (or the utility’s head-end) sanity-check the provided installation code by computing the CRC of those hex bytes (less the final two, which are the provided CRC16 from the installer) using the CRC16 algorithm described in the "CRC Algorithm Information" section (section 5.4.8.1.1.1 in the current [r15] version) of the ZigBee Smart Energy Application Profile Specification (ZigBee document 075356). The computed CRC (when converted into LSB order) for the 6/8/12/16-digit code should match the last 4 digits of the provided installation code string.
4. Once the installation code string (variable-length code + 2-byte CRC16 in LSB order) has been verified, have the ESI or its head-end compute the device-specific initial link key by performing an AES MMO hash function against the entire 8/10/14/18-byte string, using the algorithm described in the "MMO Hash Code Example" section (section 5.4.8.1.2.1 in the current [r15] version) of the ZigBee Smart Energy Application Profile Specification (ZigBee document 075356). Devices running EmberZNet 4.3.0 and later can perform this AES MMO hash operation using the emberAesHashSimple() method provided on SoC platforms or the ezspAesMmoHash() API provided on EZSP host platforms.

5. Once this key has been determined for the incoming HAN device, install a link key table entry in the HAN's trust center by using the emberSetLinkKeyTableEntry() function with the device-specific key and EUI-64.

Joining Process for New Device

If you are using installation codes to create unique keys, the application setup process discussed in the previous section "Application Setup (Run Time) Prior to Joining a New Device to the HAN" should have been performed.

1. Join the new HAN device to the ESI's HAN (using Button1 or "network join ..." CLI command or emberScanForJoinableNetwork() API function, for example).
2. The ESI (as Trust Center) will trigger its emberAfTrustCenterJoinCallback() to indicate the outcome of the initial joining process. If successful, the joining device should begin CBKE (certificate-based key establishment).
3. If both the ESI and HAN devices support CBKE and ECC and have valid certificates issued by the same authority, CBKE should succeed, and the HAN device should move on to registration (asking the ESI to bind to it on specific clusters). When the SE registration process completes, the HAN device will trigger its emberAfRegistrationCallback() to notify the application.
4. After the SE registration completes successfully, the HAN device can now send/receive messages to/from the ESI using its CBKE-authorized link key.

InSight Desktop (ISD) Capture Setup

To capture the initial joining process before CBKE, first determine the device's preconfigured link key (either the unique one generated from the installation code as described in the section "Application Setup (Run Time) Prior to Joining a New Device to the HAN" or the global one if using SE Security Test mode). Enter this code into ISD using either of the two methods described below before attempting to join the HAN.

To capture the encrypted SE traffic after CBKE/registration has been performed, determine the CBKE-authorized link key by querying either the HAN device or the ESI. If the serial CLI is still available, you can use the "keys print" command. If you are using the Ember Stack API directly, use the emberGetKey() API. Enter the discovered key into ISD (see the two methods below) after CBKE completes; remember that this key is unique for each instance of CBKE, even for the same pair of devices.

You can enter the key into ISD in either of two ways:

1. Through the main list of keys at File: Preferences: Decoding: Security Keys. This data entry method is more flexible, but keys must be entered here before beginning a capture session, else they won't be used for decryption.
2. Through the live capture keys menu at Edit: Live Capture Security Keys. This has a very basic data entry method but the keys are applied to decryption of future events right away, even in already open (or still-capturing) logs. Note that the traffic captured

before entering this key won't be decrypted unless you save and re-open the capture session.

After reading this document

If you have questions or require assistance with the procedures described in this document, contact Ember Customer Support at http://www.ember.com/support_index.html.

ember

The information in this document is subject to change without notice. The statements, configurations, technical data, and recommendations in this document are believed to be accurate and reliable but are presented without express or implied warranty. Users must take full responsibility for their applications of any products specified in this document. The information in this document is the property of Ember Corporation.

Title, ownership, and all rights in copyrights, patents, trademarks, trade secrets, and other intellectual property rights in the Ember Proprietary Products and any copy, portion, or modification thereof, shall not transfer to Purchaser or its customers and shall remain in Ember and its licensors.

No source code rights are granted to Purchaser or its customers with respect to all Ember Application Software. Purchaser agrees not to copy, modify, alter, translate, decompile, disassemble, or reverse engineer the Ember Hardware (including without limitation any embedded software) or attempt to disable any security devices or codes incorporated in the Ember Hardware. Purchaser shall not alter, remove, or obscure any printed or displayed legal notices contained on or in the Ember Hardware.