PLUG-AND-TRUST-NANO-PACKAGE

Release 1.2.4

NXP

Apr 29, 2024

CONTENTS

Content

CHAPTER ONE

CHANGELOG

Release v1.2.4

- Examples for Zephyr updated with board overlay files.
- Added manifest file for nano package integration in Zephyr OS (zephyr/west.yml)

Release v1.2.3

- Zephyr integration branch (feature/zephyr-integration) merged to master.
- Zephyr integration updated to zephyr release - v3.5.0.

Release v1.2.2

- MbedTLS alt ecdsa bug fix - Updated function used to extract r and s using mbedtls APIs

Release v1.2.1

- Updated log messages for k64 port files
- Added retry on I2C failed error. (Disabled by default). To enable - uncomment T1OI2C_RETRY_ON_I2C_FAILED in phNxpEsePal_i2c.h.
- Increased t=1oi2c retry count. (MAX_RETRY_COUNT).

Release v1.2.0

- Added security fix on 24-Feb-2023 to prevent buffer overflow on the T=1oI2C stack.
- se05x_GetInfo example added to get SE05X platform information.

Release v1.1.1

- Added a check to prevent a potential buffer overflow issue in T=1OI2C stack

Release v1.1.0

- **Features**
  - Added Secure Authenticator (Qi) examples
  - Integration of twister framework from zephyr OS

Release v1.0.0

- Initial commit
- **Features**
  - ECDSA and ECDH with NIST P256
  - AES Encrypt / Decrypt (ECB,CBC,CTR)
  - Binary Objects

- Encrypted I2C communication using PlatformSCP channel based on Global Platform SCP03
- Platforms - Linux, frdm-k64 bare metal, Zephyr OS

CHAPTER

TWO

INTRODUCTION

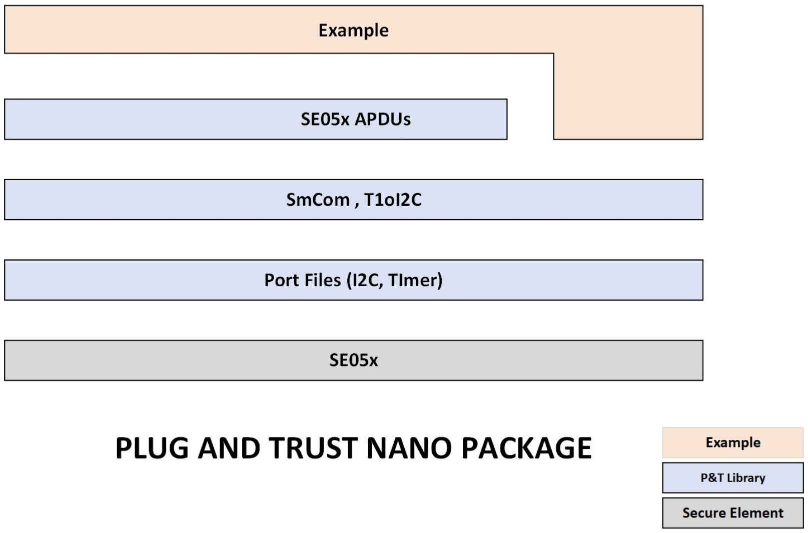
2.1 Introduction to the Plug & Trust Nano Package

The Plug & Trust Nano package is an optimized middleware for communicating between a host processor or microcontroller and the EdgeLock SE05x and A5000 secure elements and authenticators. The Plug & Trust Nano Package has been designed for memory constrained devices and consumes only ~1KB of RAM for SCP03 encrypted communication over I2C.

Note that the examples and libraries contained in the Plug & Trust Nano package have been specifically designed to fit into constrained devices and are not compatible with examples and libraries available in the standard Plug & Trust package.

The standard Plug and Trust middleware package can be downloaded from https://www.nxp.com/products/:SE050. The package has support for more crypto curves, plugins, examples and more platforms.

Nano Package -



Nano package Features

- ECDSA and ECDH with NIST P256
- AES Encrypt / Decrypt (ECB,CBC,CTR)
- Binary Objects
- Encrypted I2C communication using PlatformSCP channel baed on Global Platform SCP03 channel
- Platforms - Linux, frdm-k64 bare metal, Zephyr OS

Folder structure

```
example --- se05x examples
lib  ------ se05x library files
 |
 |__
        apdu -------- Contains se05x apdu apis with scp03 support
        platform ---- Platform specific files. Modify / add the files here to support other platform
        t1oi2c ------ T1OI2C files
        mbedtls_alt - Mbedtls ALT files to access SE05x
```

2.2 PlatformSCP03

Using nano package, host can establish encrypted I2C communication via PlatformSCP channel (based on Global Platform SCP03). This requires some host crypto operations.

Plug and Trust Nano package has these host crypto apis implemented using

- openssl (version 1.1.1). (simw-nanopkg/lib/apdu/scp03/openssl)
- tinyCrypt (simw-nanopkg/lib/apdu/scp03/tc)

To use a different host crypto, re-implement the host crypto apis - *simw-nanopkg/lib/apdu/scp03/se05x_scp03_crypto.h*

When building the example with ‘Platform SCP’ enabled, make sure to assign valid scp03 keys to session context. (DEK key is required only for key rotation - se05x_rotate_scp03_keys).

Note: Product Deployment => Make sure to store the SCP03 keys securly.

The Default Platform SCP keys for ease of use configurations are present in

- SE050 Configuration: https://www.nxp.com/docs/en/application-note/AN12436.pdf
- SE051 Configuration: https://www.nxp.com/webapp/Download?colCode=AN12973

```
void ex_set_scp03_keys(pSe05xSession_t session_ctx)
{
    session_ctx->pScp03_enc_key    = &scp03_enc_key[0];
    session_ctx->pScp03_mac_key    = &scp03_mac_key[0];
    session_ctx->pScp03_dek_key    = NULL;
    session_ctx->scp03_enc_key_len = 16;
    session_ctx->scp03_mac_key_len = 16;
    session_ctx->scp03_dek_key_len = 0;
    return;
}
```

2.3 Build option

Platform SCP03

```
-DPLUGANDTRUST_SCP03=ON : Build with Platform SCP03 enabled
-DPLUGANDTRUST_SCP03=OFF : Build with Platform SCP03 disabled
```

Debug Logs

```
-DPLUGANDTRUST_DEBUG_LOGS=ON : Build with Debug logs enabled
-DPLUGANDTRUST_DEBUG_LOGS=OFF : Build with Debug logs disabled
```

2.4 Examples

Examples on linux

Refer *simw-nanopkg/examples/<example>/readme.rst*.

Examples on FRDM-K64F

Refer *simw-nanopkg/se05x_sign/k64f/readme.rst*.

Refer *simw-nanopkg/se05x_crypto/k64f/readme.rst*.

Examples on Zephyr OS

Integration of nano package in Zephyr OS (Tested with release tag - zephyr-v3.5.0).

Refer *simw-nanopkg/zephyr/readme.rst*.

Note: To use policies with objects refer 'test_nist256_sign_policy' in 'Se05x Crypto' example. For more details on policies, Refer Section '3.7 Policies' in https://www.nxp.com/docs/en/application-note/AN12413.pdf

2.5 Porting

Platform specific files are maintained in **simw-nanopkg/lib/platform** folder.

Modify / add the files here to support other platforms. By default port files are available for Linux, Zephyr and K64 MCU.

2.6 Mbedtls Alt files

Nano package provides MbedTLS Alt files as an alternative/additional approach to access the secure element using mbedTLS.

In the current implementation only ECDSA Sign is supported via MbedTLS ALT files.

Note - The session for se05x is opened and closed for every ECDSA sign. For the product deployment, make sure to change the logic as required.

Using Mbedtls Alt files in Zephyr OS

Set **CONFIG_PLUGANDTRUST_MBEDTLS_ALT** to build Plug and Trust with Mbedtls Alt files.

GCP cloud example in Zephyr OS is modified to use SE05x for ECDSA sign.

Prerequisite - SE05x provisioned with private key at location (say 0x11223344).

Replace the private key in *zephyr/samples/net/cloud/google_iot_mqtt/src/private_info/key.c* with the reference to provisioned private key.

The following provides an example of an EC reference key. The value reserved for the private key has been used to contain:

- a pattern of `0x10..00` to fill up the datastructure MSB side to the desired key length
- a 32 bit key identifier (in the example below `0x11223344`)
- a 64 bit magic number (always `0xA5A6B5B6A5A6B5B6`)
- a byte to describe the key class (`0x10` for Key pair)
- a byte to describe the key index (use a reserved value `0x00`)

```
Private-Key: (256 bit)
priv:
    10:00:00:00:00:00:00:00:00:00:00:00:00:00:00:
    00:00:00:11:22:33:44:A5:A6:B5:B6:A5:A6:B5:B6:
    10:00
```

Refer *zephyr/samples/net/cloud/google_iot_mqtt/README.rst* to build GCP cloud example.

2.7 Memory Details

Following are the memory details of se05x sign example (plain session) when build in release mode for K64.

Text - 43136 Data - 2520 Bss - 41672

CHAPTER

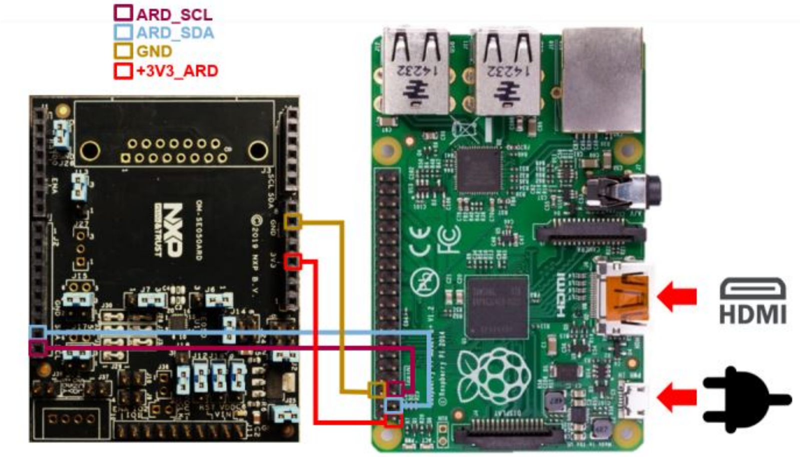
THREE

GETTING STARTED (LINUX)

Prerequisite

Raspberry Pi with raspbian OS installed.

Rpi Connections -



OM-SE05xARD wiring to the Raspberry Pi board

3.1 SE05x Crypto Example

Overview

This example demonstrates SE05x crypto functionality using se05x APIs.

Refer file - 'simw-nanopkg/examples/se05x_crypto/src/ex_se05x_crypto.c'

Note: When building the example with 'Platform SCP' enabled, make sure to assign valid scp03 keys to session context.

Linux Prerequisite

Install cmake , Openssl 1.1.1

```
sudo apt-get install cmake cmake-curses-gui cmake-gui libssl-dev
```

Linux build

To build example run

```
cd simw-nanopkg/examples/se05x_crypto/linux
mkdir build
cd build
cmake ../
make
./ex_se05x_crypto
```

Build options

Platform SCP03

```
-DPLUGANDTRUST_SCP03=ON -- Build with Platform SCP03 enabled
-DPLUGANDTRUST_SCP03=OFF -- Build with Platform SCP03 disabled
```

Debug Logs

```
-DPLUGANDTRUST_DEBUG_LOGS=ON -- Build with Debug logs enabled
-DPLUGANDTRUST_DEBUG_LOGS=OFF -- Build with Debug logs disabled
```

Sample Output

If everything is successful, the output will be similar to:

```
./ex_se05x_crypto
Plug and Trust nano package - version: 1.0.0
Establish Secure Channel to SE05x !
Get Version ==>
Applet Version 6.0.0
test_get_version complete
test_generate_nist256_key complete
test_set_get_nist256_key complete
test_nist256_sign_verify complete
test_set_certificate complete
test_ecdh complete
test_aes_ECB_NOPAD complete
test_aes_CBC_NOPAD complete
test_aes_CTR complete
test_nist256_sign_policy complete
```

3.2 SE05x Sign Example

Overview

This example demonstrates signing a data using nist256 key.

Refer file - ‘simw-nanopkg/examples/se05x_crypto/src/ex_se05x_sign.c’.

Note: When building the example with ‘Platform SCP’ enabled, make sure to assign valid scp03 keys to session context.

Linux Prerequisite

Install cmake , Openssl 1.1.1

```
sudo apt-get install cmake cmake-curses-gui cmake-gui libssl-dev
```

Linux build

To build example run:

```
cd simw-nanopkg/examples/se05x_crypto/linux
mkdir build
cd build
cmake ../
make
./ex_se05x_sign
```

Build options

Platform SCP03

```
-DPLUGANDTRUST_SCP03=ON : Build with Platform SCP03 enabled
-DPLUGANDTRUST_SCP03=OFF : Build with Platform SCP03 disabled
```

Debug Logs

```
-DPLUGANDTRUST_DEBUG_LOGS=ON : Build with Debug logs enabled
-DPLUGANDTRUST_DEBUG_LOGS=OFF : Build with Debug logs disabled
```

Sample Output

If everything is successful, the output will be similar to:

```
Plug and Trust nano package - version: 1.0.0
Generate ecc key
Signature ==>
0X30 0X46 0X2 0X21 0 0X84 0X69 0X1F 0XFE 0XBC 0XC2 0X2F 0X10 0XBB 0X8D 0X95 0X9A 0X26␣
↪0XD3 0XE2 0X11 0X62 0X81 0XF2 0X7D 0X3 0X5E 0X6B 0X42 0XC8 0X63 0X4F 0XC3 0X50 0XFF␣
↪0XE2 0X3 0X2 0X21 0 0XD6 0XEB 0XD3 0X8D 0X83 0XEB 0XF7 0X6F 0X46 0XF1 0XFA 0XF5 0XF5␣
↪0X24 0XFA 0X26 0X98 0X7A 0X92 0X79 0XBA 0X22 0XAE 0X11 0X1D 0X64 0X8E 0XB0 0XFD 0X48␣
↪0X3C 0XB7
```

3.3 SE05x GetInfo Example

Overview

This example can be used to get SE05X platform information with or without applet installed.

Refer file - 'simw-nanopkg/examples/se05x_crypto/src/ex_se05x_GetInfo.c'

Linux Prerequisite

Install cmake , Openssl 1.1.1

```
sudo apt-get install cmake cmake-curses-gui cmake-gui libssl-dev
```

Linux build

To build example run

```
cd simw-nanopkg/examples/ex_se05x_GetInfo/linux
mkdir build
cd build
cmake ../
make
./ex_se05x_GetInfo
```

Build options

Platform SCP03

```
-DPLUGANDTRUST_SCP03=OFF -- Build with Platform SCP03 disabled
```

Debug Logs

```
-DPLUGANDTRUST_DEBUG_LOGS=ON -- Build with Debug logs enabled
-DPLUGANDTRUST_DEBUG_LOGS=OFF -- Build with Debug logs disabled
```

Sample Output

If everything is successful, the output will be similar to:

```
    ./ex_se05x_GetInfo
Se05x Getinfo Example !
Plug and Trust nano package - version: 1.2.0
I2C driver supports plain i2c-level commands.
#####################################################
Close i2c device 3.
Plug and Trust nano package - version: 1.2.0
I2C driver supports plain i2c-level commands.
#####################################################
Applet Major = 7
Applet Minor = 2
Applet patch = 0
AppletConfig = 3FFF
With    ECDSA_ECDH_ECDHE
With    EDDSA
With    DH_MONT
With    HMAC
With    RSA_PLAIN
With    RSA_CRT
With    AES
With    DES
With    PBKDF
With    TLS
With    MIFARE
With    I2CM
Internal = FFFF
#####################################################
Tag value - proprietary data 0xFE = 0xFE
Length of following data 0x45 = 0x45
```

(continues on next page)

(continued from previous page)

```
Length of card identification data = 0x42
Tag configuration ID (Must be 0x01) = 0x01
OEF ID = 0xA8 0xFA
Tag patch ID (Must be 0x02) = 0x02
Tag platform build ID1 (Must be 0x03) = 0x03
JCOP Platform ID = J3R351029B411100
Tag FIPS mode (Must be 0x05) = 0x05
FIPS mode var = 0x00
Tag pre-perso state (Must be 0x07) = 0x07
Bit mask of pre-perso state var = 0x00
Tag ROM ID (Must be 0x08) = 0x08
se05x_GetInfoPlainApplet Example Success !!!...
#####################################################
Close i2c device 3.
SE05x Getinfo Example Success !
```

3.4 SE05x Mandate SCP03 Example

Overview

This example demonstrates how to mandate the use of Platform SCP by calling SetPlatformSCPRequest.

This is a persistent state.

SetPlatformSCPRequest APDU can be sent in session authenticated with 'RESERVED_ID_PLATFORM_SCP' user id.

The example can be used to either 'Mandate PlatformSCP' or remove the 'Mandate PlatformSCP' state.

When example is built with -DPLUGANDTRUST_SCP03=OFF, ex_se05x_mandate_scp03_set is built. (Set Mandate PlatformSCP).

When example is built with -DPLUGANDTRUST_SCP03=ON, ex_se05x_mandate_scp03_remove is built. (Removes Mandate PlatformSCP).

Refer file - 'simw-nanopkg/examples/se05x_mandate_scp03/src/ex_se05x_mandate_scp03.c'.

Note: When building the example with 'Platform SCP' enabled, make sure to assign valid scp03 keys to session context.

Linux Prerequisite

Install cmake , Openssl 1.1.1

```
sudo apt-get install cmake cmake-curses-gui cmake-gui libssl-dev
```

Linux build

To build example run

```
cd simw-nanopkg/examples/se05x_mandate_scp03/linux
mkdir build
cd build
cmake ../ -DPLUGANDTRUST_SCP03=OFF
```

(continues on next page)

(continued from previous page)

```
make
./ex_se05x_mandate_scp03_set

cd simw-nanopkg/examples/se05x_mandate_scp03/linux
mkdir build
cd build
cmake ../ -DPLUGANDTRUST_SCP03=ON
make
./ex_se05x_mandate_scp03_remove
```

Build options

Platform SCP03

```
-DPLUGANDTRUST_SCP03=ON : Build with Platform SCP03 enabled
-DPLUGANDTRUST_SCP03=OFF : Build with Platform SCP03 disabled
```

Debug Logs

```
-DPLUGANDTRUST_DEBUG_LOGS=ON : Build with Debug logs enabled
-DPLUGANDTRUST_DEBUG_LOGS=OFF : Build with Debug logs disabled
```

Sample Output

If everything is successful, the output will be similar to:

```
./ex_se05x_mandate_scp03_set
Plug and Trust nano package - version: 1.0.0
Sending PlatformSCPRequest_REQUIRED command
Example successful

./ex_se05x_mandate_scp03_remove
Plug and Trust nano package - version: 1.0.0
Establish Secure Channel to SE05x !
Sending PlatformSCPRequest_NOT_REQUIRED command
Example successful
```

3.5 SE05x Rotate SCP03 Example

Overview

This example demonstrates how to update SCP03 keys in SE05x.

On running the example, it will update the SCP03 keys and revert back to original keys.

The example works only with Platform SCP03 enabled.

Refer file - 'simw-nanopkg/examples/se05x_rotate_scp03_keys/src/ex_se05x_rotate_scp03_keys.c'.

Note: When building the example with 'Platform SCP' enabled, make sure to assign valid scp03 keys to session context. All keys are necessary for this operation - enc, mac, dek.

Linux Prerequisite

Install cmake , Openssl 1.1.1

```
sudo apt-get install cmake cmake-curses-gui cmake-gui libssl-dev
```

Linux build

To build example run

```
cd simw-nanopkg/examples/se05x_rotate_scp03_keys/linux
mkdir build
cd build
cmake ../ -DPLUGANDTRUST_SCP03=ON
make
./ex_se05x_rotate_scp03_keys
```

Build options

Platform SCP03

```
-DPLUGANDTRUST_SCP03=ON : Build with Platform SCP03 enabled
-DPLUGANDTRUST_SCP03=OFF : Build with Platform SCP03 disabled
```

Debug Logs

```
-DPLUGANDTRUST_DEBUG_LOGS=ON : Build with Debug logs enabled
-DPLUGANDTRUST_DEBUG_LOGS=OFF : Build with Debug logs disabled
```

Sample Output

If everything is successful, the output will be similar to:

```
./ex_se05x_rotate_scp03_keys
Plug and Trust nano package - version: 1.0.0
Establish Secure Channel to SE05x !
Changing SCP03 keys(version - 0b) to NEW KEYS
Congratulations !!! Key Rotation Successful!
Reverting SCP03 keys(version - 0b) to OLD KEYS
Congratulations !!! Key Rotation Successful!
```

3.6 SE05x SCP03 Resume example

Overview

This example demonstrates SCP03 session resumption with SE05x.

Refer file - 'simw-nanopkg/examples/se05x_resume_scp03/src/ex_resume_scp03.c'.

Note: Make sure to assign valid SCP03 keys to session context.

Linux build

To build example run:

```
cd simw-nanopkg/examples/se05x_resume_scp03/src
mkdir build
cd build
cmake ../ -DPLUGANDTRUST_SCP03
make
./ex_establish_scp03
./ex_resume_scp03
```

Build options

Debug Logs

```
-DPLUGANDTRUST_DEBUG_LOGS=ON : Build with Debug logs enabled
-DPLUGANDTRUST_DEBUG_LOGS=OFF : Build with Debug logs disabled
```

Sample Output

If everything is successful, the output will be similar to:

```
SE05x SCP03 Establish !
Plug and Trust nano package - version: 1.0.0
Establish Secure Channel to SE05x !
Simply writing the session keys to the file system is not a secure implementation. It␣
↪must not be used in production !!!...
SE05x SCP03 Establish Success !

SE05x SCP03 Resume Example !
Plug and Trust nano package - version: 1.0.0
Resuming Secure Channel to SE05x !
Simply writing the session keys to the file system is not a secure implementation. It␣
↪must not be used in production !!!...
SE05x SCP03 Resume Success !
```

CHAPTER

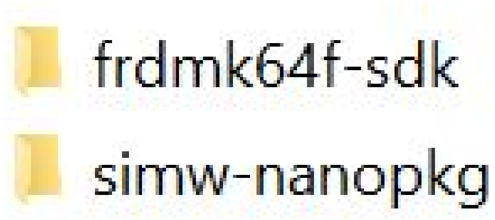
FOUR

GETTING STARTED WITH K64

4.1 SE05x Sign Example - frdm-k64

Prequisite

1. Download the frdmk64f SDK from https://mcuxpresso.nxp.com/en/select (version - 2.12). Ensure MMCAU is selected.
2. unzip and place the sdk in parallel to the nano package as shown in the image below. Rename the sdk folder to "frdmk64f-sdk".

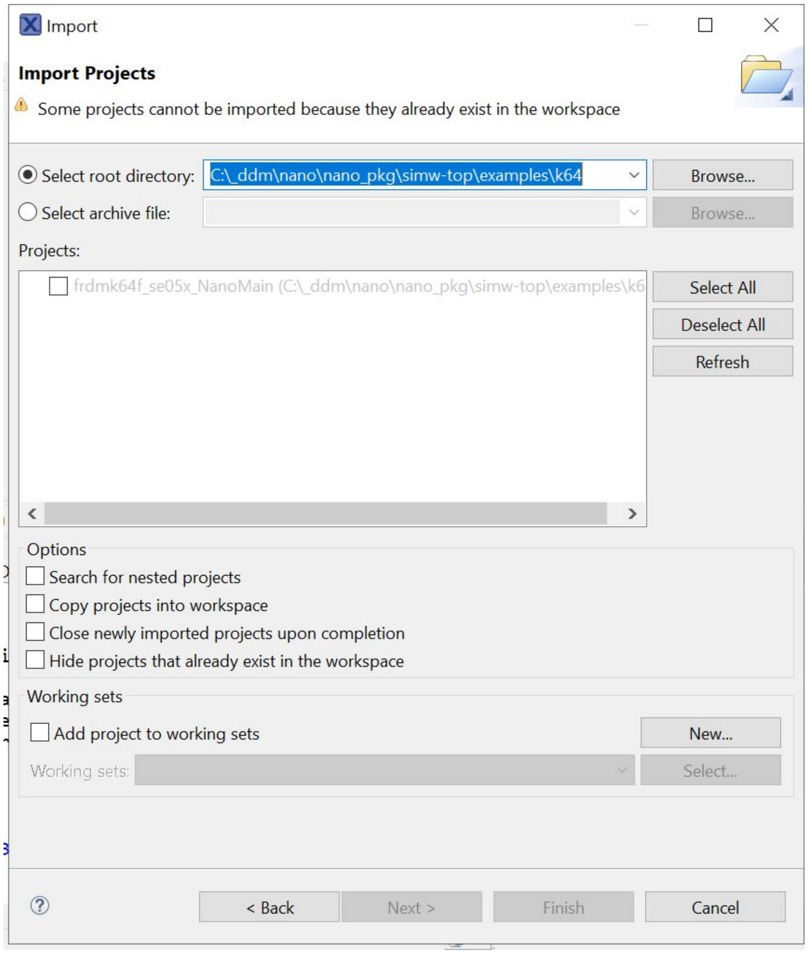


frdmk64f-sdk

simw-nanopkg

Import the project

1. Click on File, Import, Existing project to workspace and click on next.
2. Point to the "simw-nanopkg/examples/se05x_sign/k64f" folder
3. Select the Project and click on Finish



Import

Import Projects

⚠ Some projects cannot be imported because they already exist in the workspace

● Select root directory: C:_ddm\nano\nano_pkg\simw-top\examples\k64 [Browse...]

○ Select archive file: [Browse...]

Projects:

□ frdmk64f_se05x_NanoMain (C:_ddm\nano\nano_pkg\simw-top\examples\k6

[Select All] [Deselect All] [Refresh]

Options

□ Search for nested projects

□ Copy projects into workspace

□ Close newly imported projects upon completion

□ Hide projects that already exist in the workspace

Working sets

□ Add project to working sets [New...]

Working sets: [Select...]

[< Back] [Next >] [Finish] [Cancel]

Build and Debug

1. Click on Build and then Debug on the Quickstart panel to Build and Debug your project

CHAPTER

FIVE

GETTING STARTED WITH ZEPHYR + SE05X

5.1 Overview

Plug-and-Trust nano package can be used to add the EdgeLock SE05x and A5000 secure elements and authenticators support in Zephyr OS. (Tested with release tag - zephyr-v3.5.0)

Refer `doc/plug-and-trust-nano-package-api-doc.pdf` for Plug and Trust Crypto APIs.

5.2 Zephyr Integration / Build

Refer official Zephyr Getting Started Guide to Set up the Zephyr development environment.

Clone the nano package and Zephyr (required modules) as below

```
west init -m https://github.com/NXPPlugNTrust/nano-package.git --mf zephyr\west.yml workspace
cd workspace
west update
```

Note: The west.yml file will clone the Zephyr v3.5.0.

5.3 Build Options

Use the below options in prj.conf file of the example.

```
CONFIG_PLUGANDTRUST=y/n   =================> Enable / Disable Plug and Trust lib support.
CONFIG_PLUGANDTRUST_SCP03=y/n =============> Enable / Disable Platform SCP03 support.
CONFIG_PLUGANDTRUST_LOG_LEVEL_DBG=y/n =====> Enable / Disable Plug and Trust logs.
```

Use the board overlay files to set the i2c port to alias - *se05x-i2c*. Refer frdm_k64f.overlay file for reference.

5.4 Examples

Build Plug and Trust examples on Zephyr OS as

```
cd workspace/
west build -b <BOARD> modules/crypto/nxp-plugandtrust/examples/<EXAMPLE_NAME>/zephyr/ --
↪pristine
```

5.5 Test Runner (Twister)

Using the zephyr twister script, Plug and Trust examples / tests can be run on K64F as

```
python3 scripts/twister -p frdm_k64f --device-testing -device-serial <serial_port> -T ../
↪modules/crypto/nxp-plugandtrust/ --west-flash --west-runner=jlink
```

Note: Twister script is tested with ubuntu 20.04 machine and zephyr 3.0.0.

5.6 Qi examples

5.6.1 Secure Authenticator (Qi) Authentication demo

Overview

This project is used to demonstrate the Qi authentication flow between a Power Transmitter and a Power Receiver. The Power Transmitter implements 3 functions for the 3 authentication requests a Receiver can issue to the Transmitter : `GetCertificateChainDigest`, `ReadCertificates`, `Authenticate`.

Pre-requisites

- The secure element should be trust provisioned with correct keys and certificates for Qi Authentication. Keys and certificates can be provisioned for test purpose by updating keys in `examples/se05x_qi_auth/sa_qi_provisioning/sa_qi_credentials.c` and running example `Secure Authenticator (Qi) Provisioning demo`.
- By default WPC Root certificate is used in the certificate chain. If example `Secure Authenticator (Qi) Provisioning demo` is run, you would need to disable macro `USE_ROOT_WPCCA` in `sa_qi_rootcert.c` to use test RootCA:

  ```
  #ifndef USE_ROOT_WPCCA
  #define USE_ROOT_WPCCA 1
  #endif
  ```

GetCertificateChainDigest (GET_DIGESTS)

This function reads the digests of certificate chains stored inside the secure element and returns all the digests as requested by the Power Receiver.

```
void GetCertificateChainDigest(const uint8_t *pGetDigestRequest,
    const size_t getDigestRequestLen,
    uint8_t *pDigestResponse,
    size_t *pDigestResponseLen)
{
    smStatus_t retStatus        = SM_NOT_OK;
    uint32_t certChainId        = 0;
    uint16_t objectSize         = 0;
    size_t readSize             = 0;
    uint8_t *pData              = NULL;
    pSe05xSession_t session_ctx = pgSe05xSessionctx;
    uint8_t *pDigestPtr         = NULL;

    qi_error_code_t errorCode  = kQiErrorUnspecified;
    uint8_t authMsgHeader      = 0;
    uint8_t requestedSlotMask  = 0;
    uint8_t slotsPopulated     = 0x00;
    uint8_t slotsReturned      = 0x00;
    uint8_t slotsReturnedCount = 0;

    if (NULL == pGetDigestRequest || NULL == pDigestResponse) {
        LOG_E("Null buffer");
        errorCode = kQiErrorInvalidRequest;
        goto error;
    }

    authMsgHeader = pGetDigestRequest[0];

    if (getDigestRequestLen != GET_DIGESTS_CMD_LEN) {
        LOG_E("Invalid request length");
        errorCode = kQiErrorInvalidRequest;
        goto error;
    }

    requestedSlotMask = pGetDigestRequest[1] & 0x0F;

    /* Invalid slot requested */
    if (requestedSlotMask == 0) {
        errorCode = kQiErrorInvalidRequest;
        LOG_E("No slot requested");
        goto error;
    }

    /* Get all populated slots */
    retStatus = getPopulatedSlots(session_ctx, &slotsPopulated);
    if (SM_OK != retStatus) {
        errorCode = kQiErrorUnspecified;
        LOG_E("Failed to retrieve populated slots");
```

(continues on next page)

(continued from previous page)

```
        goto error;
    }

    /* Is Slot 0 empty? - Return error as slot 0 must always be populated */
    if (!(slotsPopulated & 0x01)) {
        errorCode = kQiErrorUnspecified;
        goto error;
    }

    /* We will return slots which were requested AND are provisioned */
    slotsReturned = requestedSlotMask & slotsPopulated;
    pDigestPtr    = &pDigestResponse[2];

    /* Validate response buffer size */
    for (size_t i = 0; i < MAX_SLOTS; i++) {
        if ((slotsReturned) & (0x01 << i)) {
            slotsReturnedCount++;
        }
    }
    if (*pDigestResponseLen < (size_t)((slotsReturnedCount * DIGEST_SIZE_BYTES) + 2)) {
        /* Response buffer size too less */
        errorCode = kQiErrorUnspecified;
        goto error;
    }

    pData = (uint8_t *)SSS_MALLOC(slotsReturnedCount * DIGEST_SIZE_BYTES);
    if (!pData) {
        errorCode = kQiErrorUnspecified;
        goto error;
    }

    for (size_t i = 0; i < MAX_SLOTS; i++) {
        if ((slotsReturned) & (0x01 << i)) {
            certChainId = QI_SLOT_ID_TO_CERT_ID(i);
            /* Read size of object to allocate necessary memory
             * so that we can read the complete object */
            retStatus = Se05x_API_ReadSize(session_ctx, certChainId, &objectSize);
            if (retStatus != SM_OK) {
                errorCode = kQiErrorUnspecified;
                LOG_E("Failed Se05x_API_ReadSize");
                goto error;
            }
            /* Size of binary object cannot be less than Digest size */
            if (objectSize < DIGEST_SIZE_BYTES) {
                errorCode = kQiErrorUnspecified;
                goto error;
            }
            readSize  = DIGEST_SIZE_BYTES;
            retStatus = Se05x_API_ReadObject(session_ctx, certChainId, 0, DIGEST_SIZE_
↪BYTES, pData, &readSize);
            if (retStatus != SM_OK) {
                errorCode = kQiErrorUnspecified;
```

(continues on next page)

(continued from previous page)

```
                LOG_E("Failed Se05x_API_ReadObject");
                goto error;
            }
            memcpy(pDigestPtr, pData, DIGEST_SIZE_BYTES);
            pDigestPtr += DIGEST_SIZE_BYTES;
        }
    }

    /* Successfully filled all digests - fill response buffer header now */
    pDigestResponse[0]  = (uint8_t)(authMsgHeader & 0xF0) + (uint8_t)(kQiResponseDigest);
    pDigestResponse[1]  = (uint8_t)(slotsPopulated << 4) + (uint8_t)(slotsReturned);
    *pDigestResponseLen = (slotsReturnedCount * DIGEST_SIZE_BYTES) + 2;

    if (pData) {
        SSS_FREE(pData);
    }

    return;

error:
    if (pData) {
        SSS_FREE(pData);
    }
    if (pDigestResponse) {
        pDigestResponse[0]  = (uint8_t)(authMsgHeader & 0xF0) + (uint8_
↪t)(kQiResponseError);
        pDigestResponse[1]  = (uint8_t)(errorCode);
        pDigestResponse[2]  = 0x00;
        *pDigestResponseLen = 3;
    }
}
```

ReadCertificates (GET_CERTIFICATE)

This function reads the certificate chain on the provided slot ID starting from the provided offset and reading provided length bytes.

If the provided offset exceeds `0x600` then that indicates the power transmitter to offset from the Product Unit Certificate. Otherwise the offset starts from the beginning of the certificate chain.

```
void ReadCertificates(const uint8_t *pGetCertificateRequest,
    const size_t getCertificateRequestLen,
    uint8_t *pCertificateResponse,
    size_t *pCertificateResponseLen)
{
    smStatus_t retStatus        = SM_NOT_OK;
    uint16_t objectSize         = 0;
    pSe05xSession_t session_ctx = pgSe05xSessionctx;
    uint32_t certChainId        = 0;
    size_t readSize             = 0;
    uint8_t *pData              = NULL;
```

(continues on next page)

(continued from previous page)

```
    qi_error_code_t errorCode   = kQiErrorUnspecified;
    uint8_t authMsgHeader       = 0;
    uint8_t requestedslots      = 0;
    uint8_t certificateOffsetA8 = 0;
    uint8_t certificateOffset70 = 0;
    uint8_t certificateLengthA8 = 0;
    uint8_t certificateLength70 = 0;
    uint16_t certificateOffset  = 0;
    uint16_t certificatelength  = 0;
    /* Offset first DIGEST bytes to skip certificate chain hash */
    uint16_t offset      = DIGEST_SIZE_BYTES;
    uint16_t N_MC        = 0;
    uint16_t bytesToRead = 0;

    if (NULL == pGetCertificateRequest || NULL == pCertificateResponse) {
        LOG_E("Null buffer");
        errorCode = kQiErrorInvalidRequest;
        goto error;
    }

    authMsgHeader = pGetCertificateRequest[0];

    if (getCertificateRequestLen != GET_CERTIFICATE_CMD_LEN) {
        LOG_E("Invalid request length");
        errorCode = kQiErrorInvalidRequest;
        goto error;
    }

    requestedslots      = pGetCertificateRequest[1] & 0x03;
    certificateOffsetA8 = (pGetCertificateRequest[1] & 0xE0) >> 5;
    certificateOffset70 = pGetCertificateRequest[2];
    certificateLengthA8 = (pGetCertificateRequest[1] & 0x1C) >> 2;
    certificateLength70 = pGetCertificateRequest[3];
    certificateOffset   = certificateOffsetA8 * 256 + certificateOffset70;
    certificatelength   = certificateLengthA8 * 256 + certificateLength70;

    certChainId = QI_SLOT_ID_TO_CERT_ID(requestedslots);

    retStatus = Se05x_API_ReadSize(session_ctx, certChainId, &objectSize);
    if (retStatus != SM_OK) {
        errorCode = kQiErrorUnspecified;
        LOG_E("Se05x_API_ReadSize failed");
        goto error;
    }

    /* Read length of manufacturer certificate to determine the offset for PUC */
    retStatus = getManufacturerCertificateLength(session_ctx, certChainId, &N_MC);
    if (retStatus != SM_OK) {
        LOG_E("Failed to read manufacturer certificate length");
        errorCode = kQiErrorUnspecified;
        goto error;
```

(continues on next page)

(continued from previous page)

```
    }

    if (certificateOffset >= MAXIMUM_CERT_OFFSET) {
        LOG_D("Read PUC");
        /* N_RH is length of root Hash Certificate and
         * N_MC is length of Manufacturer Certificate
         */
        offset += 2 /* Length of length field */
                  /* Length of RootHash */
                  + DIGEST_SIZE_BYTES
                  /* Length of Manufacturer certificate */
                  + N_MC
                  /* Offset from Product Unit Certificate */
                  + certificateOffset - MAXIMUM_CERT_OFFSET;
    }
    else {
        offset += certificateOffset;
    }

    /* Calculate actual bytes to read */
    if (certificatelength == 0) {
        bytesToRead = objectSize - offset;
    }
    else {
        bytesToRead = certificatelength;
    }

    if (bytesToRead == 0) {
        /* Cannot read 0 bytes */
        errorCode = kQiErrorInvalidRequest;
        goto error;
    }

    /* Bytes to read cannot exceed the total object size */
    if ((offset + bytesToRead) > objectSize) {
        errorCode = kQiErrorInvalidRequest;
        goto error;
    }

    readSize = bytesToRead;
    pData    = (uint8_t *)SSS_MALLOC(bytesToRead * sizeof(uint8_t));
    if (!pData) {
        errorCode = kQiErrorUnspecified;
        goto error;
    }

    if (*pCertificateResponseLen < (size_t)(bytesToRead + 1)) {
        LOG_E("Insufficient buffer");
        errorCode = kQiErrorUnspecified;
        goto error;
    }
```

(continues on next page)

(continued from previous page)

```
    /* Read certificate chain */
    retStatus = readCertificateChain(session_ctx, certChainId, offset, bytesToRead,␣
↪pData, &readSize);
    if (retStatus != SM_OK) {
        errorCode = kQiErrorUnspecified;
        LOG_E("Se05x_API_ReadObject failed");
        goto error;
    }
    LOG_MAU8_D("ReadCertificate object", pData, readSize);

    /* Copy the data read out to response buffer */
    memcpy(&pCertificateResponse[1], pData, readSize);
    if (pData) {
        SSS_FREE(pData);
    }

    pCertificateResponse[0]  = (uint8_t)(authMsgHeader & 0xF0) + (uint8_
↪t)(kQiResponseCertificate);
    *pCertificateResponseLen = (bytesToRead) + 1;

    return;

error:
    if (pData) {
        SSS_FREE(pData);
    }
    if (pCertificateResponse) {
        pCertificateResponse[0]  = (uint8_t)(authMsgHeader & 0xF0) + (uint8_
↪t)(kQiResponseError);
        pCertificateResponse[1]  = (uint8_t)(errorCode);
        pCertificateResponse[2]  = 0x00;
        *pCertificateResponseLen = 3;
    }
}
```

Authenticate (CHALLENGE)

This function performs the CHALLENGE operation and returns the signature R and signature S values to the power receiver.

```
void Authenticate(const uint8_t *pChallengeRequest,
    const size_t challengeRequestLen,
    uint8_t *pChallengeAuthResponse,
    size_t *pChallengeAuthResponseLen)
{
    smStatus_t retStatus                     = SM_NOT_OK;
    uint16_t objectSize                      = 0;
    pSe05xSession_t session_ctx              = pgSe05xSessionctx;
    uint8_t *pTbsAuthPtr                     = NULL;
    size_t readSize                          = 0;
    uint8_t certChainHash[DIGEST_SIZE_BYTES] = {0};
```

(continues on next page)

(continued from previous page)

```
    uint8_t hash[DIGEST_SIZE_BYTES]          = {0};
    size_t hashLen                           = sizeof(hash);

    uint8_t authMsgHeader                = 0;
    uint8_t requestedSlot                = 0;
    uint8_t slotsPopulated               = 0x00;
    uint8_t tbsAuth[TBSAUTH_MAX_SIZE]    = {0};
    size_t tbsAuthlen                    = sizeof(tbsAuth);
    uint8_t signature[MAX_SIGNATURE_LEN] = {0};
    size_t sigLen                        = sizeof(signature);
    qi_error_code_t errorCode            = kQiErrorUnspecified;
    uint32_t certChainId                 = 0;
    uint32_t keyId                       = 0;

    if (NULL == pChallengeRequest || NULL == pChallengeAuthResponse) {
        LOG_E("Null buffer");
        errorCode = kQiErrorInvalidRequest;
        goto error;
    }

    authMsgHeader = pChallengeRequest[0];

    if (challengeRequestLen != CHALLENGE_CMD_LEN) {
        LOG_E("Invalid request length");
        errorCode = kQiErrorInvalidRequest;
        goto error;
    }

    requestedSlot = pChallengeRequest[1] & 0x03;
    certChainId   = QI_SLOT_ID_TO_CERT_ID(requestedSlot);
    keyId         = QI_SLOT_ID_TO_KEY_ID(requestedSlot);

    if (*pChallengeAuthResponseLen < CHALLENGE_AUTH_RESPONSE_LEN) {
        LOG_E("Insufficient buffer");
        errorCode = kQiErrorUnspecified;
        goto error;
    }

    retStatus = getPopulatedSlots(session_ctx, &slotsPopulated);
    if (SM_OK != retStatus) {
        errorCode = kQiErrorUnspecified;
        LOG_E("Failed to retrieve populated slots");
        goto error;
    }

    if (!(slotsPopulated & 0x01)) {
        /* Slot 0 is empty */
        errorCode = kQiErrorUnspecified;
        goto error;
    }
    /* Check if the requested slot is populated */
    if (!((1 << requestedSlot) & slotsPopulated)) {
```

(continues on next page)

(continued from previous page)

```
        errorCode = kQiErrorInvalidRequest;
        LOG_E("Requested slot not populated");
        goto error;
    }

    retStatus = Se05x_API_ReadSize(session_ctx, certChainId, &objectSize);
    if (retStatus != SM_OK) {
        errorCode = kQiErrorUnspecified;
        LOG_E("Se05x_API_ReadSize failed");
        goto error;
    }

    /* Certificate chain size cannot be less than DIGEST_SIZE_BYTES */
    if (objectSize < DIGEST_SIZE_BYTES) {
        errorCode = kQiErrorUnspecified;
        goto error;
    }

    /* Read Certificate chain hash value */
    pTbsAuthPtr = &tbsAuth[1];
    readSize    = DIGEST_SIZE_BYTES;
    retStatus   = Se05x_API_ReadObject(session_ctx, certChainId, 0, DIGEST_SIZE_BYTES,␣
↪certChainHash, &readSize);
    if (retStatus != SM_OK) {
        errorCode = kQiErrorUnspecified;
        LOG_E("Se05x_API_ReadObject failed");
        goto error;
    }
    memcpy(pTbsAuthPtr, certChainHash, DIGEST_SIZE_BYTES);

    /* Copy Challenge request to TBS Auth */
    pTbsAuthPtr = &tbsAuth[TBSAUTH_CHALLENGE_REQ_OFFSET];
    memcpy(pTbsAuthPtr, pChallengeRequest, challengeRequestLen);

    pChallengeAuthResponse[0] = (uint8_t)(authMsgHeader & 0xF0) + (uint8_
↪t)(kQiResponseChallengeAuth);
    pChallengeAuthResponse[1] = (uint8_t)(AUTH_PROTOCOL_VERSION << 4) + (uint8_
↪t)(slotsPopulated);
    pChallengeAuthResponse[2] = (uint8_t)(certChainHash[DIGEST_SIZE_BYTES - 1]);

    tbsAuth[0] = CHALLENGE_AUTH_RESPONSE_PREFIX; // ASCII representation of A
    memcpy(&tbsAuth[TBSAUTH_CHALLENGE_AUTH_RESP_OFFSET], pChallengeAuthResponse, 3);

    /* Calculate SHA256 of TBSAuth for signature */
    retStatus = getSha256Hash(session_ctx, tbsAuth, tbsAuthlen, hash, &hashLen);
    if (retStatus != SM_OK) {
        errorCode = kQiErrorUnspecified;
        LOG_E("Failed getSha256Hash");
        goto error;
    }

    /* Calculate signature */
```

(continues on next page)

(continued from previous page)

```
    retStatus =
        Se05x_API_ECDSASign(session_ctx, keyId, kSE05x_ECSignatureAlgo_SHA_256, hash,␣
↪hashLen, &signature[0], &sigLen);
    if (retStatus != SM_OK) {
        LOG_E(" sss_asymmetric_sign_digest Failed...");
        errorCode = kQiErrorUnspecified;
        goto error;
    }

    /* Extract R and S values from signature */
    retStatus = EcSignatureToRandS(signature, &sigLen);
    if (retStatus != SM_OK) {
        LOG_E(" EcSignatureToRandS Failed...");
        errorCode = kQiErrorUnspecified;
        goto error;
    }
    *pChallengeAuthResponseLen = (sigLen) + 3;
    memcpy(&pChallengeAuthResponse[3], signature, sigLen);

    return;

error:
    if (pChallengeAuthResponse) {
        pChallengeAuthResponse[0]  = (uint8_t)(authMsgHeader & 0xF0) + (uint8_
↪t)(kQiResponseError);
        pChallengeAuthResponse[1]  = (uint8_t)(errorCode);
        pChallengeAuthResponse[2]  = 0x00;
        *pChallengeAuthResponseLen = 3;
    }
}
```

Building

Build the example `examples/se05x_qi_auth/sa_qi_auth` for `lpcxpresso55s69_cpu0` with Zephyr build system. See `Getting Started with Zephyr + SE05x` for details on how to build with Zephyr.

Porting

The example can be built for different controllers from zephyr build system. If you want to build for a different OS, you need to do these steps:

- Add platform specific port files in `lib/platform/<board-name>` and include it in CMake for compilation. You would need to add timer, I2C and board port files. For reference see `lib/platform/k64`.
- Add host crypto port for Qi authentication example in `examples/se05x_qi_auth/sa_qi_auth/port` to enable host verification operations.

```
/* Port to implement RNG to get 16 byte nonce value
 * for authentication operation.
 * This API does not guarantee the randomness of the RNG.
 * User should make sure that the RNG seed is from a trusted source
```

(continues on next page)

(continued from previous page)

```
 * and that the randomness of the source is NIST compliant
 */
int port_getRandomNonce(uint8_t *nonce, size_t *pNonceLen)
{
    int ret             = 0;
    size_t random_length = (*pNonceLen > NONCE_LEN) ? NONCE_LEN : (*pNonceLen);
    *pNonceLen          = random_length;
    ret                 = getRandom(nonce, random_length);
    if (0 != ret) {
        *pNonceLen = 0;
    }
    return ret;
}

/* Port to implement function which will
 * parse an X.509 certificate and extract the public key
 * from it.
 */
void port_parseCertGetPublicKey(uint8_t *pCert, size_t certLen, uint8_t *pPublicKey,
↪ size_t *publicKeylen)
{
    parseCertGetPublicKey(pCert, certLen, pPublicKey, publicKeylen);
}

/* Port to implement function which will
 * verify the complete certificate chain as passed in certificate_chain
 */
int port_hostVerifyCertificateChain(uint8_t *certificate_chain,
    size_t certificate_chain_size,
    uint16_t pucCertOffset,
    uint16_t manufacturerCertLenOffset)
{
    return hostVerifyCertificateChain(
        certificate_chain, certificate_chain_size, pucCertOffset,␣
↪manufacturerCertLenOffset);
}

/* Port to implement function which will
 * verify CHALLENGE on host
 */
int port_hostVerifyChallenge(uint8_t *pPublicKey,
    size_t publicKeyLen,
    uint8_t *pCertificateChainHash,
    uint8_t *pChallengeRequest,
    uint8_t *pChallengeResponse)
{
    return hostVerifyChallenge(pPublicKey, publicKeyLen, pCertificateChainHash,␣
↪pChallengeRequest, pChallengeResponse);
}
```

5.6.2 Secure Authenticator (Qi) Provisioning demo

This project is used to provision Qi credentials (ECDSA Key pair and Device certificate chain) inside the secure element.

> **Warning:** This example is only for demonstration purpose. Maintaining and provisioning the credentials should be done in a secure way.

The user should update the credentials `qi_ec_priv_key` and `qi_certificate_chain` in `examples/se05x_qi_auth/sa_qi_provisioning/sa_qi_credentials.c`

By default the demo will provision the credentials for Slot ID 0. The user can update the macro `QI_PROVISIONING_SLOT_ID` in `examples/se05x_qi_auth/sa_qi_provisioning/sa_qi_provisioning.h` to provision for a different slot:

```
/* Update the SLOT_ID to provision for another slot
 * Valid values are 0, 1, 2, 3
 */
#define QI_PROVISIONING_SLOT_ID 0
```

This demo requires the credentials to be provisioned using a management credential. In this example we use the demo key provisioned at `EX_MANAGEMENT_CREDENTIAL_ID` to open an AESKey session and provision the credentials. The user is expected to provision their own authentication key and use that for provisioning the Qi credentials by updating the auth object ID in `examples/se05x_qi_auth/sa_qi_provisioning/sa_qi_provisioning.c`:

```
#define EX_MANAGEMENT_CREDENTIAL_ID 0x7DA00002
uint8_t aes_scp03_enc_key[16] = {
    0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D,␣
↪0x4E, 0x4F};
uint8_t aes_scp03_mac_key[16] = {
    0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D,␣
↪0x4E, 0x4F};
uint8_t aes_scp03_dek_key[16] = {
    0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D,␣
↪0x4E, 0x4F};
```

Pre-requisites

- AESKey must be provisioned on the Secure Element before running this demo

Building the Demo

Build the example `examples/se05x_qi_auth/sa_qi_provisioning` for `lpcxpresso55s69_cpu0` with Zephyr build system. See *Getting Started with Zephyr + SE05x* for details on how to build with Zephyr.