AN12663

EdgeLock^{TM} SE05x to implement TPM-like functionality

Rev. 1.0 — 13 April 2021 Application note

Document information

Information	Content
Keywords	EdgeLock SE050, EdgeLock SE051, TPM functionality, TPM Software Stack (TSS)
Abstract	This document introduces the benefits provided by EdgeLock SE05x to implement TPM-like functionalities in your IoT devices. It also describes the TSS wrapper layer implemented in the Plug & Trust middleware to simplify integration of EdgeLock SE05x and to enable fast migration from a traditional TPM, and explains how to run and evaluate the TPM project examples provided in the support package.

Revision history

Revision history

Revision number	Date	Description
1.0	2021-04-13	First document release

1 Introduction

For more than a decade, the computing industry has relied on a special type of secure crypto-processor, called a Trust Platform Module (TPM), to provide hardware-based protection of PCs, laptops, networking equipment, and other computing devices. TPM functionality is specified as ISO/IEC 118889, and TPM operation is certified by the Trusted Computing Group (TCG), an industry organization formed by leading computer-platform companies.

In computing, the TPM is a tamper resilient coprocessor chip used to securely store the credentials required for user password protection, disk encryption and trusted execution. TPM chips can also store Platform Configuration Registers (PCRs), which allow tracking the installed SW and system configuration and help ensure the computing platform’s trustworthiness over time.

IoT devices face some of the same risks as network-connected computers. In resource-constrained IoT devices that require flexible crypto functionality in lightweight implementations, adding a traditional TPM can create excess overhead in terms of size of the SW stack and platform resources required.

In addition, traditional TPMs are not flexible enough to provide the crypto functionality required to support IoT-specific tasks, such as creating a secure network connection, onboarding on multiple cloud, storing multiple keys to authenticate data or securely connect to multiple other devices, among others.

Furthermore, the different threat model and form factor associated with IoT devices require features typically not implemented in a TPM, such as: secure binding to the host controller, a small footprint to fit in compact devices, or programmability to adapt the security logic to the type of IoT device.

As a result, a secure element equipped with TPM functionality, like EdgeLock SE05x, can add high-level protection in a format better suited for IoT operation. In addition, to simplify integration of TPM functionality using EdgeLock SE05x, the Plug & Trust middleware provides an adaptation layer for easy integration into the TPM Software Stack (TSS), as outlined in Section 3.

2 EdgeLock SE05x to implement TPM-like functionality

The EdgeLock SE05x is a tamper-resistant secure element able to bring TPM functionality to IoT applications. The entire EdgeLock SE05x secure element family is delivered with a pre-installed applet optimized for IoT use cases that also provides TPM-like functions, such as secure cryptographic processing, secure key storage, unique ID generation and storage, attestation capabilities, and PCRs to remotely verify device health and ensure trust.

The EdgeLock SE05x goes beyond baseline TPM operation to provide special support for IoT operation, including:

- Flexible approach to manage credentials and user policies (i.e. more user/policy combinations are possible per credential object).
- Support secure binding to a host MCU (e.g. using GlobalPlatform's SCP03 standard protocol).
- Ability to freeze keys (and avoid deletion by other stakeholders).
- Configuration of access-right policies on the on-chip memory (in combination with NXP EdgeLock 2GO service, supports management of keys and digital certificates over the air, in the field).
- Multi-tenancy, where multiple stakeholders can use the same EdgeLock SE05x secure element to securely store their sensitive data and credentials.

In addition, the EdgeLock SE05x is part of NXP EdgeLock Assurance Program and provides certified security according to Common Criteria framework with EAL 6+ resistance level at hardware but also at operating system level. The EdgeLock SE05x secure elements are also designed for scalability, and can easily be configured to support existing and upcoming standards, such as CHIP (Connected Home over IP) for Smart Home, DLMS-COSEM for Smart Metering, ISA/IEC 62443 for Industrial Control Security and the Open Platform Communication United Architecture (OPC UA), which defines data-exchange standards for industrial communication.

As a result, the major advantage of EdgeLock SE05x over traditional TPMs is that it supports more IoT-relevant features, a wider variety of development and usage models, and can be used in tiny sensors as well as powerful IoT equipment such as edge computing platforms.

3 TSS implementation in Plug & Trust middleware

The Plug & Trust middleware provides already an OpenSSL engine to let standard applications use cryptography via the secure element without influence on the applications code. In case the applications do not use OpenSSL as cryptographic API the Plug & Trust middleware provides a TSS adaptation layer for integration into the TPM Software Stack (TSS) to enable a fast migration from a traditional TPM to an embedded secure element. Refer to Appendix A for some additional details about TPM 2.0 and TPM Software Stack (TSS).

The TSS implementation available at https://github.com/tpm2-software/tpm2-tss is used by the Plug & Trust middleware to provide TPM functions. An Esys wrapper software implementation, interfacing with the ESAPI and FAPI layers, takes care of translating TPM commands to commands that can be managed by the Plug & Trust middleware. This architecture is shown in Figure 1:

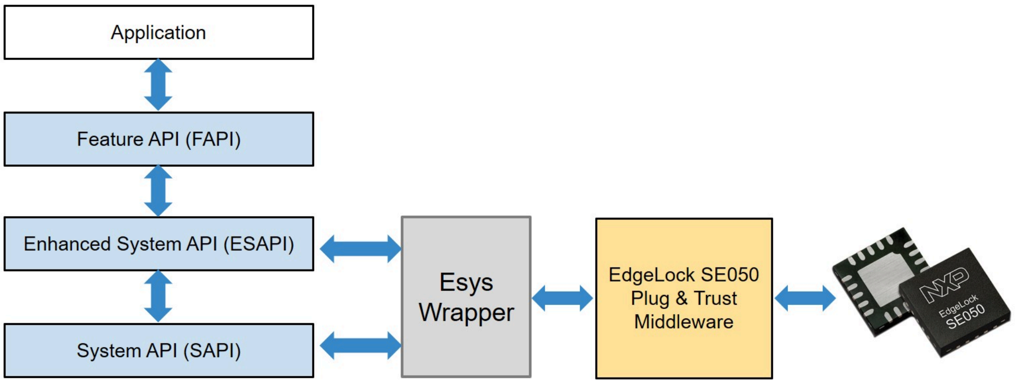


Figure 1. TSS architecture in EdgeLock SE05x

The functions supported by the EdgeLock SE05x TPM implementation are listed in Table 1. For a list of limitations and unsupported features, please refer to Section 6.

Table 1. TPM Functions supported by Plug & Trust middleware

Function	TPM APIs	Supported Algorithms
Asymmetric Signing and Verification	Esys_VerifySignature () Esys_Sign ()	RSA-SSA (TPM2_ALG_RSASSA) RSA-PSS (TPM2_ALG_RSAPSS) RSA-ECDSA (TPM2_ALG_ ECDSA)
Asymmetric RSA Encryption and Decryption	Esys_RSA_Encrypt () Esys_RSA_Decrypt ()	RSA-OAEP (TPM2_ALG_OAEP) RSA (TPM2_ALG_RSAES)
AES Encryption & Decryption	Esys_EncryptDecrypt () Esys_EncryptDecrypt2 ()	AES-CTR (TPM2_ALG_CTR) AES-CBC (TPM2_ALG_CBC) AES-ECB (TPM2_ALG_ECB)
Hashing	Esys_Hash ()	SHA1 (TPM2_ALG_SHA1) SHA256 (TPM2_ALG_SHA256) SHA384 (TPM2_ALG_SHA384) SHA512 (TPM2_ALG_SHA512)

Table 1. TPM Functions supported by Plug & Trust middleware*...continued*

Function	TPM APIs	Supported Algorithms
HMAC	Esys_HMAC ()	SHA1 (TPM2_ALG_SHA1) SHA256 (TPM2_ALG_SHA256) SHA384 (TPM2_ALG_SHA384) SHA512 (TPM2_ALG_SHA512)
Random number generation	Esys_GetRandom ()	-
PCR	Esys_PCR_Extend () Esys_PCR_Event () Esys_PCR_Read () Esys_PCR_Allocate () Esys_PCR_Reset ()	-
Support functions	Esys_ReadPublic ()	-

4 Run the Plug & Trust middleware TPM examples

This section describes how to compile and run the TPM examples provided as part of the Plug & Trust middleware. The examples use the TPM2-Tools as a convenient way to demonstrate the TPM capabilities of EdgeLock SE05x. The TPM2-Tools are only supported in Linux, but the underlying TPM library can also be used in other operating systems.

4.1 Hardware preparation

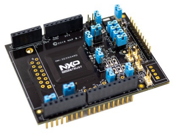
In this section the necessary hardware for running the Plug & Trust middleware with the TPM examples is described.

4.1.1 Required hardware

The following hardware is used to run the TPM project examples:

1. OM-SE05xARD development kit:
   The EdgeLock SE05x support package provides development boards for evaluating EdgeLock SE050 and EdgeLock SE051 features. Select the development board of the product you want to evaluate. Table 2 details the ordering details of the EdgeLock SE05x development boards.

Table 2. EdgeLock SE05x development boards.

Part number	12NC	Description	Picture
OM-SE050ARD	935383282598	SE050 Arduino® compatible development kit	
OM-SE051ARD	935399187598	SE051 Arduino® compatible development kit	

Note: The pictures in this guide will show EdgeLock SE050, but EdgeLock SE051 can be used as well with the same configuration.

2. OM-SE050RPI adapter board for Raspberry Pi:

Table 3. OM-SE050RPI adapter board details

Part number	12NC	Content	Picture
OM-SE050RPI	935379833598	Raspberry Pi to OM-SE05xARD adapter	

3. Raspberry Pi board:

Table 4. Raspberry Pi

Part number	Content	Picture
Raspberry Pi	Any Raspberry Pi model is sufficient, usually models 2, 3 and 4 are used	

4.1.2 Hardware setup

The hardware setup consists of two steps:

1. Make sure the jumpers in your OM-SE05xARD board are configured as shown in Figure 2:

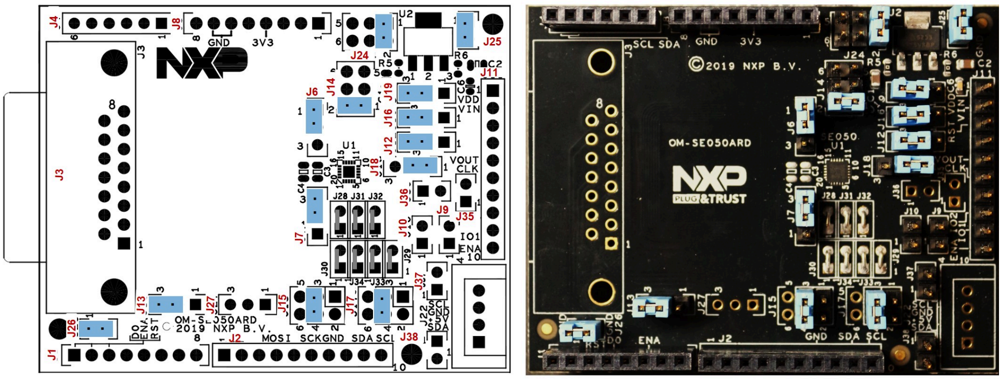


Figure 2. OM-SE05xARD jumper configuration

For more information on the hardware refer to AN12395 - OM-SE050ARD hardware overview.

2. Connect the OM-SE05xARD to the Raspberry Pi, following the steps shown in Figure 3: First mount the OM-SE05xARD on top of the OM-SE050RPI board using

the Arduino connectors. Then mount the two boards on top of the Raspberry Pi using the Raspberry connectors in the OM-SE050RPI. The result is three boards stacked together, with the OM-SE050RPI the board in between the Raspberry Pi and OM-SE05xARD.

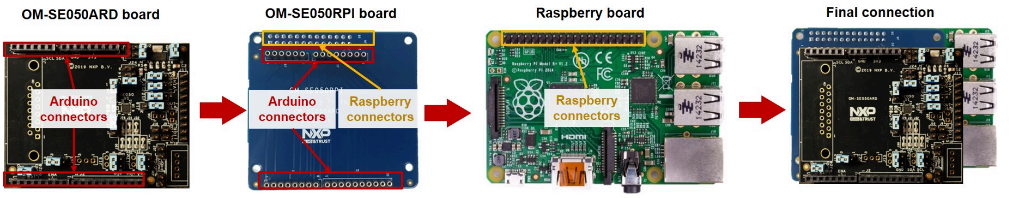


Figure 3. OM-SE05xARD connection to the Raspberry Pi using the OM-SE050RPI adapter board

Note: In case you do not have the OM-SE050RPI adapter board, you can also manually wire the Raspberry Pi to the OM-SE05xARD using the external I^{2}C connector. For more information refer AN12570 - Quick start guide with Raspberry Pi.

4.2 Install Raspberry OS

The Raspberry OS installation consists of two steps:

1. Install your preferred Linux distribution in your device as described in Section 4.2.1.
2. Enable the I^{2}C interface in your Linux distribution to allow the communication with the security IC of the OM-SE05xARD board as described in Section 4.2.2.

4.2.1 Installation

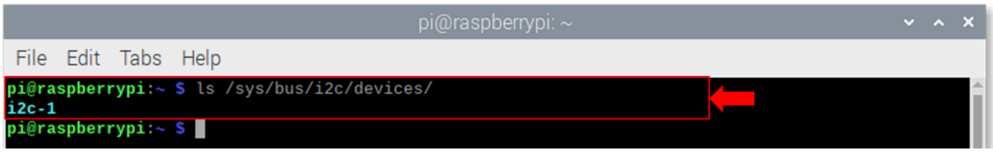
First, we need to install the OS for our Raspberry Pi. For that, we use the latest Raspbian OS version available in the Raspberry website. It recommends tw options:

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.
   Note: *Raspbian is used just as a reference; you can use your preferred Linux distribution.*

4.2.2 Enable the I2C interface

The Raspberry Pi board communicates with the OM-SE05xARD security IC through the I^{2}C interface. The I^{2}C interface is not enabled by default in Raspbian and must be activated before the Plug & Trust middleware test examples can be executed. To enable I^{2}C, open a Terminal window and follow these steps:

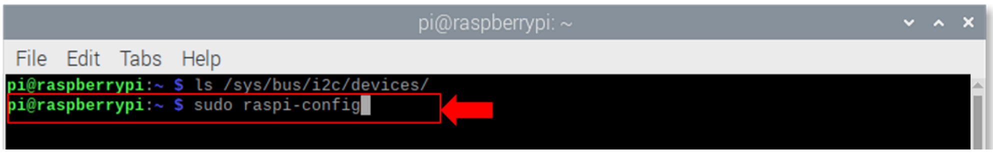
1. Verify if I²C is active by listing the available I²C interfaces:
   Send >> `ls /sys/bus/i2c/devices/`
   If the *i2c-x* interface is listed, as shown in Figure 4 , then you can skip this section and proceed to Section 4.3.
   Note: *The I²C interface number might be different.*



```
pi@raspberrypi:~ $ ls /sys/bus/i2c/devices/
i2c-1
pi@raspberrypi:~ $ 
```

Figure 4. List I²C interfaces

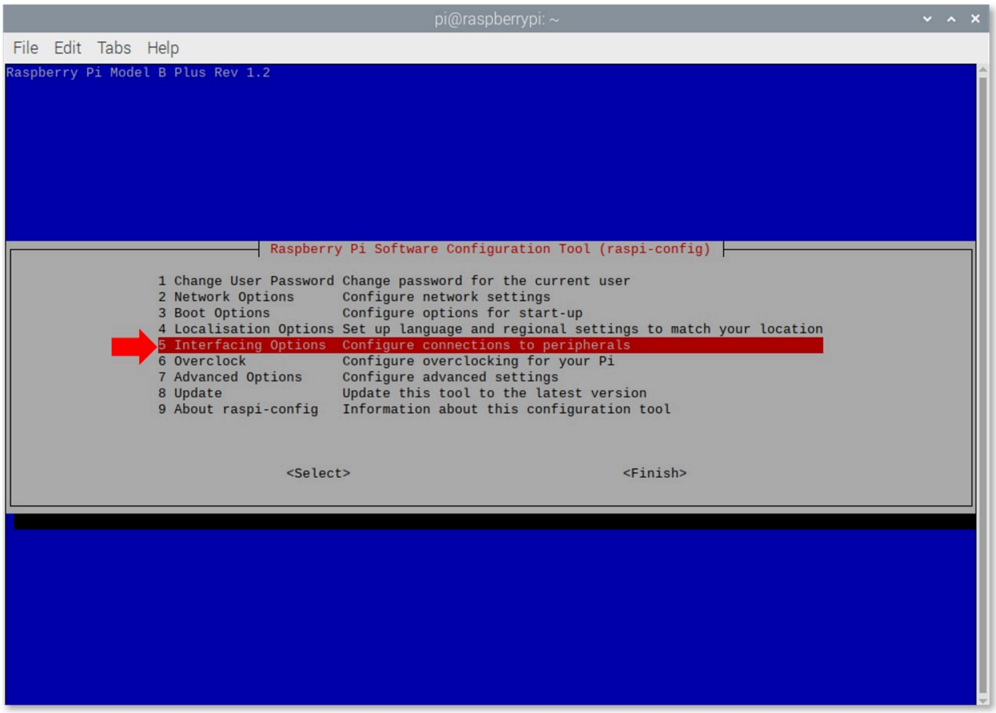
2. Open the Raspberry Pi software configuration tool, as shown in Figure 5:
   Send >> `sudo raspi-config`



```
pi@raspberrypi:~ $ ls /sys/bus/i2c/devices/
pi@raspberrypi:~ $ sudo raspi-config
```

Figure 5. Open the Raspberry Pi software configuration tool

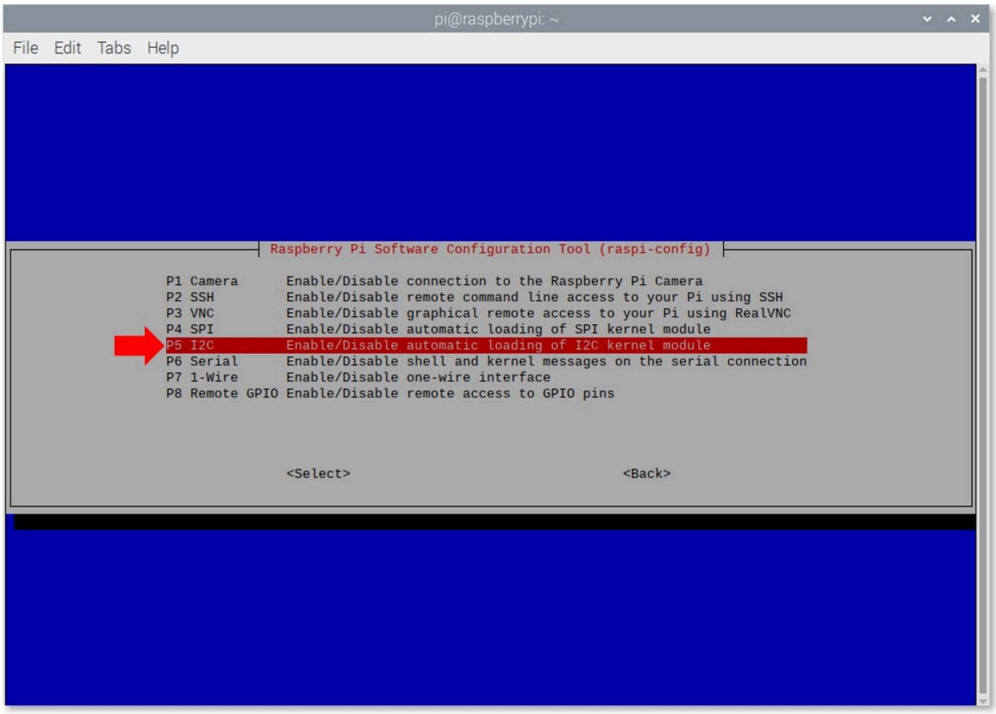
3. Use the up and down arrow keys to select the 5th menu entry (Interfacing Options) and then press Enter, as shown in Figure 6:



```
Raspberry Pi Model B Plus Rev 1.2

                     Raspberry Pi Software Configuration Tool (raspi-config)

         1 Change User Password Change password for the current user
         2 Network Options      Configure network settings
         3 Boot Options         Configure options for start-up
         4 Localisation Options Set up language and regional settings to match your location
         5 Interfacing Options  Configure connections to peripherals
         6 Overclock            Configure overclocking for your Pi
         7 Advanced Options     Configure advanced settings
         8 Update               Update this tool to the latest version
         9 About raspi-config   Information about this configuration tool

                   <Select>                                  <Finish>
```

Figure 6. Enable I²C interface

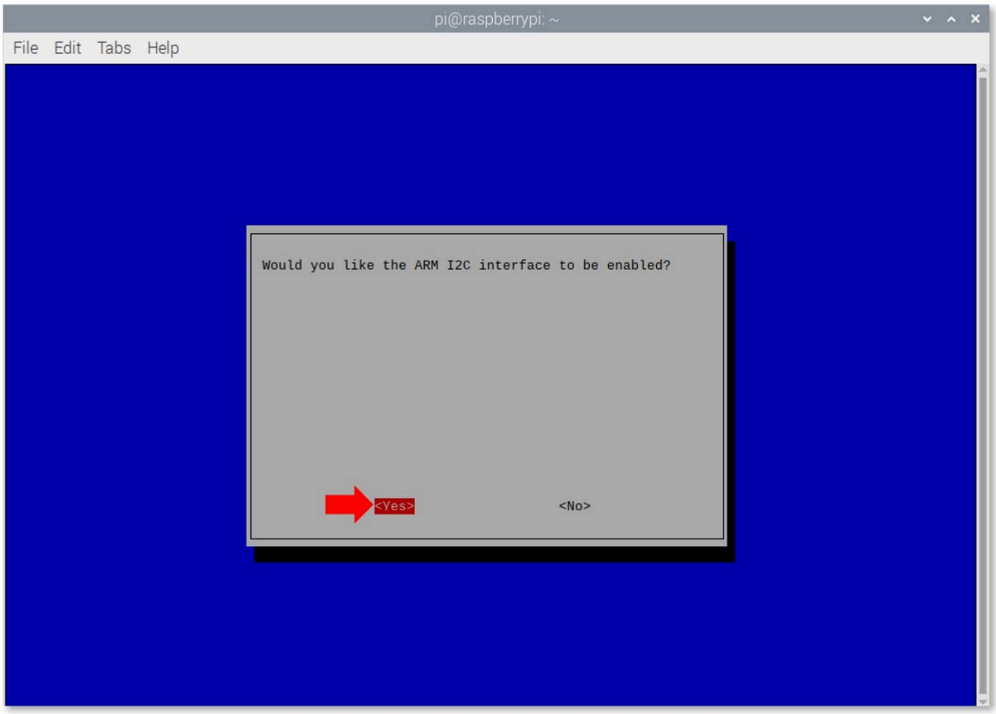
4. Use the up and down arrow keys to select the 5th menu option (I^2C) and then press Enter, as shown in Figure 7:



```
                  Raspberry Pi Software Configuration Tool (raspi-config)

                   P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
                   P2 SSH         Enable/Disable remote command line access to your Pi using SSH
                   P3 VNC         Enable/Disable graphical remote access to your Pi using RealVNC
                   P4 SPI         Enable/Disable automatic loading of SPI kernel module
                   P5 I2C         Enable/Disable automatic loading of I2C kernel module
                   P6 Serial      Enable/Disable shell and kernel messages on the serial connection
                   P7 1-Wire      Enable/Disable one-wire interface
                   P8 Remote GPIO Enable/Disable remote access to GPIO pins


                                  <Select>                                  <Back>
```

Figure 7. Enable I^2C interface

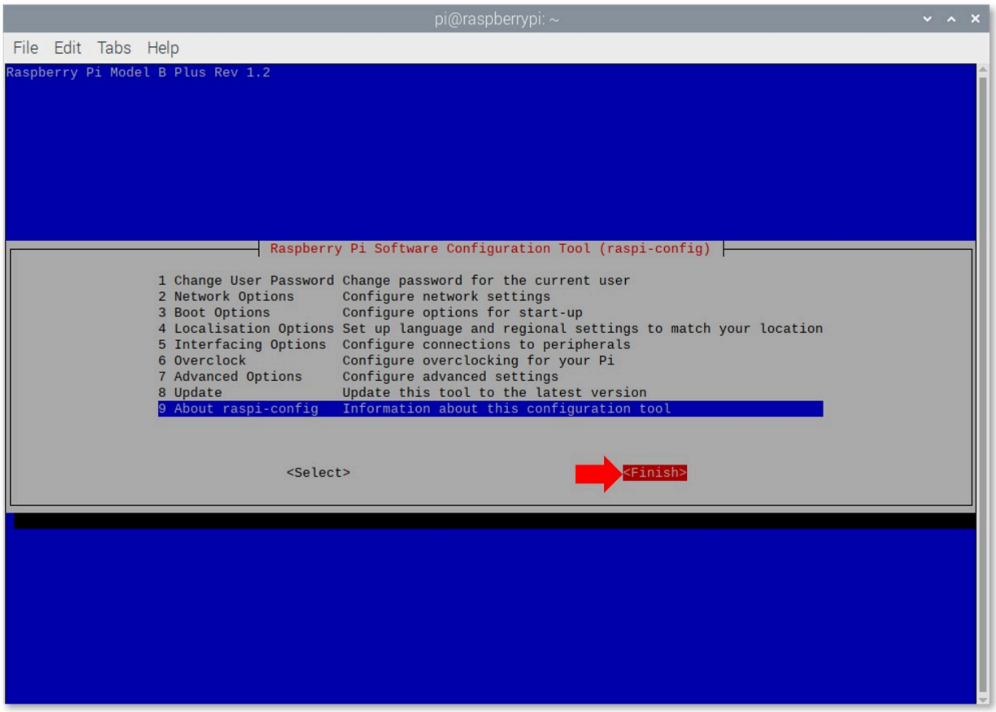
5. You will be asked to confirm your choice to activate the I^{2}C interface. Use the left and right arrow keys to select the Yes option and then press Enter, as shown in Figure 8:



```
Would you like the ARM I2C interface to be enabled?
```

Figure 8. Enable I^{2}C interface

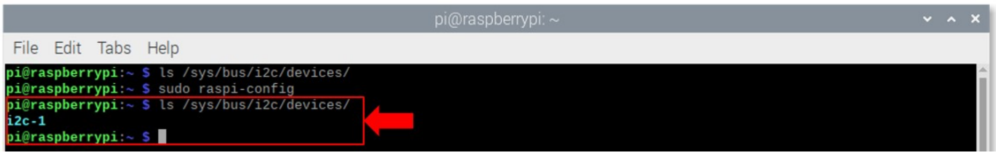
6. Close the Raspberry Pi software configuration tool. Use the left and right arrow keys to select the Finish option and then press Enter, as shown in Figure 9:



```
Raspberry Pi Model B Plus Rev 1.2

                ┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

                   1 Change User Password Change password for the current user
                   2 Network Options      Configure network settings
                   3 Boot Options         Configure options for start-up
                   4 Localisation Options Set up language and regional settings to match your location
                   5 Interfacing Options  Configure connections to peripherals
                   6 Overclock            Configure overclocking for your Pi
                   7 Advanced Options     Configure advanced settings
                   8 Update               Update this tool to the latest version
                   9 About raspi-config   Information about this configuration tool


                          <Select>                              <Finish>
```

Figure 9. Close the Raspberry Pi sofware configuration tool

7. Verify the correct activation of the I^{2}C interface, as shown in Figure 10:
Send >> `ls /sys/bus/i2c/devices/`
The *i2c-x* interface should now be listed.
Note: *The I^{2}C interface number might be different.*



```
pi@raspberrypi:~ $ ls /sys/bus/i2c/devices/
pi@raspberrypi:~ $ sudo raspi-config
pi@raspberrypi:~ $ ls /sys/bus/i2c/devices/
i2c-1
pi@raspberrypi:~ $
```

Figure 10. List I^{2}C interfaces

4.3 Compile and run the Plug & Trust middleware with TPM examples

This section details the steps required from the moment you download the Plug & Trust middleware and the TPM addon until you are able to run a TPM test example.

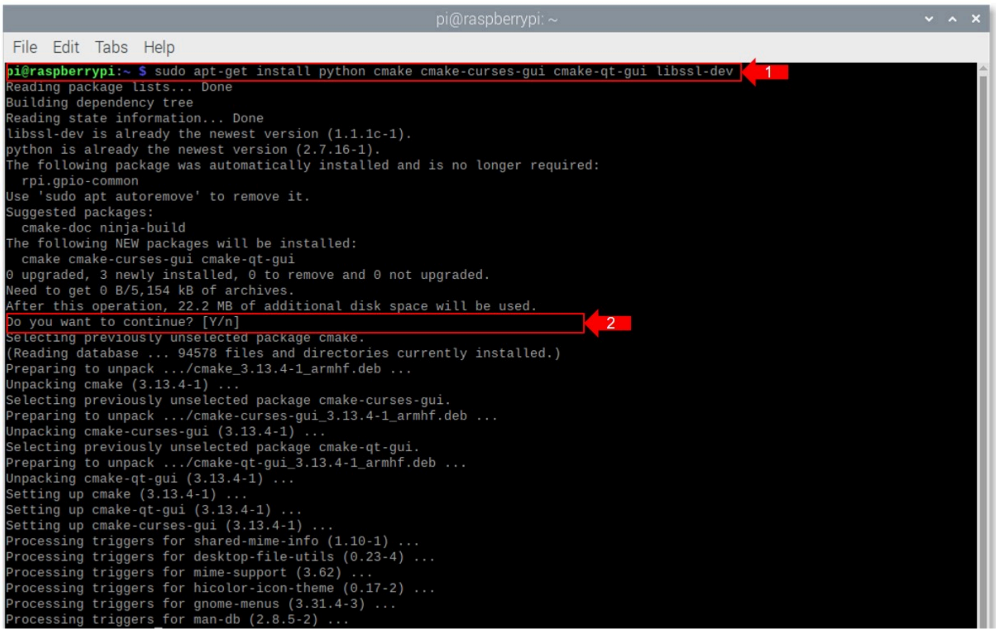
4.3.1 Install build tools

To build the Plug & Trust 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).

- In order to download Python, refer to its website https://www.python.org/downloads/.

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 11:

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



```
pi@raspberrypi:~ $ sudo apt-get install python cmake cmake-curses-gui cmake-qt-gui libssl-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
libssl-dev is already the newest version (1.1.1c-1).
python is already the newest version (2.7.16-1).
The following package was automatically installed and is no longer required:
  rpi.gpio-common
Use 'sudo apt autoremove' to remove it.
Suggested packages:
  cmake-doc ninja-build
The following NEW packages will be installed:
  cmake cmake-curses-gui cmake-qt-gui
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/5,154 kB of archives.
After this operation, 22.2 MB of additional disk space will be used.
Do you want to continue? [Y/n]
Selecting previously unselected package cmake.
(Reading database ... 94578 files and directories currently installed.)
Preparing to unpack .../cmake_3.13.4-1_armhf.deb ...
Unpacking cmake (3.13.4-1) ...
Selecting previously unselected package cmake-curses-gui.
Preparing to unpack .../cmake-curses-gui_3.13.4-1_armhf.deb ...
Unpacking cmake-curses-gui (3.13.4-1) ...
Selecting previously unselected package cmake-qt-gui.
Preparing to unpack .../cmake-qt-gui_3.13.4-1_armhf.deb ...
Unpacking cmake-qt-gui (3.13.4-1) ...
Setting up cmake (3.13.4-1) ...
Setting up cmake-qt-gui (3.13.4-1) ...
Setting up cmake-curses-gui (3.13.4-1) ...
Processing triggers for shared-mime-info (1.10-1) ...
Processing triggers for desktop-file-utils (0.23-4) ...
Processing triggers for mime-support (3.62) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for gnome-menus (3.31.4-3) ...
Processing triggers for man-db (2.8.5-2) ...
```

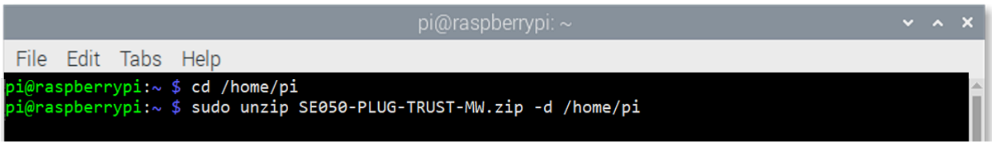
Figure 11. Install build tools

4.3.2 Download the Plug & Trust middleware and the TPM addon

To prepare the folders that will be used during the implementation of the TPM examples follow the steps below:

1. Download the Plug & Trust middleware from NXP website and place the .zip file in the */home/pi* directory of your Raspbian distribution.

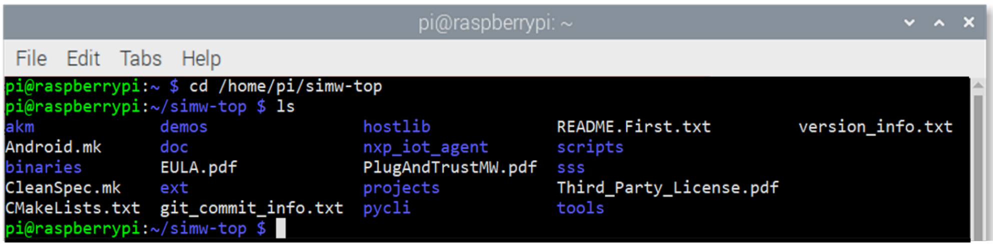
2. Open a Terminal window and follow the next steps as shown in Figure 12:
   a. Move to the *home* directory:
      Send >> `cd /home/pi/`
   b. Unzip the Plug & Trust middleware in the */home/pi* folder:
      Send >> `unzip SE-PLUG-TRUST-MW.zip -d /home/pi`
      Notes:
      - *The name of the zip file might be different.*
      - *This command may take a few seconds to complete.*
      - *Inside this archive you will find some documenation in PDF ("PlugAndTrustMWTPM.pdf") and HTML format (doc/html/).*



```
pi@raspberrypi:~ $ cd /home/pi
pi@raspberrypi:~ $ sudo unzip SE050-PLUG-TRUST-MW.zip -d /home/pi
```

Figure 12. Unzip the Plug & Trust 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 */home/pi* folder:
      Send >> `cd /home/pi/simw-top`
   b. List the content of the *simw-top* folder:
      Send >> `ls`
      The content of the folder should be the same as shown in Figure 13:

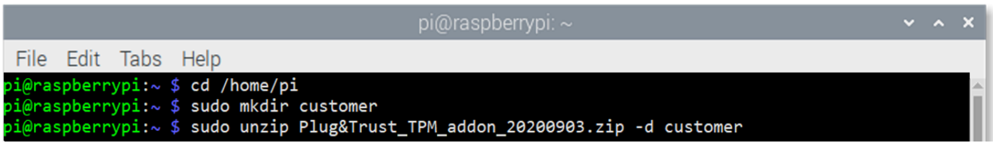


```
pi@raspberrypi:~ $ cd /home/pi/simw-top
pi@raspberrypi:~/simw-top $ ls
akm             demos               hostlib            README.First.txt          version_info.txt
Android.mk      doc                 nxp_iot_agent      scripts
binaries        EULA.pdf            PlugAndTrustMW.pdf sss
CleanSpec.mk    ext                 projects           Third_Party_License.pdf
CMakeLists.txt  git_commit_info.txt pycli              tools
pi@raspberrypi:~/simw-top $ 
```

Figure 13. simw-top folder content

4. Obtain the TPM addon from your NXP representative and place the .zip file in the */home/pi* directory of your Raspbian distribution.

5. Open a Terminal window and follow the next steps as shown in Figure 14:
   a. Move to the *home/pi* directory:
      Send >> `cd /home/pi`
   b. Create a folder called *customer:*
      Send >> `mkdir customer`
   c. Unzip the TPM addon in the *customer* folder:
      Send >> `unzip SE-PLUG-TRUST-MW-ADDON-TPM.zip -d customer`
      Note: *The name of the zip file might be different.*
      Note: *This command may take a few seconds to complete.*



```
pi@raspberrypi:~ $ cd /home/pi
pi@raspberrypi:~ $ sudo mkdir customer
pi@raspberrypi:~ $ sudo unzip Plug&Trust_TPM_addon_20200903.zip -d customer
```

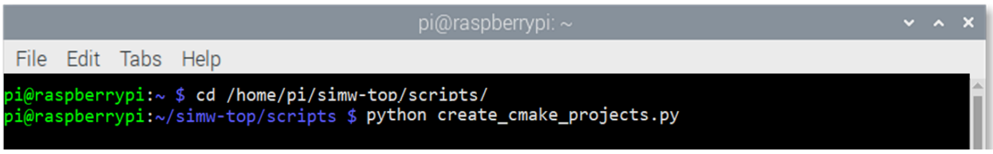
Figure 14. Unzip the TPM addon folder

6. You can verify that the files have been correctly unzipped inside the customer folder by following these steps:
   a. Move to the *customer* folder inside the */home/pi* folder:
      Send >> `cd /home/pi/customer`
   b. List the content of the *customer* folder:
      Send >> `ls`
      Now you should find the subfolder *tpm2* there.

4.3.3 Build the Plug & Trust middleware

We can use Cmake to build Plug & Trust middleware into our Raspbian Image. Open a terminal windown and follow the steps as shown in Figure 15:

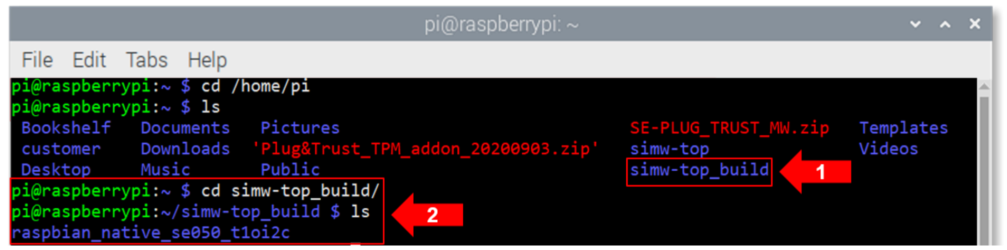
1. Go to the folder with the unzipped SE050 middleware:
   Send >> `cd /home/pi/simw-top/scripts`
2. Generate the Plug & Trust middleware project examples:
   Send >> `python create_cmake_projects.py`
   Note: *This command may take a few seconds to complete.*



```
pi@raspberrypi:~ $ cd /home/pi/simw-top/scripts/
pi@raspberrypi:~/simw-top/scripts $ python create_cmake_projects.py
```

Figure 15. Build Plug & Trust middleware

3. If the compilation is successful you should (1) see a new *simw-top_build* folder inside the */home/pi* folder and (2) a new folder inside the `simw-top_build` folder as shown in Figure 16:



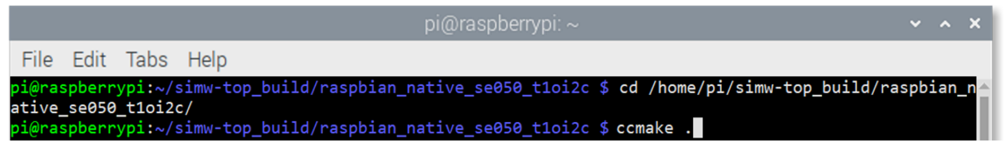
```
pi@raspberrypi:~ $ cd /home/pi
pi@raspberrypi:~ $ ls
 Bookshelf   Documents   Pictures                              SE-PLUG_TRUST_MW.zip  Templates
 customer    Downloads  'Plug&Trust_TPM_addon_20200903.zip'    simw-top              Videos
 Desktop     Music       Public                                simw-top_build
pi@raspberrypi:~ $ cd simw-top_build/
pi@raspberrypi:~/simw-top_build $ ls
raspbian_native_se050_t1oi2c
```

Figure 16. EdgeLock SE05x middleware project structure

4.3.4 Build the TPM test examples

We can use cmake to compile the TPM test examples. Open a Terminal window and follow these steps:

1. Go to the created build folder
   Send >> `cd /home/pi/simw-top_build/raspbian_native_se050_t1oi2c`
2. Open the cmake configuration interface, as shown in Figure 17:
   Send >> `ccmake .`
   Note: *You can use the graphical interface by sending* `cmake-gui .` *instead.*

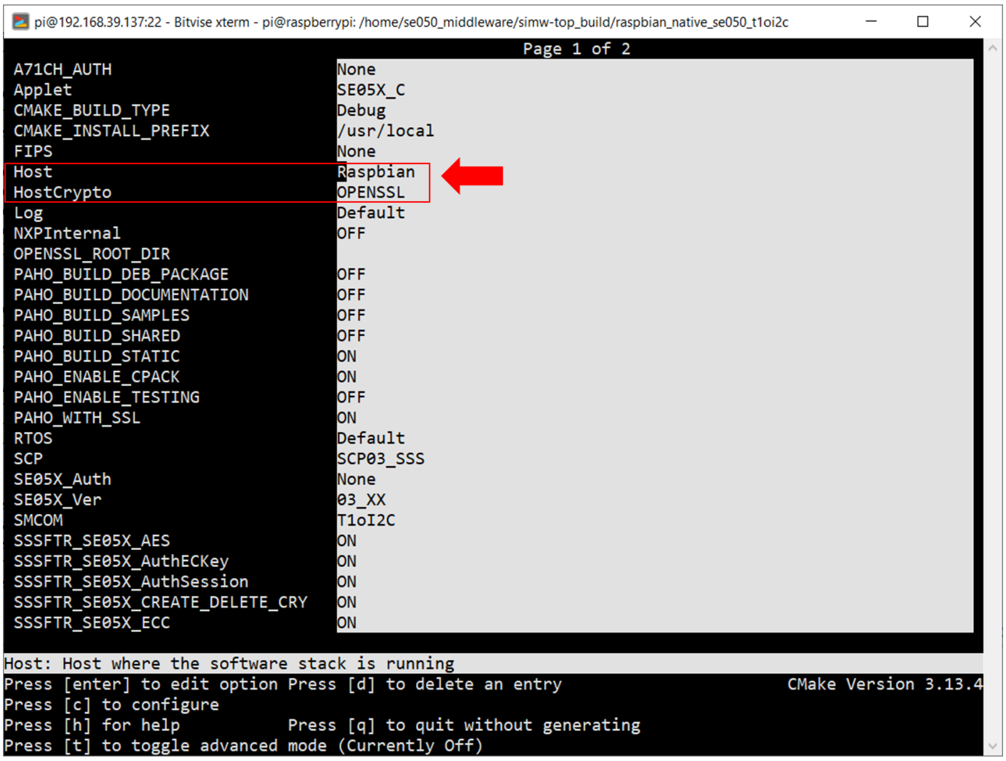


```
pi@raspberrypi:~/simw-top_build/raspbian_native_se050_t1oi2c $ cd /home/pi/simw-top_build/raspbian_n
ative_se050_t1oi2c/
pi@raspberrypi:~/simw-top_build/raspbian_native_se050_t1oi2c $ ccmake .
```

Figure 17. Open CMake configuaration interface

3. Review the build configuration and make sure that the *Host* parameter is set to the value *Raspbian*, the *HostCrypto* is set to the value *OPENSSL*, as shown in Figure 18. For changing 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.

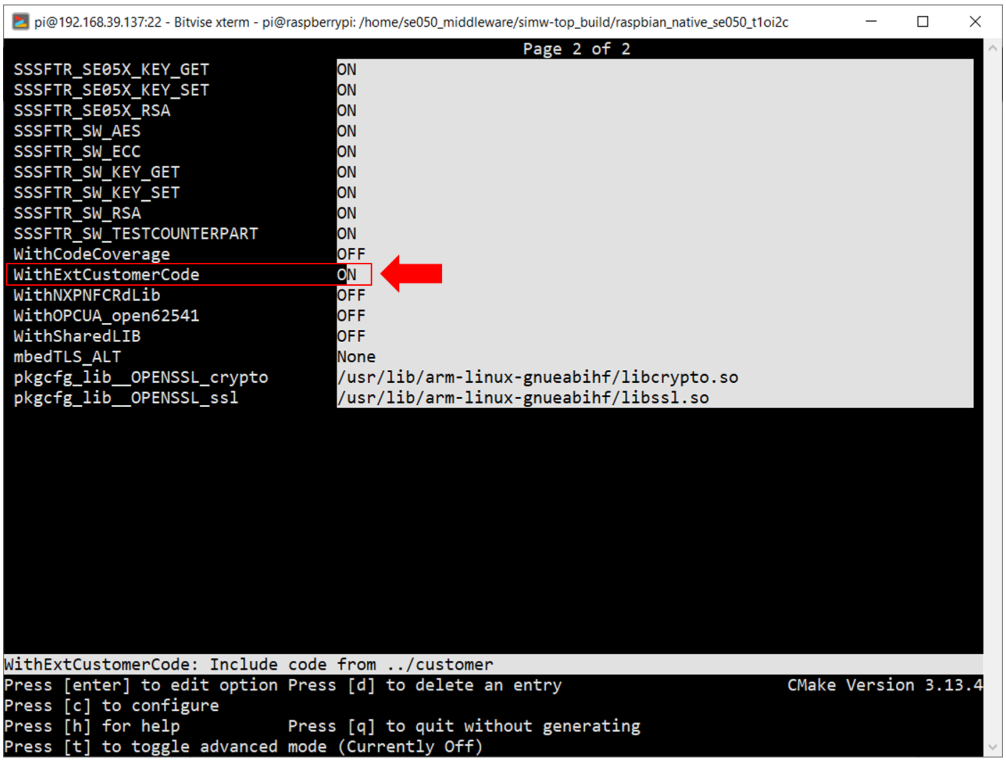


pi@192.168.39.137:22 - Bitvise xterm - pi@raspberrypi: /home/se050_middleware/simw-top_build/raspbian_native_se050_t1oi2c

```
                                                 Page 1 of 2
 A71CH_AUTH                       None
 Applet                           SE05X_C
 CMAKE_BUILD_TYPE                 Debug
 CMAKE_INSTALL_PREFIX             /usr/local
 FIPS                             None
 Host                             Raspbian
 HostCrypto                       OPENSSL
 Log                              Default
 NXPInternal                      OFF
 OPENSSL_ROOT_DIR
 PAHO_BUILD_DEB_PACKAGE           OFF
 PAHO_BUILD_DOCUMENTATION         OFF
 PAHO_BUILD_SAMPLES               OFF
 PAHO_BUILD_SHARED                OFF
 PAHO_BUILD_STATIC                ON
 PAHO_ENABLE_CPACK                ON
 PAHO_ENABLE_TESTING              OFF
 PAHO_WITH_SSL                    ON
 RTOS                             Default
 SCP                              SCP03_SSS
 SE05X_Auth                       None
 SE05X_Ver                        03_XX
 SMCOM                            T1oI2C
 SSSFTR_SE05X_AES                 ON
 SSSFTR_SE05X_AuthECKey           ON
 SSSFTR_SE05X_AuthSession         ON
 SSSFTR_SE05X_CREATE_DELETE_CRY   ON
 SSSFTR_SE05X_ECC                 ON

Host: Host where the software stack is running
Press [enter] to edit option Press [d] to delete an entry                    CMake Version 3.13.4
Press [c] to configure
Press [h] for help           Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

Figure 18. Review build configuration 1

Turn to the next page using the down arrow key and check that the *WithExtCustomerCode* is set to *ON* as shown in Figure 19.



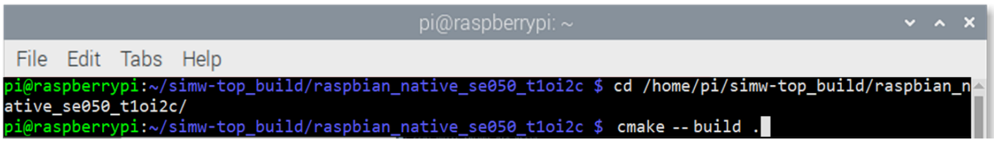
pi@192.168.39.137:22 - Bitvise xterm - pi@raspberrypi: /home/se050_middleware/simw-top_build/raspbian_native_se050_t1oi2c

```
                                                 Page 2 of 2
 SSSFTR_SE05X_KEY_GET             ON
 SSSFTR_SE05X_KEY_SET             ON
 SSSFTR_SE05X_RSA                 ON
 SSSFTR_SW_AES                    ON
 SSSFTR_SW_ECC                    ON
 SSSFTR_SW_KEY_GET                ON
 SSSFTR_SW_KEY_SET                ON
 SSSFTR_SW_RSA                    ON
 SSSFTR_SW_TESTCOUNTERPART        ON
 WithCodeCoverage                 OFF
 WithExtCustomerCode              ON
 WithNXPNFCRdLib                  OFF
 WithOPCUA_open62541              OFF
 WithSharedLIB                    OFF
 mbedTLS_ALT                      None
 pkgcfg_lib__OPENSSL_crypto       /usr/lib/arm-linux-gnueabihf/libcrypto.so
 pkgcfg_lib__OPENSSL_ssl          /usr/lib/arm-linux-gnueabihf/libssl.so

WithExtCustomerCode: Include code from ../customer
Press [enter] to edit option Press [d] to delete an entry                    CMake Version 3.13.4
Press [c] to configure
Press [h] for help           Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

Figure 19. Review build configuration 2

After editing the configuration, press `c` (configure) and then `g` (generate) to apply the changes.

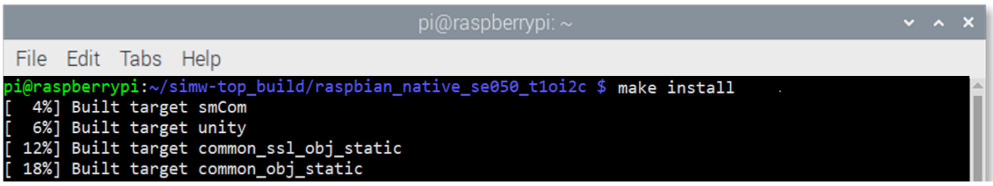
4. Build the project examples, as shown in Figure 20:
   Send >> `cmake --build .`
   Note: *This command may take a few seconds to complete.*



```
pi@raspberrypi:~/simw-top_build/raspbian_native_se050_t1oi2c $ cd /home/pi/simw-top_build/raspbian_n
ative_se050_t1oi2c/
pi@raspberrypi:~/simw-top_build/raspbian_native_se050_t1oi2c $ cmake --build .
```

Figure 20. Build project examples

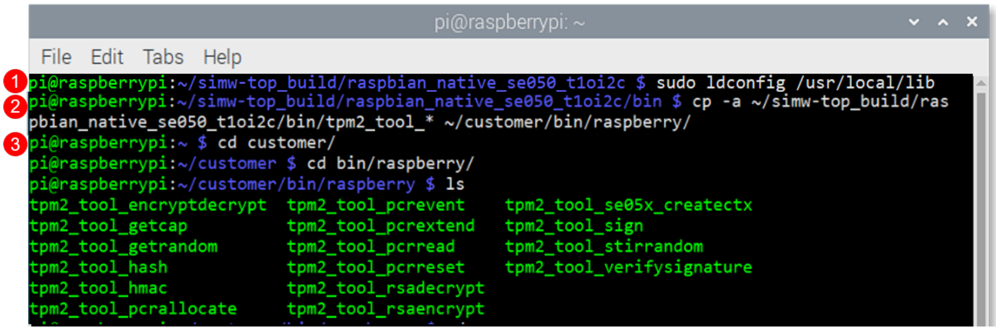
5. Install the projects in the system as shown in Figure 21:
   Send >> `make install`
   Note: *This command may take a few seconds to complete.*



```
pi@raspberrypi:~/simw-top_build/raspbian_native_se050_t1oi2c $ make install
[  4%] Built target smCom
[  6%] Built target unity
[ 12%] Built target common_ssl_obj_static
[ 18%] Built target common_obj_static
```

Figure 21. Install projects in the system

6. Update the cache to include the newly installed libraries.
   (1) Send >> `sudo ldconfig /usr/local/lib`
   Copy the generated TPM binaries from the `sim-top_build` folder to the `customer`.
   (2) Send >> `cp -a ~/simw-top_build/raspbian_native_se050_t1oi2c/bin/tpm2_tool_* ~/customer/bin/raspberry/`
   (3) Check the `/customer/bin/raspberry/`folder.
   Send >> `cd /home/pi/customer/bin/raspberry/`
   Send >> `ls`



```
pi@raspberrypi:~/simw-top_build/raspbian_native_se050_t1oi2c $ sudo ldconfig /usr/local/lib
pi@raspberrypi:~/simw-top_build/raspbian_native_se050_t1oi2c/bin $ cp -a ~/simw-top_build/ras
pbian_native_se050_t1oi2c/bin/tpm2_tool_* ~/customer/bin/raspberry/
pi@raspberrypi:~ $ cd customer/
pi@raspberrypi:~/customer $ cd bin/raspberry/
pi@raspberrypi:~/customer/bin/raspberry $ ls
tpm2_tool_encryptdecrypt  tpm2_tool_pcrevent    tpm2_tool_se05x_createctx
tpm2_tool_getcap          tpm2_tool_pcrextend   tpm2_tool_sign
tpm2_tool_getrandom       tpm2_tool_pcrread     tpm2_tool_stirrandom
tpm2_tool_hash            tpm2_tool_pcrreset    tpm2_tool_verifysignature
tpm2_tool_hmac            tpm2_tool_rsadecrypt
tpm2_tool_pcrallocate     tpm2_tool_rsaencrypt
```

Figure 22. Load new installed libraries

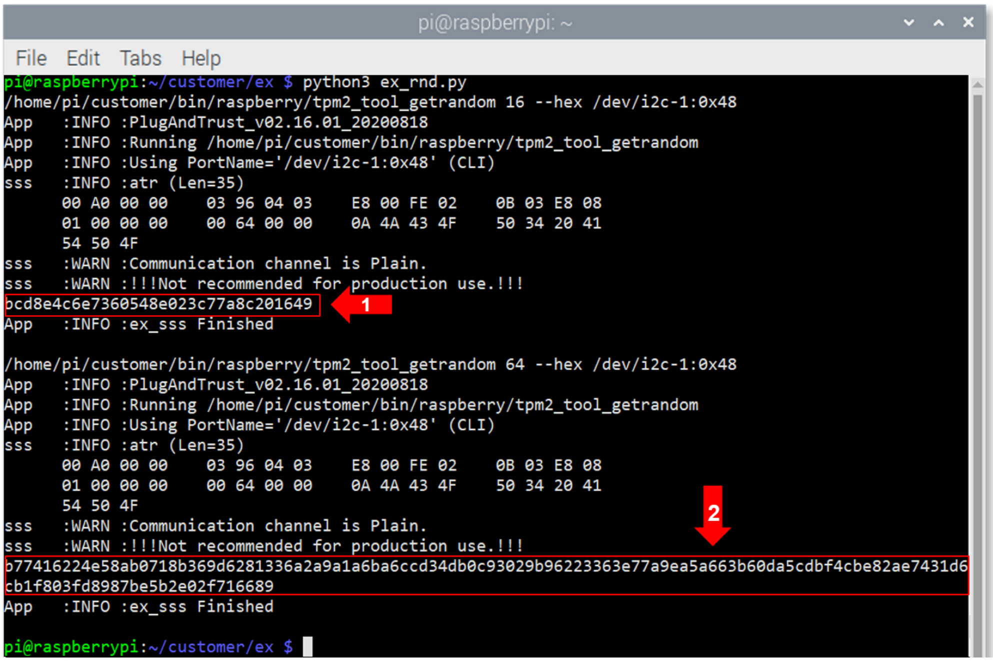
4.3.5 Execute TPM examples

This section explains how to run the TPM test example called `ex_rnd.py`. The `ex_rnd.py` is a python code that calls twice the `tpm2_tool_getrandom` binary with an specific input. This binary generates a random number with variable length, in the case of the `ex_rnd.py` the output is two random numbers of 16 and 64 Bytes. To execute the `ex_rnd.py` test example follow these steps:

Open a Terminal window and follow the steps as shown in Figure 23:

1. Move to the directory containing the examples:
   Send >> `cd /home/pi/customer/ex`

2. Run the `ex_rnd.py` example:
   Send >> `python3 ex_rnd.py`
   You should see the two random generated numbers as described above: (1) 16 Bytes, (2) 64 Bytes.



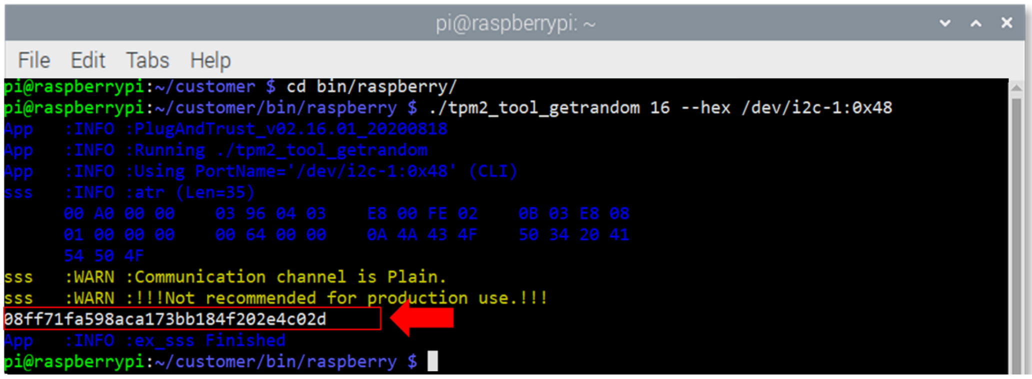
```
pi@raspberrypi:~/customer/ex $ python3 ex_rnd.py
/home/pi/customer/bin/raspberry/tpm2_tool_getrandom 16 --hex /dev/i2c-1:0x48
App   :INFO :PlugAndTrust_v02.16.01_20200818
App   :INFO :Running /home/pi/customer/bin/raspberry/tpm2_tool_getrandom
App   :INFO :Using PortName='/dev/i2c-1:0x48' (CLI)
sss   :INFO :atr (Len=35)
      00 A0 00 00    03 96 04 03    E8 00 FE 02    0B 03 E8 08
      01 00 00 00    00 64 00 00    0A 4A 43 4F    50 34 20 41
      54 50 4F
sss   :WARN :Communication channel is Plain.
sss   :WARN :!!!Not recommended for production use.!!!
bcd8e4c6e7360548e023c77a8c201649
App   :INFO :ex_sss Finished

/home/pi/customer/bin/raspberry/tpm2_tool_getrandom 64 --hex /dev/i2c-1:0x48
App   :INFO :PlugAndTrust_v02.16.01_20200818
App   :INFO :Running /home/pi/customer/bin/raspberry/tpm2_tool_getrandom
App   :INFO :Using PortName='/dev/i2c-1:0x48' (CLI)
sss   :INFO :atr (Len=35)
      00 A0 00 00    03 96 04 03    E8 00 FE 02    0B 03 E8 08
      01 00 00 00    00 64 00 00    0A 4A 43 4F    50 34 20 41
      54 50 4F
sss   :WARN :Communication channel is Plain.
sss   :WARN :!!!Not recommended for production use.!!!
b77416224e58ab0718b369d6281336a2a9a1a6ba6ccd34db0c93029b96223363e77a9ea5a663b60da5cdbf4cbe82ae7431d6
cb1f803fd8987be5b2e02f716689
App   :INFO :ex_sss Finished

pi@raspberrypi:~/customer/ex $
```

Figure 23. Run ex_rnd.py

Another way to run the examples is to directly execute the binary that the `.py` is calling, in this case, `ex_rnd.py` is calling: `tpm2_tool_getrandom`.

Open a Terminal window and follow the steps as shown in Figure 24:

1. Move to the directory containing the binaries in the customer folder:
   Send >> `cd /home/pi/customer/bin/raspberry`
2. Run the `tpm2_tool_getrandom` binary with the desired length as input. In our case we have chosen as length 16 Bytes, then after the function call we write `16 --hex /dev/i2c-1:0x48`:
   Send >> `./tpm2_tool_getrandom 16 --hex /dev/i2c-1:0x48`
   You should see the random number generated.



```
pi@raspberrypi:~/customer $ cd bin/raspberry/
pi@raspberrypi:~/customer/bin/raspberry $ ./tpm2_tool_getrandom 16 --hex /dev/i2c-1:0x48
App    :INFO :PlugAndTrust_v02.16.01_20200818
App    :INFO :Running ./tpm2_tool_getrandom
App    :INFO :Using PortName='/dev/i2c-1:0x48' (CLI)
sss    :INFO :atr (Len=35)
      00 A0 00 00    03 96 04 03    E8 00 FE 02    0B 03 E8 08
      01 00 00 00    00 64 00 00    0A 4A 43 4F    50 34 20 41
      54 50 4F
sss    :WARN :Communication channel is Plain.
sss    :WARN :!!!Not recommended for production use.!!!
08ff71fa598aca173bb184f202e4c02d
App    :INFO :ex_sss Finished
pi@raspberrypi:~/customer/bin/raspberry $
```

Figure 24. Run tpm2_tool_getrandom

5 Appendix A: TPM 2.0 specification and TPM Software Stack (TSS)

The Trusted Platform Module (TPM) was conceived by a computer industry consortium called Trusted Computing Group (TCG). The TPM specification version 1.2 was published in 2011 and, in 2016, TPM 2.0 specification was released, with better support for algorithms and higher cryptographic capabilities.

The TPM 2.0 specification defines the core capabilities and commands of a TPM secure cryptoprocessor. It supports a wide variety of functions, algorithms and capabilities upon which platform-specific specifications are based. For example, the TCG PC Client Platform TPM Profile (PTP) specification defines the additional requirements that a PC TPM shall meet while the TCG TPM 2.0 Mobile Common Profile specification does the same for mobile devices. Regardless of the specific TPM platform implementation, a TPM shall implement the components shown in Figure 25

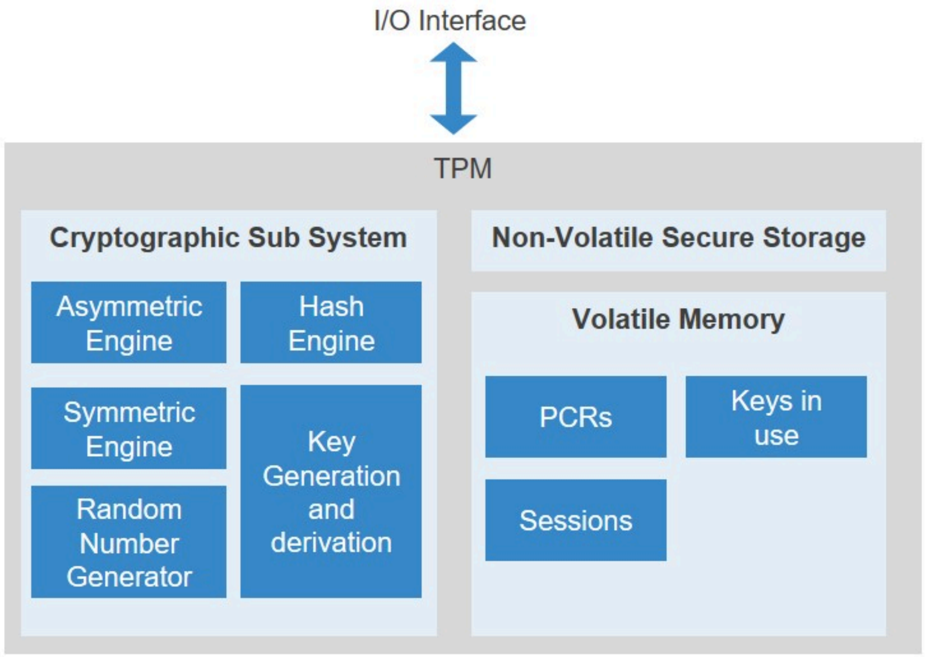


Figure 25. TPM 2.0 architecture

The TPM Software Stack (TSS) defined by the Trusted Computing Group (TCG) provides a standard API for accessing the functions of a TPM. Its objective is to provide a hardware abstraction layer so that the developers can write applications that will work regardless of the underlying hardware, OS or environment used .

The TSS layers are shown in Figure 26. The top layer offers the highest level of abstraction to developers while the bottom layers offer the lowest level of abstraction:

- **Feature API (FAPI):** this API is meant to be a very high-level API, aimed at exposing most of the TPM functions a programmer needs without having to know the low-level implementation details of TPM. This API includes functions like *Fapi_GetRandom ()* to get a random number or *Fapi_CreateKey ()* to create a new key in the TPM;
- **Enhanced System API (ESAPI):** The ESAPI provides a 1-to-1 mapping of all TPM functions and is meant to be used by expert programmers whenever the required functionality is not available as part of the FAPI;

- **System API (SAPI):** this API implements all TPM functions with all possible variations of inputs and outputs. Given its complexity, it is meant to be used by upper layers (FAPI and ESAPI) and not directly by programmers;
- **TPM Command Transmission Interface (TCTI):** this interface handles all the communication to and from the lower layers of the TSS. Different interfaces are needed for different TPM implementations, for example of a hardware TPM and firmware TPM.

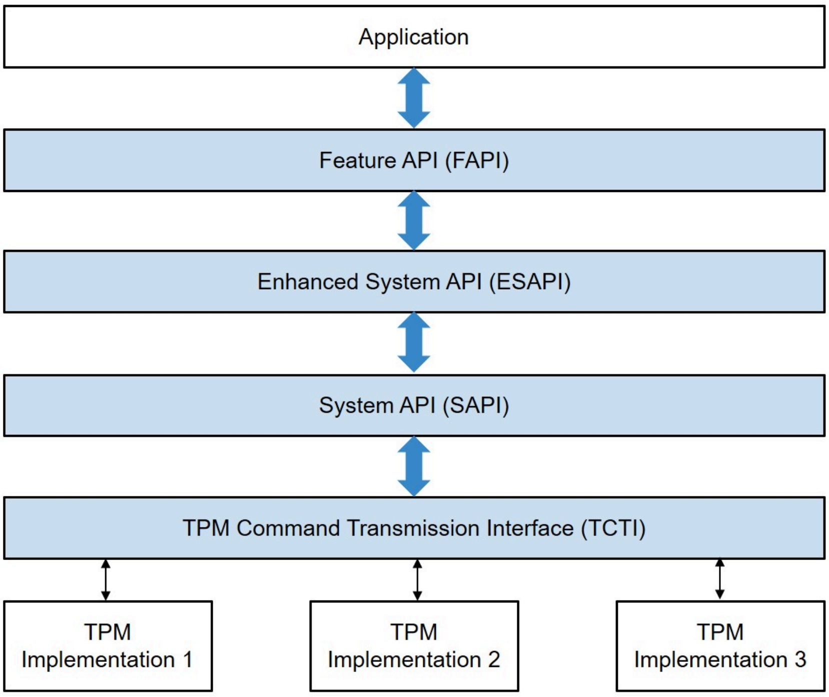


Figure 26. TSS architecture

An open source implementation of the TSS in available at https://github.com/tpm2-software/tpm2-tss

6 Appendix B: TPM unsupported features or constraints

Some limitations apply to Plug & Trust middleware TSS support:

Table 5. Limitations and unsupported features of Plug & Trust middleware TSS integration

Domain	Limitation
TPM Hierarchies	Hierarchies are not supported.
Object injection	It is recommended to use SE05x or SSS APIs instead of TSS to inject objects. This allows you to set the access policies on the injected object.
Object policies	Object policies cannot be changed after object injection in EdgeLock SE05x which includes values of PCRs. The TPM standard allows updating policies via authorization, so TPM APIs related with object policies have been disabled.
API Compatibility for Polices of TPM	TPM supports policies like Simple assertion, Multi-assertion and Compound policies. The only practical policy implementation in EdgeLock SE05x is using PCRs.
Context management	In EdgeLock SE05x users can export only transient Secure Contexts to a non-trusted environment.
Audit commands	TPM audit commands are not supported.
Non-volatile commands	Non-volatile TPM APIs are not implemented in the current build of the project

7 Legal information

7.1 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.

7.2 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.

7.3 Trademarks

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

NXP — wordmark and logo are trademarks of NXP B.V.

EdgeLock — is a trademark of NXP B.V.

Tables

Figures

Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

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

Date of release: 13 April 2021
Document identifier: AN12663