AN12570

EdgeLock SE05x Quick start guide with Raspberry Pi

Rev. 1.5 — 3 August 2022 Application note
565815

Document information

Information	Content
Keywords	EdgeLock SE05x, EdgeLock A5000, EdgeLock SE Plug & Trust Middleware
Abstract	This document explains how to get started with the SE05x/A5000 development board and the Raspberry Pi board, as a reference for any other device running a Linux distribution. This guide provides detailed instructions for connecting the boards and running the project examples included in EdgeLock SE Plug & Trust Middleware.

Revision history

Revision history

Revision number	Date	Description
1.0	2019-08-30	First document release.
1.1	2020-02-06	Added OM-SE050RPI adapter board.
1.2	2020-12-07	Updated to latest template and fixed broken links.
1.3	2021-01-22	Added EdgeLock SE051, terminal Figure changes and appendix addition to show the ssscli command line interface.
1.4	2022-03-28	Add EdgeLock SE050E and EdgeLock A5000 product variants. Update Table 1, Figure 1, Figure 2, Figure 3, Add note (step 2) in Section 3.3 Build EdgeLock SE Plug & Trust Middleware examples. Add Section Section 4 Product specific CMake build settings. Add Section Section 5 Binding EdgeLock SE05x to a host using Platform SCP. Add Section Section 6 Manage access from multiple Linux processes to the EdgeLock SE05x.
1.5	2022-08-03	Clarify to install python 3 in Section 2.2.2. Update to EdgeLock SE Plug & Trust Middleware version 04.02.xx. Update note (step 2) in Section 3.3 Build EdgeLock SE Plug & Trust Middleware examples. Update Section Section 4 Product specific CMake build settings. Update Section Section 5 Binding EdgeLock SE05x to a host using Platform SCP.

1 Required hardware

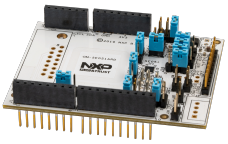
The EdgeLock SE05x works as an auxiliary security device attached to a host controller, communicating with through an I²C interface. To follow the instructions provided in this document, you need an EdgeLock SE05x development board and a Raspberry Pi board, acting as a host controller.

1.1 Required hardware

The following hardware will be used throughout the document:

1. **EdgeLock SE05x development boards ordering details**
   The EdgeLock SE05x and EdgeLock A5000 product support packages are providing development boards for evaluating EdgeLock SE05x and EdgeLock A5000 features. Select the development board of the product you want to evaluate. Table 1 details the ordering details of the EdgeLock SE05x and EdgeLock A5000 development boards.

Table 1. EdgeLock SE05x development boards.

Part number	12NC	Description	Picture
OM-SE050ARD-E	9354 332 66598	SE050E Arduino® compatible development kit	
OM-SE050ARD-F	9354 357 63598	SE050 Arduino® compatible development kit	
OM-SE050ARD	9353 832 82598	SE050F Arduino® compatible development kit	
OM-SE051ARD	9353 991 87598	SE051 Arduino® compatible development kit	
OM-A5000ARD	9354 243 19598	A5000 Arduino® compatible development kit	

Note: The pictures in this guide will show EdgeLock SE05xE, but all boards in Table 1 can be used as well with the same hardware configuration.

2. **OM-SE050RPI adapter board for Raspberry Pi:**

Table 2. OM-SE050RPI adapter board details

Part number	12NC	Content	Picture
OM-SE050RPI	935398642598	Raspberry Pi to OM-SE05xARD adapter	

3. **Raspberry Pi board:**

Table 3. Raspberry Pi

Part number	Content	Picture
Raspberry Pi	Any Raspberry Pi model	

2 Prepare your Raspberry Pi

This section explains how to get your Raspberry Pi ready to execute the EdgeLock SE Plug & Trust Middleware. For that, you need to go through the following steps:

1. Hardware setup for Raspberry Pi
2. Software setup for Raspberry Pi

2.1 Hardware setup

The hardware setup consists of two steps:

1. Configuring the OM-SE05xARD jumpers, as described in Section 2.1.1.
2. Connecting the OM-SE05xARD to the Raspberry Pi, as described in Section 2.1.2.

2.1.1 Jumper configuration

Make sure the jumpers in your OM-SE05xARD board are configured as shown in Figure 1:

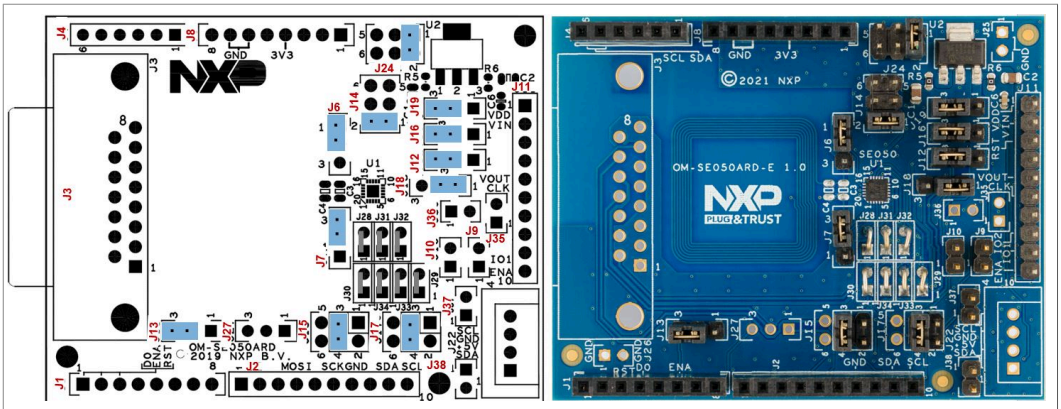


Figure 1. OM-SE05xARD jumper configuration

***Note**: For more information about the jumper settings, refer to AN13539 OM-SE05xARD hardware overview.*

2.1.2 Connecting the OM-SE05xARD to the Raspberry Pi

You have two options to connect the Raspberry Pi to the OM-SE05xARD board:

1. Using the OM-SE05xRPI adapter board, as described in Section 2.1.2.1
2. Using the OM-SE05xARD connected with wires, as described in Section 2.1.2.2

2.1.2.1 Using the OM-SE05xRPI adapter board

The Raspberry Pi and the OM-SE05xARD boards can be directly connected using the OM-SE050RPI adapter board. Follow the steps shown in Figure 2:

1. Mount the OM-SE05xARD on top of the OM-SE05xRPI board using the Arduino connectors.
2. Mount the two boards on top of the Raspberry Pi using the Raspberry connectors in the OM-SE05xRPI.

The result of it is three boards stacked together, being the OM-SE05xRPI the board in between the Raspberry Pi and OM-SE05xARD.

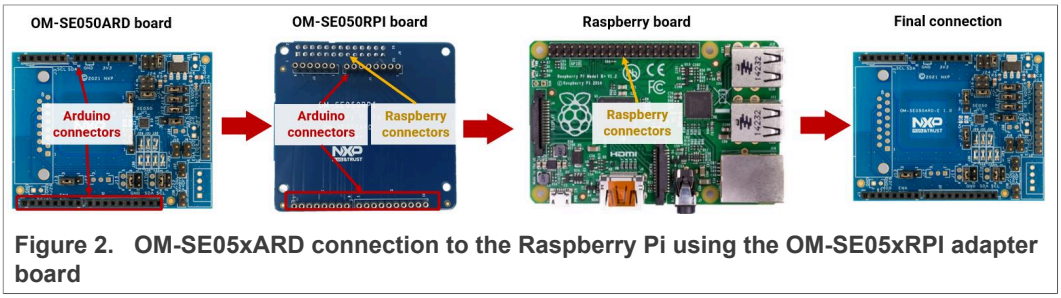


Figure 2. OM-SE05xARD connection to the Raspberry Pi using the OM-SE05xRPI adapter board

2.1.2.2 Connecting the OM-SE05xARD with wires

In case you do not have the OM-SE05xRPI adapter board, you can also manually wire the Raspberry Pi to the OM-SE05xARD using the I^2C connector, as shown in Figure 3:

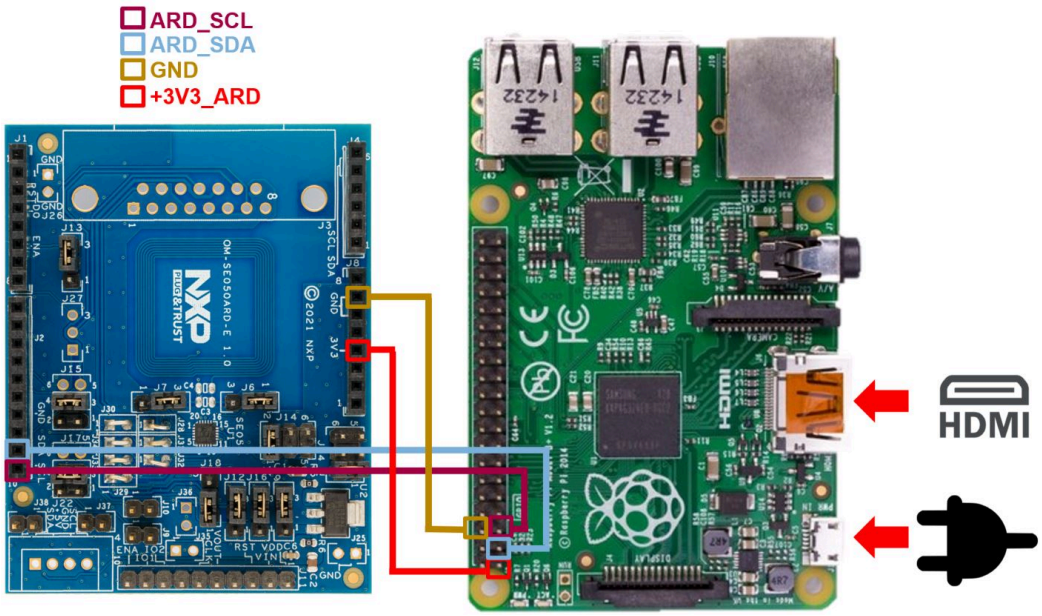


Figure 3. OM-SE05xARD wiring to the Raspberry Pi board

Table 4 shows the detailed connection of the OM-SE05xARD to the Raspberry Pi:

Table 4. OM-SE05xARD wiring to the Raspberry Pi board

OM-SE05xARD (# jumper - # pin)	Raspberry Pi (# jumper - # pin)
J2-P10 (ARD_SCL)	J8-P5 (SCL)
J2-P9 (ARD_SDA)	J8-P3 (SDA)
J8-P7 (GND)	J8-P6 (GND)
J8-P4 (3V3_ARD)	J8-P1 (3V3)

2.2 Software setup

The software setup consists of three steps:

1. Install your preferred Linux distribution in your device. In this guide the Raspberry Pi board running the Raspbian operating system is used as a reference. Raspbian can be installed as described in Section 2.2.1.
2. Install the build tools necessary to build the EdgeLock SE Plug & Trust Middleware and the test project examples. The procedure for the Raspbian operating system is described in Section 2.2.2.
3. Enable the I^2C interface in your Linux distribution to allow the communication with the security IC of the OM-SE05xARD board. The procedure for the Raspbian operating system is described in Section 2.2.3.

2.2.1 Install Raspbian

Before executing the steps described in this guide, it is necessary to install the Raspbian operating system in the Raspberry Pi. The official Raspberry website recommends two options:

1. Using New Out of Box Software (NOOBS), an easy operating system installation manager for the Raspberry Pi. This tool is the easiest and most recommended option, but requires a screen to go through the initial installation process. Installation instructions are provided in the official Raspberry NOOBS webpage.
2. Downloading the official Raspbian image from the official Raspberry Pi image repository and then flashing the image in the SD card by following the instructions provided in the official documentation.

The steps described in this guide use the latest Raspbian release at the time of writing (Raspbian 10 Buster).

2.2.2 Install build tools

To build the EdgeLock SE Plug & Trust Middleware middleware and the example projects, it is necessary to have the Python and CMake packages installed in the system along with the libssl library (part of OpenSSL toolkit). CMake GUI packages are also required if you want to use the CMake graphical user interface. You can install the required packages by opening a Terminal window and following the steps as shown in Figure 4:

1. You can install all the required packages with a single command by sending:
   `>> sudo apt-get install python3 cmake cmake-curses-gui cmake-qt-gui libssl-dev`
2. You may be asked to proceed with the installation:
   Send `>> y`

```
pi@raspberrypi:~ $ sudo apt-get install python3 cmake cmake-curses-gui cmake-qt-gui libssl-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.9.2-3).
python3 set to manually installed.
The following additional packages will be installed:
  cmake-data libjsoncpp24 librhash0
Suggested packages:
  cmake-doc ninja-build libssl-doc
The following NEW packages will be installed:
  cmake cmake-curses-gui cmake-data cmake-qt-gui libjsoncpp24 librhash0 libssl-dev
0 upgraded, 7 newly installed, 0 to remove and 40 not upgraded.
Need to get 11.1 MB of archives.
After this operation, 50.2 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Figure 4. Install build tools

2.2.3 Enable the I²C interface

The Raspberry Pi board communicates with the OM-SE05xARD security IC through the I²C interface. The I²C interface is not enabled by default in Raspbian and must be activated before the EdgeLock SE Plug & Trust Middleware test examples can be executed. To enable I²C, open a Terminal window and follow these steps:

1. Verify if I²C is active by listing the available I²C interfaces:
   `>> ls /sys/bus/i2c/devices/`
   If the *i2c-x* interface is listed, as shown in Figure 5 , then you can skip this section and proceed to Section 3.
   Note: *the I²C interface number might be different.*

Figure 5. List I²C interfaces

2. Open the Raspberry Pi software configuration tool, as shown in Figure 6:
   `>> sudo raspi-config`

Figure 6. Open the Raspberry Pi software configuration tool

3. Use the up and down arrow keys to select the 5th menu entry (Interfacing Options) and then press Enter, as shown in Figure 7:

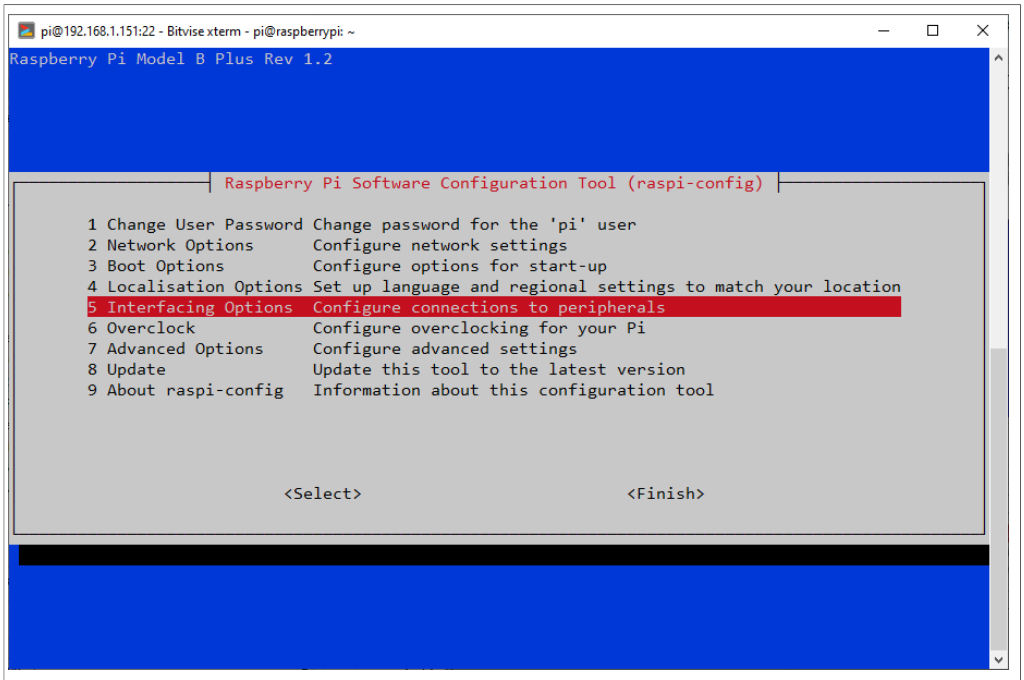


Figure 7. Enable I²C interface

4. Use the up and down arrow keys to select the 5th menu option (I^2C) and then press Enter, as shown in Figure 8:

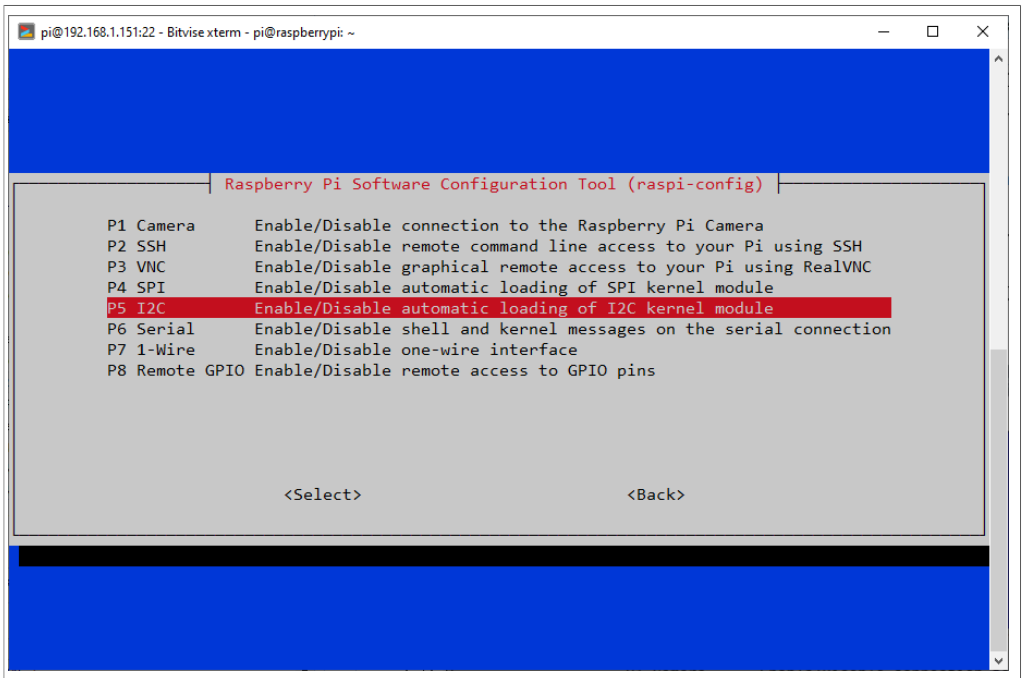


Figure 8. Enable I^2C interface

5. You will be asked to confirm your choice to activate the I^2C interface. Use the left and right arrow keys to select the Yes option and then press Enter, as shown in Figure 9:

Figure 9. Enable I^2C interface

6. Close the Raspberry Pi software configuration tool. Use the left and right arrow keys to select the Finish option and then press Enter, as shown in Figure 10:

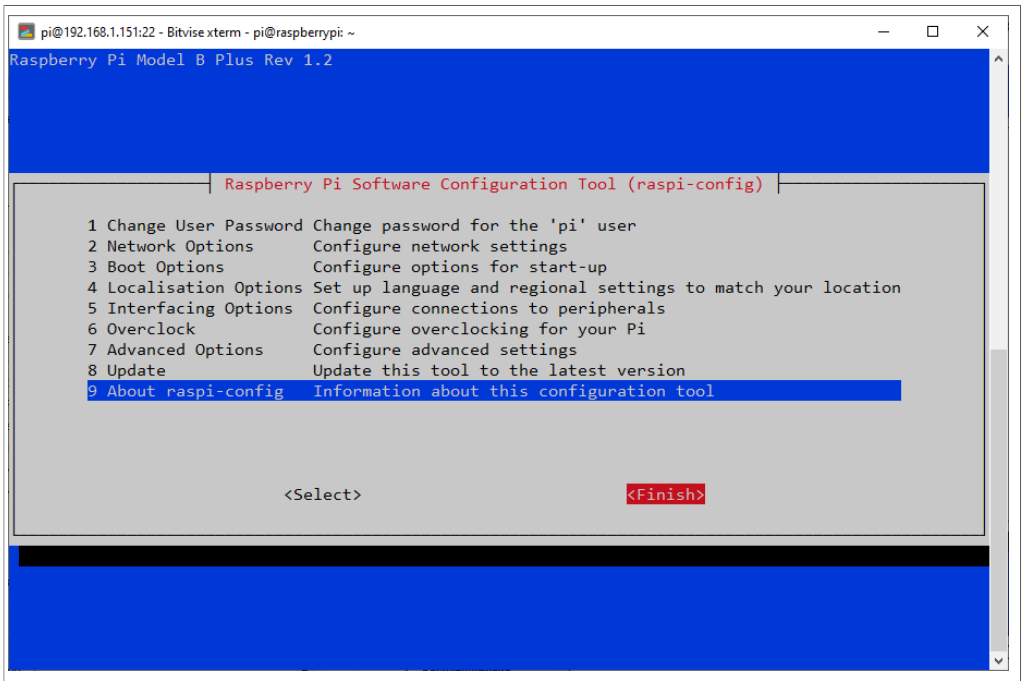


Figure 10. Close the Raspberry Pi sofware configuration tool

7. Verify the correct activation of the I^{2}C interface, as shown in Figure 11:
   `>> ls /sys/bus/i2c/devices/`
   The *i2c-x* interface should now be listed.
   Note: *the I^{2}C interface number might be different.*

Figure 11. List I^{2}C interfaces

3 Run EdgeLock SE Plug & Trust Middleware test examples

This section details the steps required from the moment you download EdgeLock SE Plug & Trust Middleware until you are able to run an EdgeLock SE Plug & Trust Middleware test example.

3.1 Download EdgeLock SE Plug & Trust Middleware

The EdgeLock SE Plug & Trust Middleware stack includes several project examples for cloud service onboarding. To prepare the EdgeLock SE Plug & Trust Middleware:

1. Download the EdgeLock SE Plug & Trust Middleware from NXP website and place the .zip file in the */home/user* directory of your Raspbian distribution.
   Note:*The user folder can have different names, in this example the user folder's name is pi*

2. Open a Terminal window and follow the next steps as shown in Figure 12:
   a. Move to the user's *home* directory:
      (1) `>> cd ~`
   b. Create a folder called *se050_middleware:*
      (2) `>> mkdir se_mw`
   c. Unzip the EdgeLock SE Plug & Trust Middleware in the *se050_middleware* folder:
      (3) `>> unzip SE-PLUG-TRUST-MW.zip -d se_mw`
      Note:*The name of the zip file might be different.*
      Note:*This command may take a few seconds to complete.*

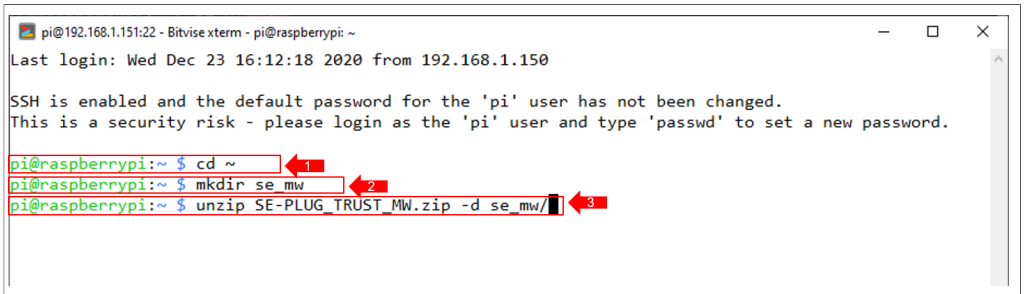


Figure 12. Create se050_middleware folder

3. You can verify that the files have been correctly unzipped by following these steps:
   a. Move to the *simw-top* folder inside the *se_mw* folder:
      `>> cd se_mw/simw-top`
   b. List the content of the *simw-top* folder:
      `>> ls`
      The content of the folder should be the same as shown in Figure 13:

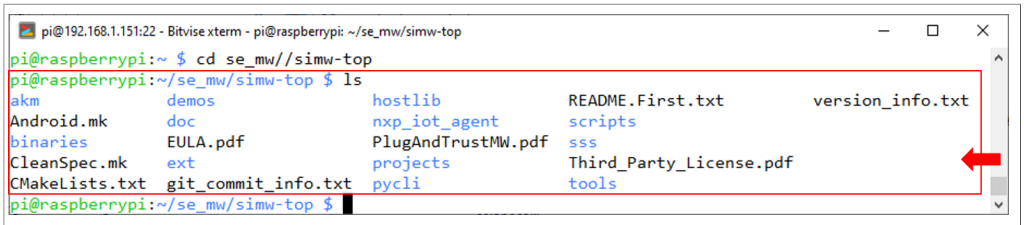


Figure 13. simw-top folder content

3.2 Build EdgeLock SE Plug & Trust Middleware

The EdgeLock SE Plug & Trust Middleware uses CMake for building the project examples into your local machine. To build the EdgeLock SE Plug & Trust Middleware middleware, open a Terminal window and follow the next steps as shown in Figure 14:

1. Go to the folder with the unzipped SE050 middleware:
   (1) `>> cd /home/pi/se_mw/simw-top/scripts`

2. Generate the EdgeLock SE Plug & Trust Middleware project examples:
   (2) >> `python create_cmake_projects.py rpi`
   Note: *This command may take a few seconds to complete. The last parameter rpi circumvents auto-detection of the host and enforces to build for Raspberry Pi.*

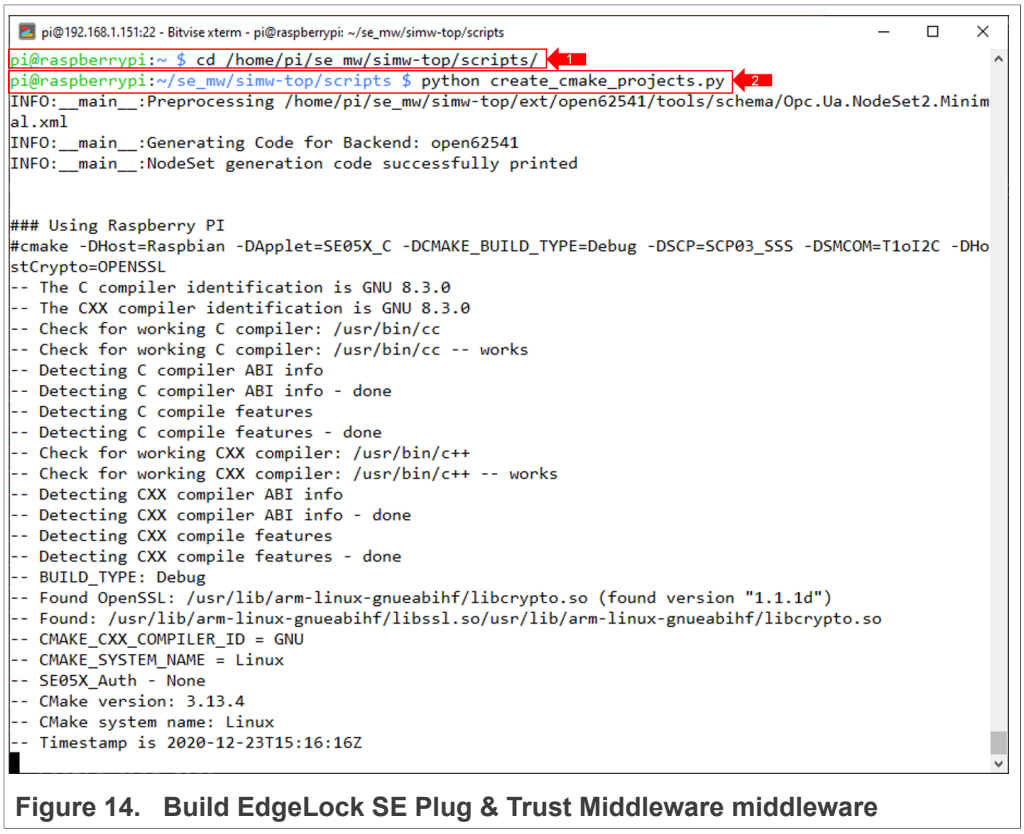


Figure 14. Build EdgeLock SE Plug & Trust Middleware middleware

3. If the compilation is successful you should (1) see a new *simw-top_build* folder inside the se_mw folder and (2) a new folder inside the `simw-top_build` folder as shown in Figure 15:

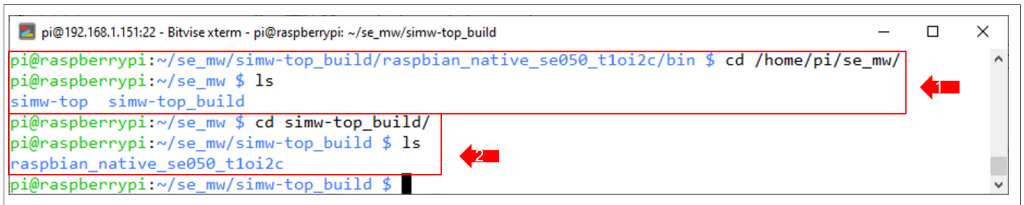


Figure 15. EdgeLock SE05x middleware project structure

3.3 Build EdgeLock SE Plug & Trust Middleware examples

The EdgeLock SE Plug & Trust Middleware contains several examples used to verify atomic EdgeLock SE05x security IC features. This section explains how to compile the EdgeLock SE Plug & Trust Middleware examples. Open a Terminal window and follow these steps:

1. Move to the folder that contains the examples and the source code of the Raspbian EdgeLock SE05x libraries:
   `>> cd /home/pi/se_mw/simw-top_build/raspbian_native_se050_t1oi2c`
2. **Note:** The default build configuration of the EdgeLock SE Plug & Trust Middleware ≥ `V04.02.0x` generates code for the OM-SE050ARD-E development board. You need to adapt the CMake settings in case you are using a different EdgeLock secure element development board or a different secure element product IC. The settings are described in Section 4 Product specific CMake build settings.
   Open the CMake configuration interface, as shown in Figure 16 to change build settings:
   `>> ccmake .`
   Note: *You can use the graphical interface by sending* `cmake-gui .` *instead.*

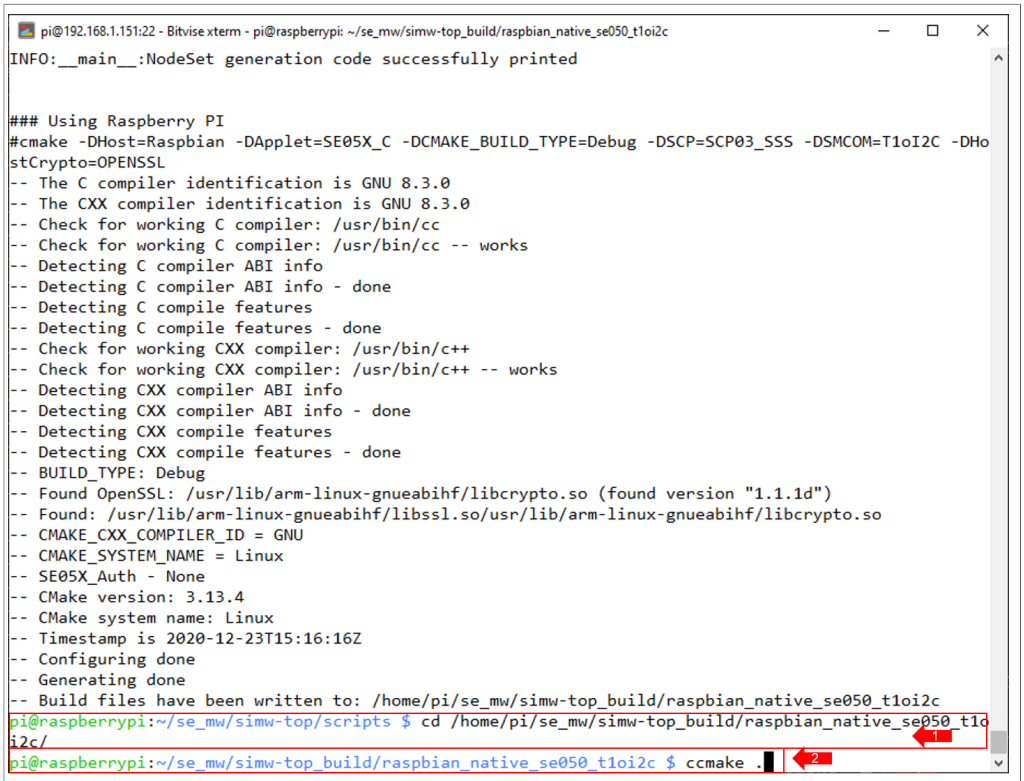


Figure 16. Open CMake configuration interface

3. Review the build configuration and make sure that the *Host* parameter is set to the value *Raspbian*, as shown in Figure 17. Leave the default settings and press *q* to return to the console.
   Note: *If you want to change the configuration you can use the up and down arrow keys to navigate through the available options and the left and right arrow keys to*

change the option value. In case you edit the configuration, press `c` *(configure) and then* `g` *(generate)* to apply the changes.

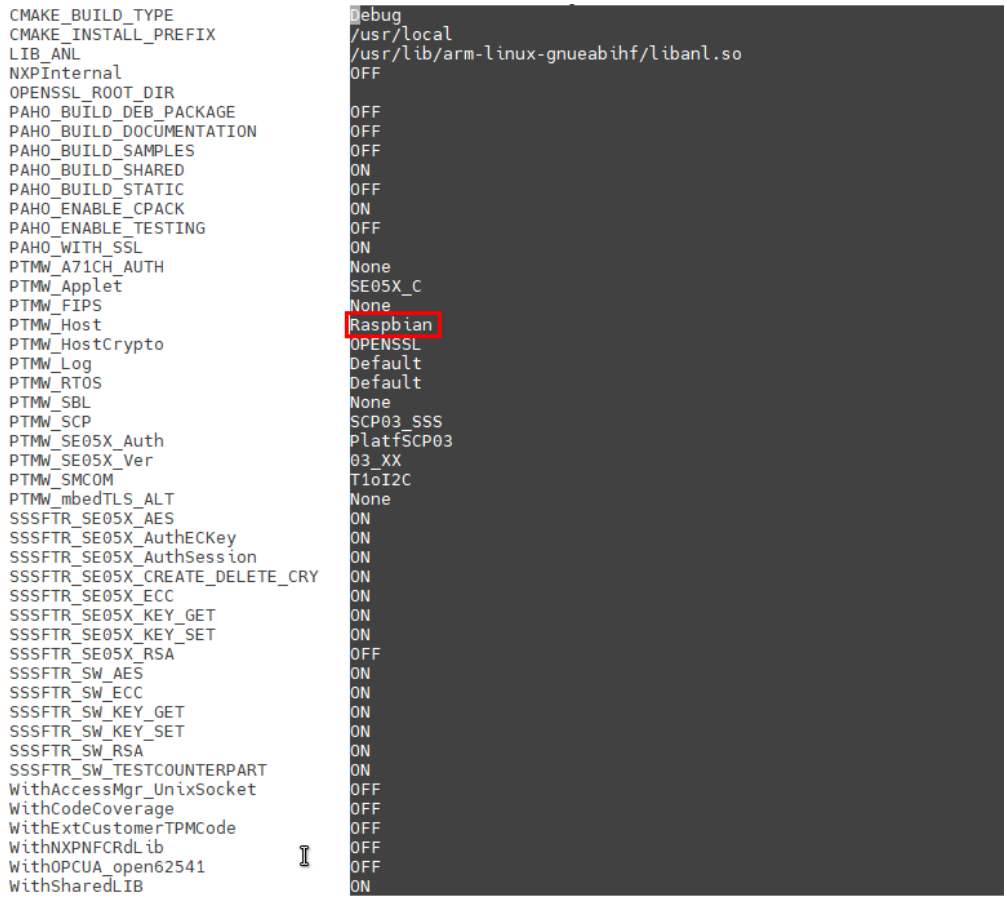


Figure 17. Review build configuration

4. Build the project examples, as shown in Figure 18:
   `>> cmake --build .`
   Note: *This command may take a few seconds to complete.*

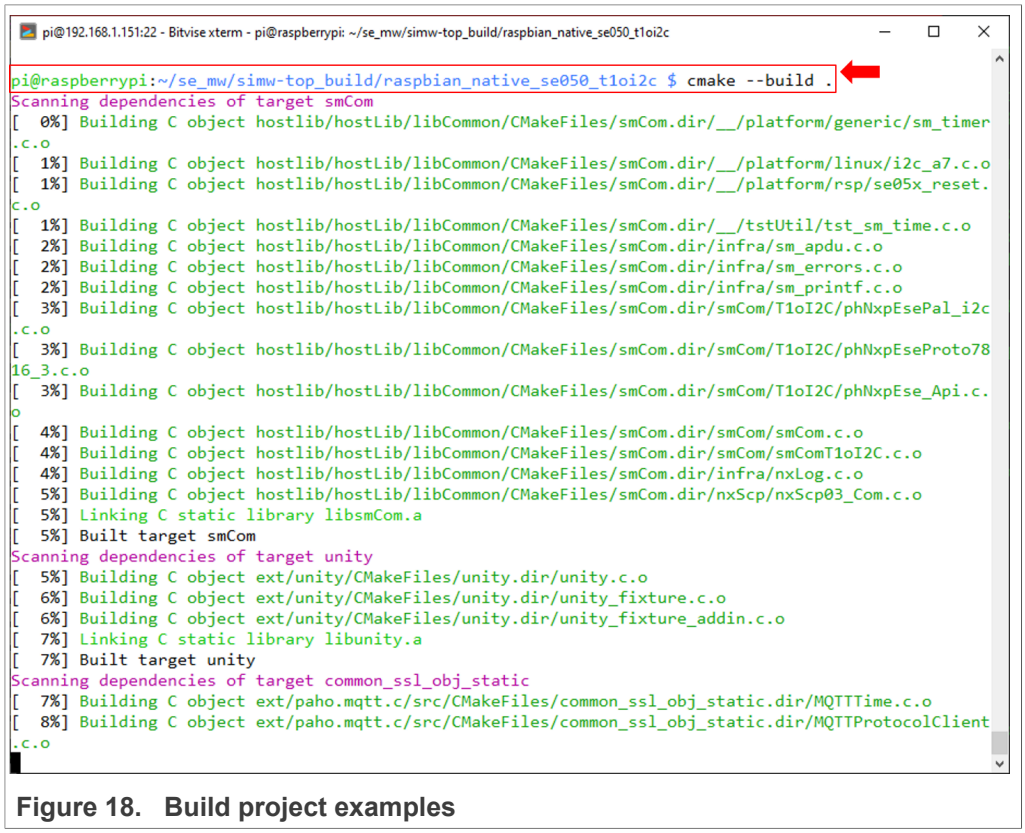


Figure 18. Build project examples

5. Install the projects in the system as shown in Figure 19:
   ```
   >> sudo make install
   ```
   Note: *This command may take a few seconds to complete.*

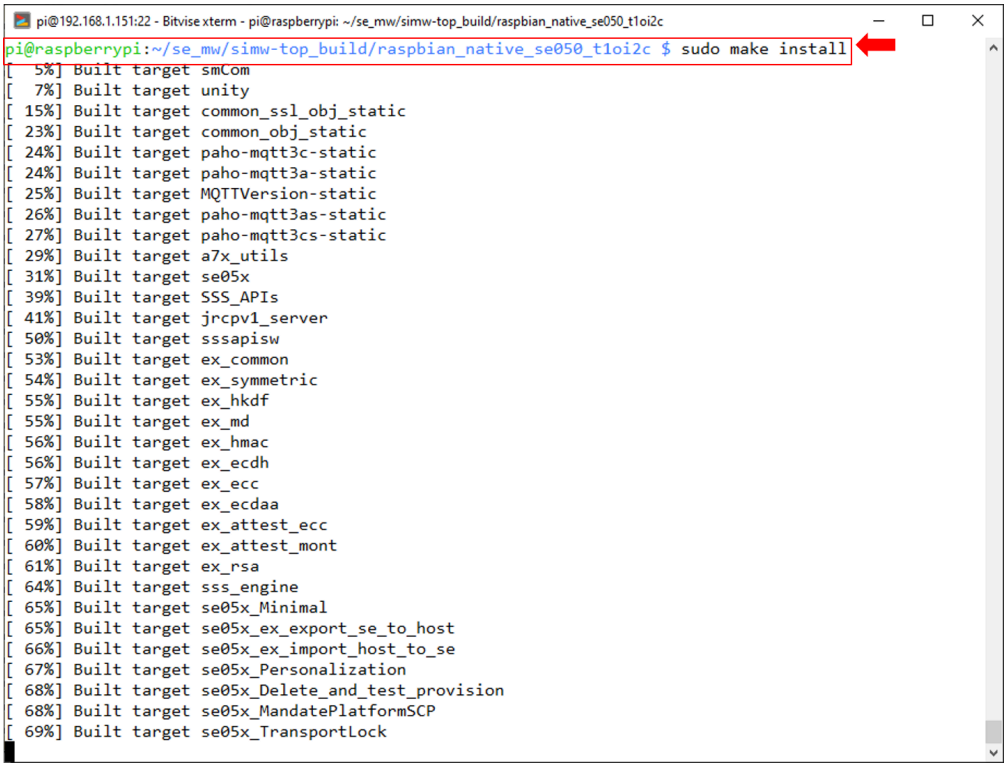


Figure 19. Install projects in the system

6. Update the cache to include the newly installed libraries as shown in Figure 20:
   `>> sudo ldconfig /usr/local/lib`

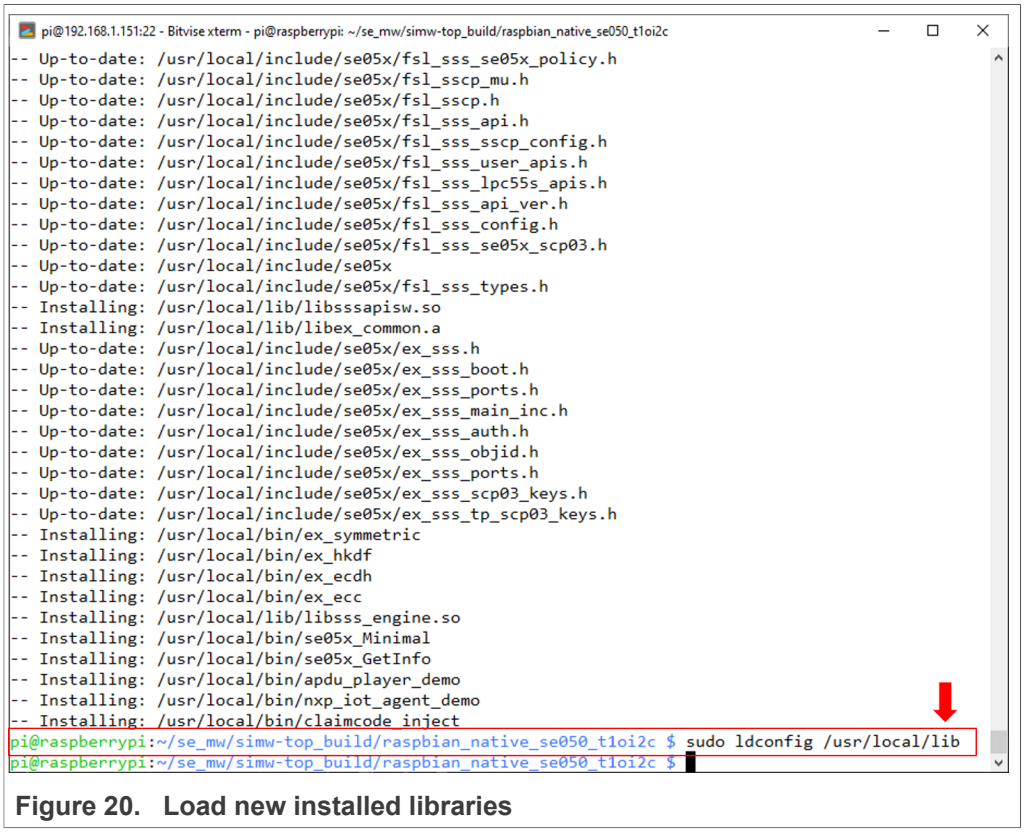


Figure 20. Load new installed libraries

3.4 Execute EdgeLock SE Plug & Trust Middleware test example

This section explains how to run the EdgeLock SE Plug & Trust Middleware test example called `se05x_minimal`. The `se05x_minimal` project outputs the memory left in the EdgeLock SE05x security IC. To execute the `se05x_minimal` test example follow these steps:

1. Connect the OM-SE05xARD board to the Raspberry Pi as described in Section 2.1.

2. Open a Terminal window and follow the steps as shown in Figure 21:
   a. Move to the directory containing the examples binaries:
      (1) >> cd /home/pi/se_mw/simw-top_build/raspbian_native_se050_t1oi2c/bin/
   b. Run the se05x_minimal example:
      (2) >> ./se05x_Minimal
      (3) You should see the EdgeLock SE05x IC available memory (in this case, 32767)

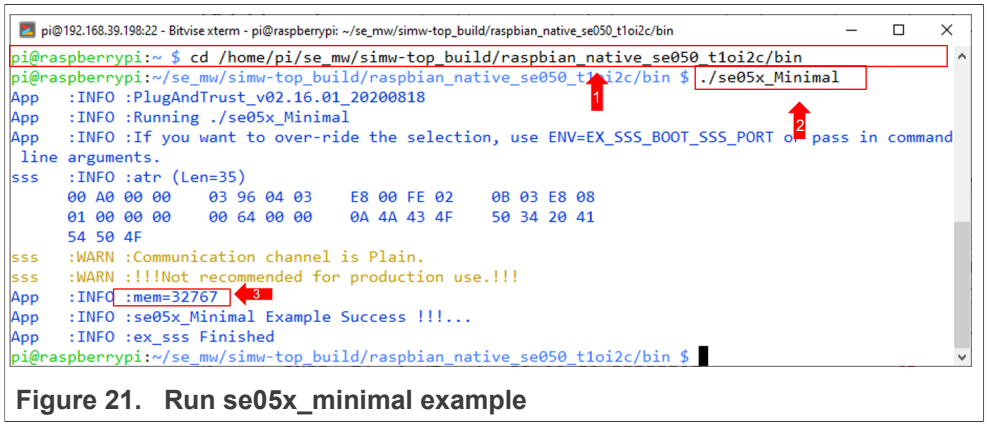


Figure 21. Run se05x_minimal example

4 Product specific CMake build settings

The NXP Plug & Trust middleware supports the SE05x Secure Elements, the A5000 Secure Authenticator, and the legacy A71CH products.

The EdgeLock Plug & Trust middleware is delivered with CMake files that include the set of directives and instructions describing the project's source files and the build targets. The CMake files are used to select a dedicated EdgeLock product IC and the corresponding IoT applet or Authenticator application.

The SE050 product identification can be obtained as described in AN12436 chapter 1 *Product Information*. AN12973 describes the same procedure for the SE051 product family.

The following tables show the required PTMW CMake options to build a dedicated product variant. The SSSFTR_ SE05X_RSA CMake option is used to optimize the memory footprint for product variants that do not support RSA.

Table 5. CMake Settings for SE050E product variants

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
SE050E Dev. Board OM-SE050ARD-E	A921	SE050_E	None	07_02	any option	None or SCP03_ SSS	disabled
SE050E2	A921						

Table 6. CMake Settings for SE050F product variants

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
SE050F Dev.Board OM-SE050ARD-F	A92A	SE05X_C	SE050	03_XX	PlatfSCP03 or UserID_PlatfSCP03 or AESKey_PlatfSCP03 or ECKey_PlatfSCP03	SCP03_ SSS	enabled
SE050F2	A92A						

Table 7. CMake Settings for SE050 Previous Generation product variants

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
SE050A1	A204	SE05X_A	None	03_XX	any option	None or SCP03_ SSS	disabled
SE050A2	A205						
SE050B1	A202	SE05X_B	None	03_XX	any option	None or SCP03_ SSS	enabled
SE050B2	A203						
SE050C1	A200	SE05X_C	None	03_XX	any option	None or SCP03_ SSS	enabled
SE050C2	A201						
SE050 Dev Board OM-SE050ARD	A1F4						
SE050F2	A77E[1]	SE05X_C	SE050	03_XX	PlatfSCP03 or UserID_PlatfSCP03 or AESKey_PlatfSCP03 or ECKey_PlatfSCP03	SCP03_ SSS	enabled

[1] All SE050F2 with variant A77E have date code in year 2021. All the SE050F2 with date code in the year 2022 have the variant identifier A92A.

Table 8. CMake Settings for SE051 product variants

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
SE051A2	A920	SE05X_A	None	07_02	any option	None or SCP03_ SSS	disabled

Table 8. CMake Settings for SE051 product variants...*continued*

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
SE051C2	A8FA	SE05X_C	None	07_02	any option	None or SCP03_ SSS	enabled
SE051W2	A739	SE05X_C	None	07_02	any option	None or SCP03_ SSS or SCP03_ SSS	enabled
SE051A2	A565	SE05X_A	None	06_00	any option	None or SCP03_ SSS	disabled
SE051C2	A564	SE05X_C	None	06_00	any option	None or SCP03_ SSS	enabled

Table 9. CMake Settings for A5000 product variants

Variant	OEF ID	PTMW_ Applet	PTMW_ FIPS	PTMW_ SE05X_ Ver	PTMW_SE05X_Auth	PTMW_ SCP	SSSFTR_ SE05X_ RSA
OM-A5000ARD	A736	AUTH	None	07_02	any option	None or SCP03_ SSS	disabled
A5000	A736						

4.1 Example: SE050E CMake build settings

To build the Plug & Trust Middleware to support the SE050E Secure Element applet the following CMake setting needs to be modified before building the middleware

according to Table 5:

- Select SE05X_E *for the CMake option* PTWM_Applet.
- Select None for the CMake option PTWM_FIPS.
- Select 07_02 *for the CMake option* PTWM_SE05X_Ver
- Disable the CMake option SSSFTR_SE05X_RSA

In this example we use plain communication. Plain communication for the example execution is enabled by selecting the following options:

- Select None for the CMake option PTMW_SE05X_Auth.
- Select None for the CMake option PTMW_SCP.

How to enable Platform SCP is described in How to enable Platform SCP in the CMake-based build system.

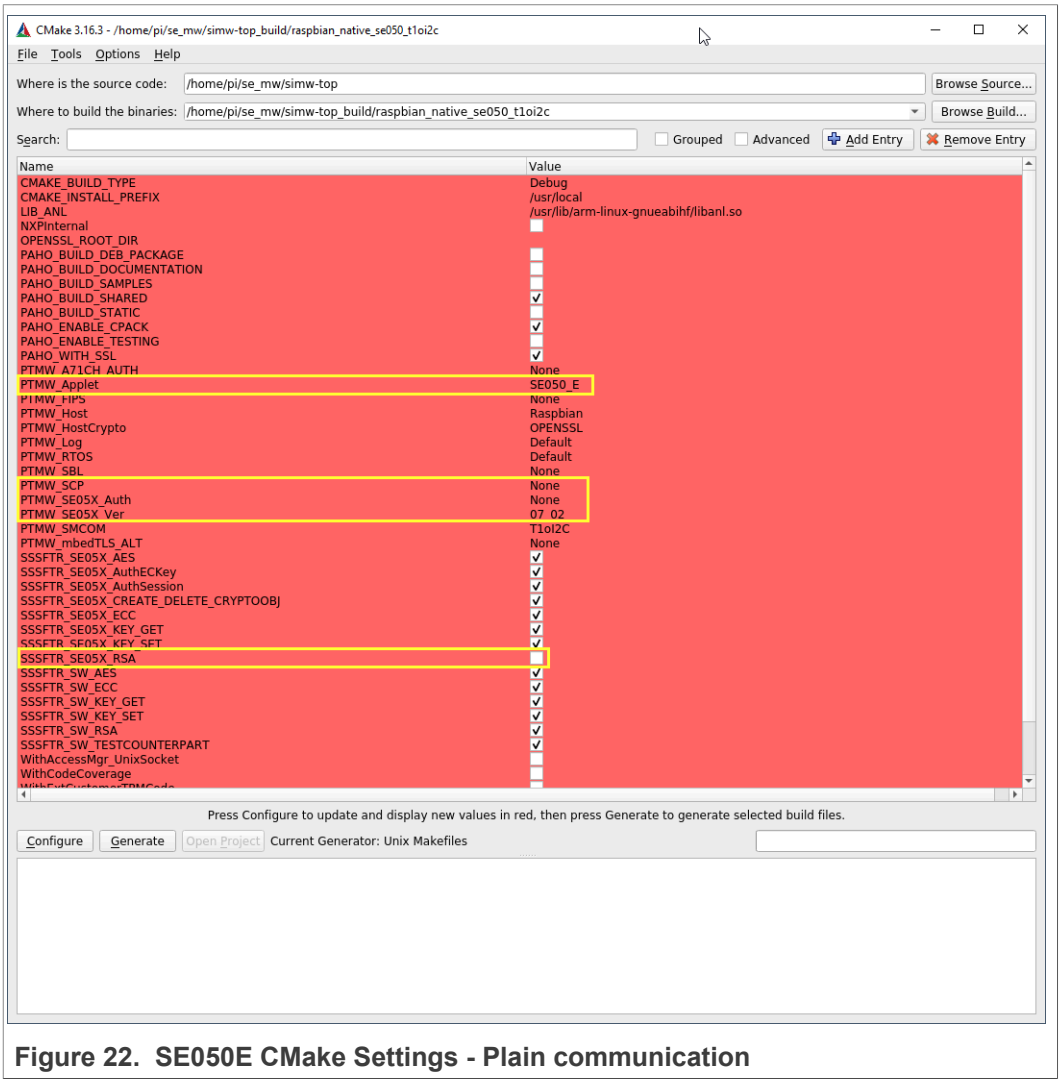


Figure 22. SE050E CMake Settings - Plain communication

Run the following commands to update the CMake settings and rebuild the EdgeLock SE Plug & Trust Middleware:

```
cd ~/se_mw/simw-top_build/raspbian_native_se050_t1oi2c
```

```
cmake-gui .
```

Note: *You can use the commandline interface by sending* `ccmake .` *instead (see also Section 3.3).*

Update the CMake settings as explained above. Press first the ***Configure*** button and second the ***Generate*** button and close the CMake GUI.

```
cmake --build .
```

```
sudo make install
```

```
sudo ldconfig /usr/local/lib/
```

5 Binding EdgeLock SE05x to a host MCU/MPU using Platform SCP

Binding is a process to establish a pairing between the IoT device host MPU/MCU and EdgeLock SE05x, so that only the paired MPU/MCU is able to use the services offered by the corresponding EdgeLock SE05x and vice versa.

A mutually authenticated, encrypted channel will ensure that both parties are indeed communicating with the intended recipients and that local communication is protected against local attacks, including man-in-the-middle attacks aimed at intercepting the communication between the MPU/MCU and the EdgeLock SE05x and physical tampering attacks aimed at replacing the host MPU/MCU or EdgeLock SE05x .

EdgeLock SE05x natively supports Global Platform Secure Channel Protocol 03 (SCP03) for this purpose. PlatformSCP uses SCP03 and can be enabled to be mandatory.

This chapter describes the required steps to enable Platform SCP in the middlware for EdgeLock SE05x.

The following topics are discussed:

- Section 5.1 Introduction to the Global Platform Secure Channel Protocol 03 (SCP03)
- Section 5.2 How to configure the EdgeLock SE05x product specific SCP keys in the EdgeLock SE Plug & Trust Middleware
- Section 5.3 How to enable Platform SCP in the EdgeLock SE Plug & Trust Middleware

5.1 Introduction to the Global Platform Secure Channel Protocol 03 (SCP03)

The Secure Channel Protocol SCP03 authenticates and protects locally the bidirectional communication between host and EdgeLock SE05x against eavesdropping on the physical I2C interface.

EdgeLock SE05x can be bound to the host by injecting in both the host and EdgeLock SE05x the same unique SCP03 AES key-set and by enabling the Platform SCP feature in the EdgeLock SE Plug & Trust Middleware. The AN12662 *Binding a host device to EdgeLock SE05x* describes in detail the concept of secure binding.

SCP03 is defined in Global Platform Secure Channel Protocol '03' - Amendment D v1.2 specification.

SCP03 can provide the following three security goals:

- **Mutual authentication (MA)**
  - Mutual authentication is achieved through the process of initiating a Secure Channel and provides assurance to both the host and the EdgeLock SE05x entity that they are communicating with an authenticated entity.
- **Message Integrity**
  - The Command- and Response-MAC are generated by applying the CMAC according to NIST SP 800-38B.
- **Confidentiality**
  - The message data field is encrypted across the entire data field of the command message to be transmitted to the EdgeLock SE05x, and across the response transmitted from the EdgeLock SE05x.

The SCP03 secure channel is set up via the EdgeLock SE05x Java Card OS Manager using the standard ISO7816-4 secure channel APDUs.

The establishment of an SCP03 channel requires three static 128-bit AES keys shared between the two communicating parties: `Key-ENC`, `Key-MAC` and `Key-DEK`. These keys are stored in the Java Card Supplementary Security Domain (SSD) and not in the secure authenticator applet.

`Key-ENC` and `Key-MAC` keys are used during the SCP03 channel establishment to generate the session keys. Session Keys are generated to ensure that a different set of keys are used for each Secure Channel Session to prevent replay attacks.

`Key-ENC` is used to derive the session key `S-ENC`. The `S-ENC` key is used for encryption/decryption of the exchanged data. The session keys `S-MAC` and `R-MAC` are derived from `Key-MAC` and used to generate/verify the integrity of the exchanged data (C-APDU and R-APDU).

`Key-DEK` key is used to encrypt new SCP03 keys in case they get updated.

Table 10. Static SCP03 keys

Key	Description	Usage	Key Type
`Key-ENC`	Static Secure Channel Encryption Key	Generate session key for Decryption/ Encryption (AES)	AES 128
`Key-MAC`	Static Secure Channel Message Authentication Code Key	Generate session key for Secure Channel authentication and Secure Channel MAC Verification/Generation (AES)	AES 128
`Key-DEK`	Data Encryption Key	Sensitive Data Decryption (AES)	AES 128

The session key generation is performed by the EdgeLock SE Plug & Trust Middleware host crypto.

Table 11. SCP03 session keys

Key	Description	Usage	Key Type
`S-ENC`	Session Secure Channel Encryption Key	Used for data confidentiality	AES 128
`S-MAC`	Secure Channel Message Authentication Code Key for Command	Used for data and protocol integrity	AES 128
`S-RMAC`	Secure Channel Message Authentication Code Key for Response	User for data and protocol integrity	AES 128

Note: *For further details please refer to* Global Platform Secure Channel Protocol '03' - Amendment D v1.2.

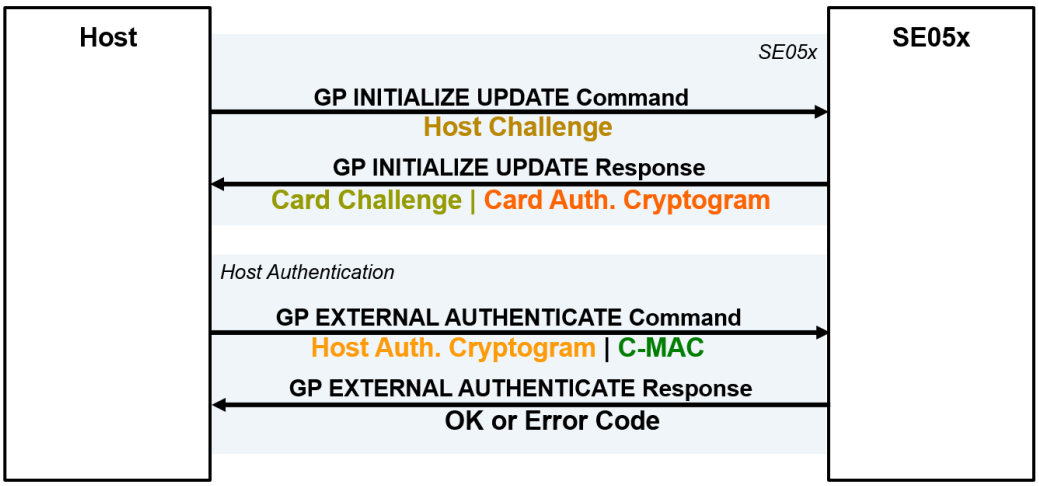


Figure 23. SPC03 mutual authentication – principle

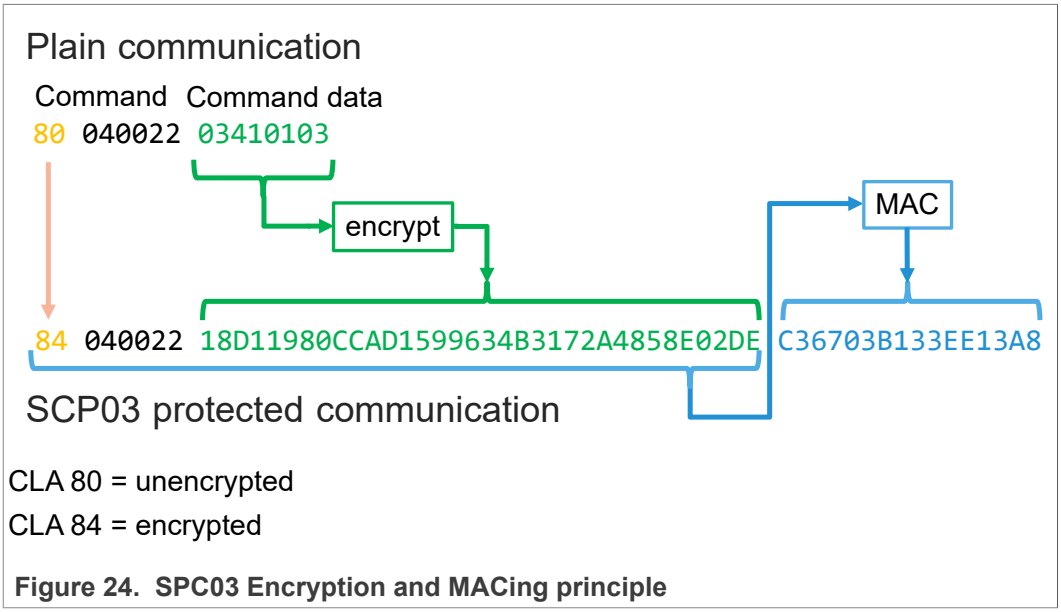


Figure 24. SPC03 Encryption and MACing principle

5.2 How to configure the product specific default Platform SCP keys

The initial Platform SCP key values are described for the SE050 product variants in AN12436 and for the SE051 variants in AN12973.

For evaluation purpose, the Platform SCP keys can be defined either in the EdgeLock SE Plug & Trust Middleware source code (see Section 5.2.1) or provided as text file (see Section 5.2.2).

Note: *In this example the Raspberry Pi is used for evaluation purpose only. Because different host MCU/MPU platforms are providing different hardware security mechanisms to protect keys it is not in the scope of this document to demonstrate how to store the Platform SCP shared binding keys securely. For commercial deployment the secure storage of Platform SCP keys must be adapted accordingly.*

5.2.1 Defining the deault Platfrom SCP keys in the EdgeLock SE Plug & Trust Middleware source code

The EdgeLock SE Plug & Trust Middleware header file `ex_sss_tp_scp03_keys.h` contains the default values of all EdgeLock SE05x, EdgeLock A5000 and A71CH product variants.

The `ex_sss_tp_scp03_keys.h` header file can be found in the following location: `/home/pi/se_mw/simw-top/sss/ex/inc/`

```
 GNU nano 3.2                                                ex_sss_tp_scp03_keys.h

#define SSS_AUTH_KEY_MAC \
    { 0x4F, 0x16, 0x3F, 0x59, 0xF0, 0x74, 0x31, 0xF4, 0x3E, 0xE2, 0xEE, 0x18, 0x34, 0xA5, 0x23, 0x34, }
#define SSS_AUTH_KEY_DEK \
    { 0xD4, 0x76, 0xCF, 0x47, 0xAA, 0x27, 0xB5, 0x4A, 0xB3, 0xDB, 0xEB, 0xE7, 0x65, 0x6D, 0x67, 0x70, }
#endif // SSS_PFSCP_ENABLE_SE051A_0001A920

// SSS_PFSCP_ENABLE_SE050E_0001A921
#if defined (SSS_PFSCP_ENABLE_SE050E_0001A921) && SSS_PFSCP_ENABLE_SE050E_0001A921 == 1
#define SSS_AUTH_KEY_ENC \
    { 0xD2, 0xDB, 0x63, 0xE7, 0xA0, 0xA5, 0xAE, 0xD7, 0x2A, 0x64, 0x60, 0xC4, 0xDF, 0xDC, 0xAF, 0x64, }
#define SSS_AUTH_KEY_MAC \
    { 0x73, 0x8D, 0x5B, 0x79, 0x8E, 0xD2, 0x41, 0xB0, 0xB2, 0x47, 0x68, 0x51, 0x4B, 0xFB, 0xA9, 0x5B, }
#define SSS_AUTH_KEY_DEK \
    { 0x67, 0x02, 0xDA, 0xC3, 0x09, 0x42, 0xB2, 0xC8, 0x5E, 0x7F, 0x47, 0xB4, 0x2C, 0xED, 0x4E, 0x7F, }
#endif // SSS_PFSCP_ENABLE_SE050E_0001A921

// SSS_PFSCP_ENABLE_SE051W_0005A739
#if defined (SSS_PFSCP_ENABLE_SE051W_0005A739) && SSS_PFSCP_ENABLE_SE051W_0005A739 == 1
#define SSS_AUTH_KEY_ENC \
    { 0x18, 0xB3, 0xB4, 0xE3, 0x40, 0xC0, 0x80, 0xD9, 0x9B, 0xEB, 0xB8, 0xB8, 0x64, 0x4B, 0x8C, 0x52, }
#define SSS_AUTH_KEY_MAC \
    { 0x3D, 0x0C, 0xFA, 0xC8, 0x7B, 0x96, 0x7C, 0x00, 0xE3, 0x3B, 0xA4, 0x96, 0x61, 0x38, 0x38, 0xA2, }
#define SSS_AUTH_KEY_DEK \
    { 0x68, 0x06, 0x83, 0xF9, 0x4E, 0x6B, 0xCB, 0x94, 0x73, 0xEC, 0xC1, 0x56, 0x7A, 0x1B, 0xD1, 0x09, }
#endif // SSS_PFSCP_ENABLE_SE051W_0005A739

// SSS_PFSCP_ENABLE_A5000_0004A736
#if defined (SSS_PFSCP_ENABLE_A5000_0004A736) && SSS_PFSCP_ENABLE_A5000_0004A736 == 1
#define SSS_AUTH_KEY_ENC \
    { 0xC9, 0x11, 0x85, 0x00, 0xB5, 0xFF, 0xA1, 0x43, 0x3A, 0x50, 0x22, 0x6F, 0x48, 0x9A, 0x0A, 0xA5, }
#define SSS_AUTH_KEY_MAC \
    { 0x29, 0xD2, 0xFE, 0x28, 0xF7, 0xFE, 0xEB, 0x15, 0x30, 0x68, 0xBE, 0x38, 0x1F, 0x61, 0xBC, 0x01, }
#define SSS_AUTH_KEY_DEK \
    { 0x61, 0x24, 0xD3, 0x84, 0x02, 0x11, 0x80, 0x60, 0xED, 0x91, 0x03, 0x60, 0xFC, 0x5A, 0x42, 0x78, }
#endif // SSS_PFSCP_ENABLE_A5000_0004A736
```

Figure 25. Default Platform SCP keys are defined in the `ex_sss_tp_scp03_keys.h` header file

The `fsl_sss_ftr.h.in` file includes options to select one of the predefined default Platform SCP keys. This file is located in: `/home/pi/se_mw/simw-top/sss/inc`.

Select the desired value of the compilation option by setting exclusively the corresponding C-preprocessor define `SSS_PFSCP_ENABLE_xx` to `1` (enable). All other values for the same option (represented by C-preprocessor defines `SSS_PFSCP_ENABLE_xx`) must be set to 0.

```
 GNU nano 3.2                                                  fsl_sss_ftr.h.in


/* Enable one of these
 * If none is selected, default config would be used
 */
#define SSS_PFSCP_ENABLE_SE050A1 0
#define SSS_PFSCP_ENABLE_SE050A2 0
#define SSS_PFSCP_ENABLE_SE050B1 0
#define SSS_PFSCP_ENABLE_SE050B2 0
#define SSS_PFSCP_ENABLE_SE050C1 0
#define SSS_PFSCP_ENABLE_SE050C2 0
#define SSS_PFSCP_ENABLE_SE050_DEVKIT 0
#define SSS_PFSCP_ENABLE_SE051A2 0
#define SSS_PFSCP_ENABLE_SE051C2 0
#define SSS_PFSCP_ENABLE_SE050F2 0
#define SSS_PFSCP_ENABLE_SE051C_0005A8FA 0
#define SSS_PFSCP_ENABLE_SE051A_0001A920 0
#define SSS_PFSCP_ENABLE_SE050E_0001A921 1
#define SSS_PFSCP_ENABLE_SE051W_0005A739 0
#define SSS_PFSCP_ENABLE_A5000_0004A736 0
#define SSS_PFSCP_ENABLE_SE050F2_0001A92A 0
#define SSS_PFSCP_ENABLE_OTHER 0
```

Figure 26. Select the actual Platform SCP keys in the `infsl_sss_ftr.h.in` file

The Plug & Trust Middleware uses a feature file to select/detect used/enabled features within the middleware stack. The file `fsl_sss_ftr.h` is automatically generated into the used build directory. CMake is overwritting the `fsl_sss_ftr.h` file every time CMake is invoked. CMake is using the SCP key settings of the `fsl_sss_ftr.h.in` file as input to generate the the `fsl_sss_ftr.h` file. You do not have to manually edit the `fsl_sss_ftr.h` feature file. Selections from CMake edit cache automatically updates the generated feature file.

Note: *The Platform SCP key selection in the `fsl_sss_ftr.h.in` CMake input file is persistent.*

The location of the generated fsl_sss_ftr.h feature header file is: `/home/pi/se_mw/simw-top_build/raspbian_native_se050_t1oi2c`

The following tables contains the the Platform SCP key header file define to be set to `1` (enable) for the different secure element and secure authenticator product variants.

Table 12. Platform SCP key define prefix for SE050E product variants

Variant	OEF ID	Platform SCP key define to be set to '1'
SE050E Dev. Board OM-SE050ARD-E	A921	`SSS_PFSCP_ENABLE_SE050E_0001A921`
SE050E2	A921	`SSS_PFSCP_ENABLE_SE050E_0001A921`

Table 13. Platform SCP key define prefix for SE050F product variants

Variant	OEF ID	Platform SCP key define to be set to '1'
SE050F Dev.Board OM-SE050ARD-F	A92A	`SSS_PFSCP_ENABLE_SE050F2_0001A92A`
SE050F2	A92A	`SSS_PFSCP_ENABLE_SE050F2_0001A92A`

Table 14. Platform SCP key define prefix for SE050 Previous Generation product variants

Variant	OEF ID	Platform SCP key define to be set to '1'
SE050A1	A204	`SSS_PFSCP_ENABLE_SE050A1`

Table 14. Platform SCP key define prefix for SE050 Previous Generation product variants*...continued*

Variant	OEF ID	Platform SCP key define to be set to '1'
SE050A2	A205	SSS_PFSCP_ENABLE_SE050A2
SE050B1	A202	SSS_PFSCP_ENABLE_SE050B1
SE050B2	A203	SSS_PFSCP_ENABLE_SE050B2
SE050C1	A200	SSS_PFSCP_ENABLE_SE050C1
SE050C2	A201	SSS_PFSCP_ENABLE_SE050C2
SE050 Dev Board OM-SE050ARD	A1F4	SSS_PFSCP_ENABLE_SE050_DEVKIT
SE050F2	A77E[1]	SSS_PFSCP_ENABLE_SE050F2

[1] All SE050F2 with variant A77E have date code in year 2021. All the SE050F2 with date code in the year 2022 have the variant identifier A92A.

Table 15. Platform SCP key define prefix for SE051 product variants

Variant	OEF ID	Platform SCP key define to be set to '1'
SE051A2	A920	SSS_PFSCP_ENABLE_SE051A_0001A920
SE051C2	A8FA	SSS_PFSCP_ENABLE_SE051C_0005A8FA
SE051W2	A739	SSS_PFSCP_ENABLE_SE051W_0005A739
SE051A2	A565	SSS_PFSCP_ENABLE_SE051A2
SE051C2	A564	SSS_PFSCP_ENABLE_SE051C2

Table 16. Platform SCP key define prefix for A5000 product variants

Variant	OEF ID	Platform SCP key define to be set to '1'
A5000 Dev. Board OM-A5000ARD	A736	SSS_PFSCP_ENABLE_A5000_0004A736
A5000	A736	SSS_PFSCP_ENABLE_A5000_0004A736

5.2.2 Defining the default Platfrom SCP keys in a text file

For evaluation purpose the EdgeLock SE Plug & Trust Middleware supports to store the Platform SCP key in a plain text file. For further details see EdgeLock SE Plug & Trust Middleware documentation chapter *11.10 Using own Platform SCP03 keys*.

The following Linux commands can be used to create the Platform SCP key text file (`se050_Dev_Kit_scp_keys.txt`):

The Platform SCP key text file can be stored in any location. In this example the file is stored in: `~/se_mw/simw-top_build/raspbian_native_se050_t1oi2c/bin`

```
cd ~/se_mw/simw-top_build/raspbian_native_se050_t1oi2c/bin
echo ENC D2DB63E7A0A5AED72A6460C4DFDCAF64 > se050E_scp_keys.txt
echo MAC 738D5B798ED241B0B24768514BFBA95B >> se050E_scp_keys.txt
echo DEK 6702DAC30942B2C85E7F47B42CED4E7F >> se050E_scp_keys.txt
```

Check the `se050E_scp_keys.txt` file content:

```
cat se050E_scp_keys.txt
```

The Linux environment variable `EX_SSS_BOOT_SCP03_PATH` is used to define the Platform SCP key textfile (filename and location).

```
export EX_SSS_BOOT_SCP03_PATH=~/se_mw/simw-top_build/
raspbian_native_se050_t1oi2c/bin/se050E_scp_keys.txt
```

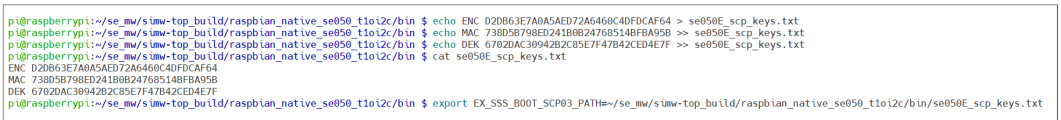


Figure 27. EdgeLock SE05xPlatform SCP plain text key file

Note: *The EdgeLock SE Plug & Trust Middleware will first look for the default path `/tmp/SE05X/plain_scp.txt`, if it is not able to find the file, it will try to use the environment variable EX_SSS_BOOT_SCP03_PATH, and lastly, it will fall back to pre-compiled keys.*

5.3 How to enable Platform SCP in the CMake-based build system

To enable Platform SCP is required to rebuild the SDK with the following CMake options:

- Select `SCP03_SSS` for the CMake option `PTMW_SCP`.
- Select `PlatfSCP03` for the CMake option `PTMW_SE05X_Auth`.

The following images show the configuration for the SE050E development board OM-SE05ARD-E.

Figure 28. SE050E CMake Settings - PlatformSCP enabled

Run the following commands to update the CMake settings and rebuild the EdgeLock SE Plug & Trust Middleware:

```
cd ~/se_mw/simw-top_build/raspbian_native_se050_t1oi2c
cmake-gui .
```

Update the CMake settings as explained above. Press first the ***Configure*** button and second the ***Generate*** button and close the CMake GUI.

```
cmake --build .
sudo make install
sudo ldconfig /usr/local/lib/
```

In the next step we can verify if we successfully enabled Platform SCP. For this purpose we run again the se05x_minimal example:

```
cd bin
./se05x_Minimal
```

Figure 29 shows the log output in case the Platform SCP keys are defined in the EdgeLock SE Plug & Trust Middleware source code (see Section 5.2.1).

```
pi@raspberrypi:~/se_mw/simw-top_build/raspbian_native_se050_t1oi2c $ cd bin
pi@raspberrypi:~/se_mw/simw-top_build/raspbian_native_se050_t1oi2c/bin $ ./se05x_Minimal
App   :INFO :PlugAndTrust_v04.00.00_20211026
App   :INFO :Running ./se05x_Minimal
App   :INFO :If you want to over-ride the selection, use ENV=EX_SSS_BOOT_SSS_PORT or pass in command line arguments.
App   :INFO :Using default PlatfSCP03 keys. You can use keys from file using ENV=EX_SSS_BOOT_SCP03_PATH
sss   :INFO :atr (Len=35)
      01 A0 00 00    03 96 04 03    E8 00 FE 02    0B 03 E8 00
      01 00 00 00    00 64 13 88    0A 00 65 53    45 30 35 31
      00 00 00
App   :INFO :mem=32767
App   :INFO :se05x_Minimal Example Success !!!...
App   :INFO :ex_sss Finished
pi@raspberrypi:~/se_mw/simw-top_build/raspbian_native_se050_t1oi2c/bin $
```

Figure 29. Run se05x_minimal example with platformSCP enabled - SCP keys defined in the EdgeLock SE Plug & Trust Middleware source

The log output for defining the Platform SCP keys via a text file (see Section 5.2.2) is shown in Figure 30.

```
pi@raspberrypi:~/se_mw/simw-top_build/raspbian_native_se050_t1oi2c/bin $ ./se05x_Minimal
App   :INFO :PlugAndTrust_v04.00.00_20211026
App   :INFO :Running ./se05x_Minimal
App   :INFO :If you want to over-ride the selection, use ENV=EX_SSS_BOOT_SSS_PORT or pass in command line arguments.
App   :WARN :Using SCP03 keys from:'/home/pi/se_mw/simw-top_build/raspbian_native_se050_t1oi2c/bin/se050E_scp_keys.txt' (ENV=EX_SSS_BOOT_SCP03_PATH)
sss   :INFO :atr (Len=35)
      01 A0 00 00    03 96 04 03    E8 00 FE 02    0B 03 E8 00
      01 00 00 00    00 64 13 88    0A 00 65 53    45 30 35 31
      00 00 00
App   :INFO :mem=32767
App   :INFO :se05x_Minimal Example Success !!!...
App   :INFO :ex_sss Finished
```

Figure 30. Run se05x_minimal example with platformSCP enabled - SCP keys defined in a text file

The Plug & Trust Middleware provides the following additional examples to rotate the PlatformSCP Keys and to mandate Platform SCP.

- **SE05x Rotate PlatformSCP Keys example**: Showcases authentication with default Platform SCP keys and the rotation (update) of those keys with user defined keys. The example documentation is available in the EdgeLock SE05x Plug & Trust Middleware documentation (`~/se_mw/simw-top/doc/demos/se05x/se05x_RotatePlatformSCP03Keys/Readme.html`). The example source code is available at `~/se_mw/simw-top/demos/se05x/se05x_RotatePlatformSCP03Keys`.

- **SE05X Mandate SCP example**: Showcases how to make Platform SCP authentication mandatory in EdgeLock SE05x. The example documentation is available in the EdgeLock SE05x Plug & Trust Middleware documentation (`~/se_mw/simw-top/doc/demos/se05x/se05x_MandatePlatformSCP/Readme.html`). The example source code is available at `~/se_mw/simw-top/demos/se05x/se05x_MandatePlatformSCP`.

- **SE05x AllowWithout PlatformSCP example**: This project demonstrates how to configure SE05X to allow without platform SCP. The example documentation is available in the EdgeLock SE05x Plug & Trust Middleware documentation (`~/se_mw/simw-top/doc/demos/se05x/se05x_AllowWithoutPlatformSCP/Readme.html`). The example source code is available at `~/se_mw/simw-top/demos/se05x/se05x_AllowWithoutPlatformSCP`.

6 Manage access from multiple Linux processes to the EdgeLock SE05x

The EdgeLock SE Plug & Trust Middleware provides the Access Manager to support concurrent access from multiple linux processes to the EdgeLock SE05x IoT applet. The Access Manager can establish a connection to the EdgeLock SE05x IoT applet either as a plain connection or using Platform SCP.

Client processes are connecting over the JRCPv1 protocol to the Access Manager.

Please refer to the EdgeLock SE Plug & Trust Middleware documentation chapter Access Manager: Manage access from multiple (Linux) processes to an SE05x IoT Applet for more details.

7 Appendix A: Using the ssscli tool

EdgeLock SE Plug & Trust Middleware also provides the `ssscli` tool. This tool can be used to interact with the EdgeLock SE05x security IC without having to write any code.

The ssscli is a command line tool that can be used to send commands to EdgeLock SE05x interactively through the command line. For example, you can use the ssscli to create keys and credentials in the EdgeLock SE05x security IC during evaluation, development and testing phases. The ssscli tool is written in Python and supports complex provisioning scripts that can be run in Windows, Linux, OS X and other embedded devices. It can be used to:

- Insert keys and certificates in DER or PEM format into the EdgeLock SE05x.
- Retrieve the public keys and certificates form EdgeLock SE05x and store the key into a DER (Distinguished Encoding Rules) or PEM (Privacy Enhanced Mail) formatted file.
- Create reference-keys and store the key into a DER or PEM formatted file.
- Delete EdgeLock SE05x (erase) keys and certificates inside.
- Generate keys inside the EdgeLock SE05x.
- Attach policies to objects.
- List all EdgeLock SE05x secure objects.
- Retrieve the EdgeLock SE05x device unique ID.
- Run some basic EdgeLock SE05x operations like sign/verify and encrypt/decrypt operations.

Please refer to the EdgeLock SE Plug & Trust Middleware documentation chapter "9. CLI Tool" for detailed description how to use ssscli tool.

For installing the ssscli tool follow the steps below shown in Figure 31:

1. Move to the user directory
   >> `cd /home/pi`
2. Ensure pytyhon3-pip is installed
   `>> sudo apt-get install python3-pip`

3. Ensure libffi-dev is installed:
   >> `sudo apt-get install libffi-dev`
   ***Note**: In this case, the packages were already installed*

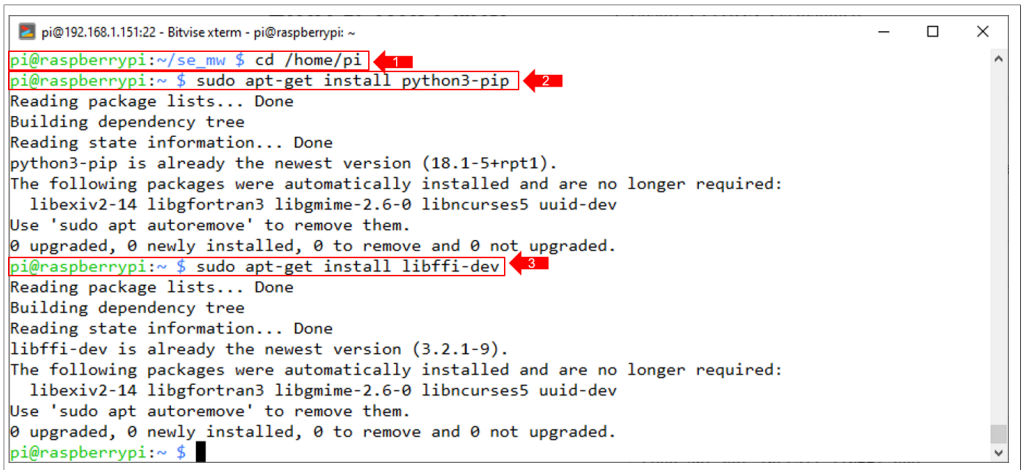


Figure 31. Install python and libffi-dev

Make sure you have cmake installed and configured for the Raspbian Host as done in Section 3.3.

4. Ensure click, cryptography and func-timeout modules are installed. Figure 32 shows how to install these modules, change directory to:
   >> `cd /home/pi/se_mw/simw-top/pycli`
5. and run the following command:
   >> `pip3 install -r requirements.txt`
6.

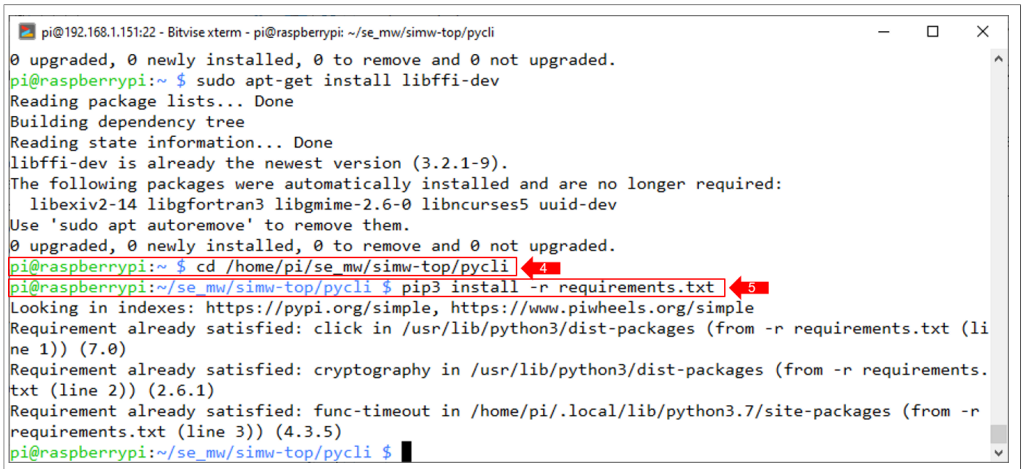


Figure 32. Install required modules

7. Install the `ssscli` tool as Figure 33 shows:
   >> `cd src`
   >> `sudo python3 setup.py develop`

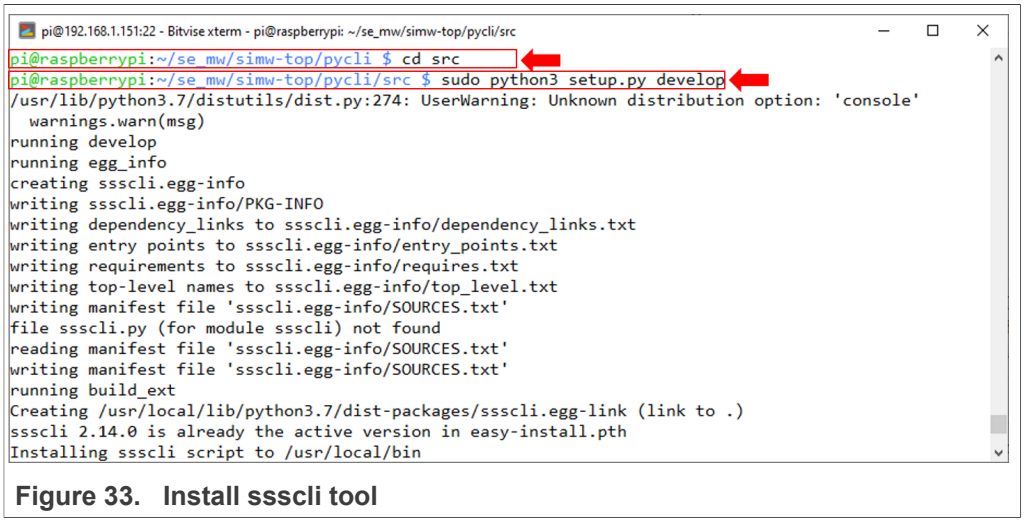


Figure 33. Install ssscli tool

To start the ssscli tool, send the commands shown in Figure 34:

1. Move to the user directory:
   >> `cd /home/pi`
2. Open the connection
   >> `ssscli connect se05x t1oi2c none`

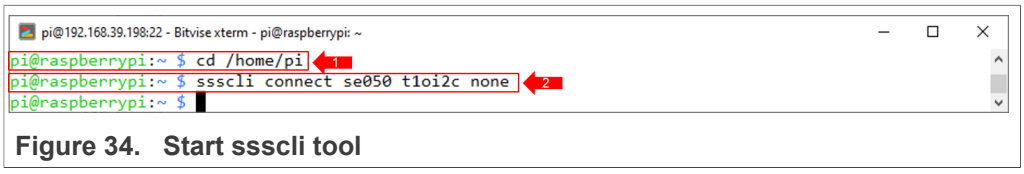


Figure 34. Start ssscli tool

The SE05x ssscli tool supports several operations. To check which commands are supported by the ssscli tool (Figure 35):

>> `ssscli --help`

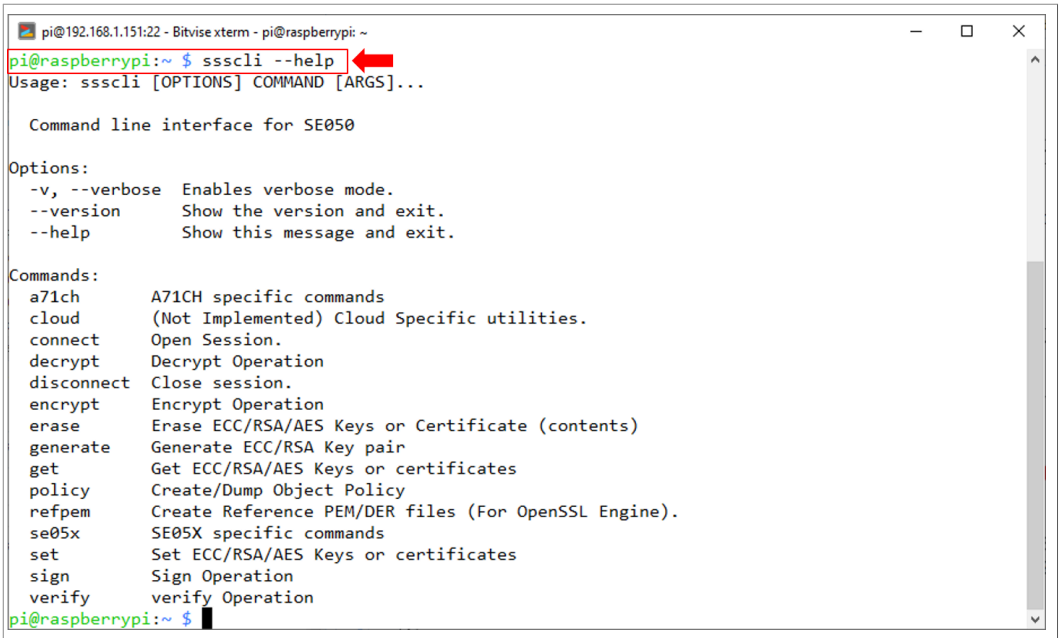


Figure 35. ssscli tool help menu

Each of these options provides information about the syntax used for each specific command. For instance, the se05x option:

```
>> ssscli se05x
```

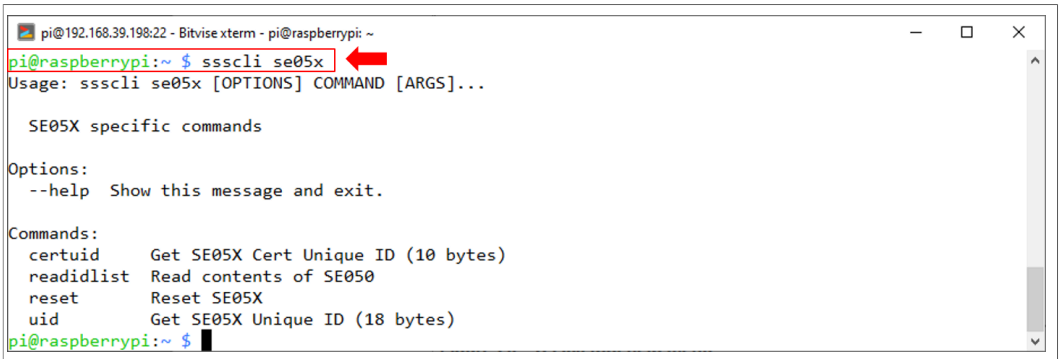


Figure 36. ssscli se05x help menu

To read the credentials and secure objects stored in the EdgeLock SE05x, you can send the following command (Figure 37):

```
>> ssscli se05x readidlist
```

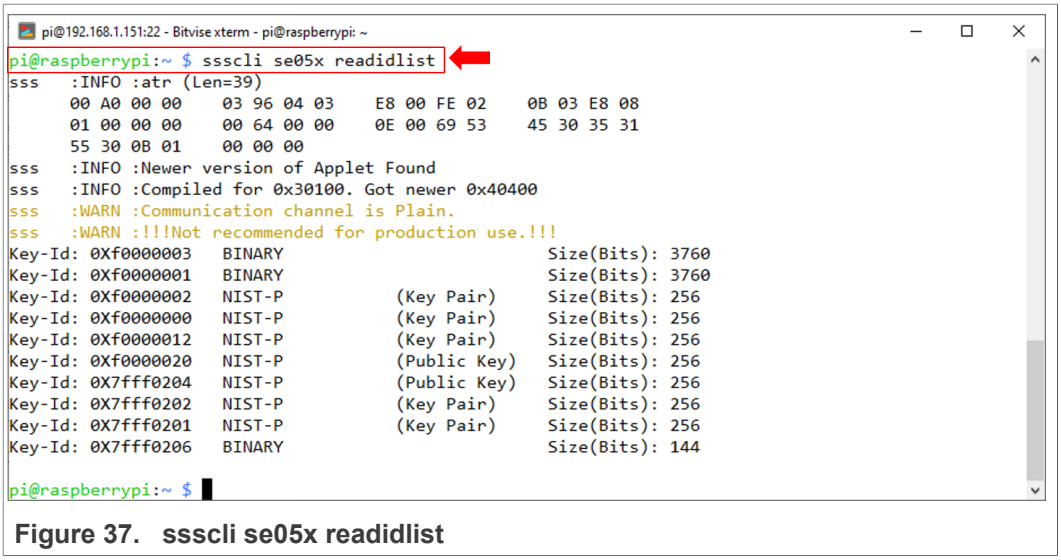


Figure 37. ssscli se05x readidlist

8 Legal information

8.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

8.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors. In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory. Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer. In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages. Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

8.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

Tables

Figures

Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 3 August 2022
Document identifier: AN12570
Document number: 565815