AN12906

HSM and SHE on i.MX 8QXP and i.MX 8DXL

Rev. 0 — June 2020

Application Note

Contents

1 Introduction

1.1 Purpose

Hardware Secure Module (HSM) and Secure Hardware Extension (SHE) are two security services for NXP products, that supports in various types of automotive applications. This document provides an initial knowledge of the architecture of these services and helps to develop an application using the HSM and SHE APIs.

The HSM architecture is compatible with only i.MX 8QXP Rev C0 and i.MX 8DXL.

1.2 Audience

This document is targeted for i.MX 8 and 8X family (excluding the i.MX 8M families), and provides the information of:

- The architecture of a system running HSM or SHE services
- Development of an application which utilizes HSM or SHE services

The user must be familiar with basic cryptography principles such as symmetric-key cryptography, certificate signing, and hashing.

1.3 Acronyms and abbreviations

Table 1. Acronyms and abbreviations

Term	Description
HSM	Hardware Secure Module
SHE	Secure Hardware Extension
SCU	System Controller Unit
SCFW	System Controller Unit Firmware
SECO	Security Controller
SECO FW	Security Controller Firmware
MU	Messaging Unit
CAAM	Cryptographic Accelerator and Assurance Module
ECC	Elliptic-Curve Cryptography

Table continues on the next page...

Table 1. Acronyms and abbreviations (continued)

Term	Description
RSA	Public-key cryptosystem which is widely used for secure data transmission
RNG	Random Number Generation
DID	Domain ID
MAC	Message Authentication Code
NVM	Non-Volatile Memory
TZ	TrustZone
V2X	Vehicle-to-everything

2 Overview

HSM and SHE are the two Security Controller Firmware (SECO FW) components that are accessed by using the seco_libs API and provides security features to various kinds of applications. HSM is developed to support V2X use cases and SHE is an implementation of the SHE specification which provides security features to automotive applications. SHE is part of the baseline SECO FW available on all i.MX 8 and 8X family (excluding the i.MX 8M families), and HSM is an optional extension to these services. The presence of the HSM extension does not impact the availability of all the other baseline services.

Security Controller (SECO) runs on a dedicated M0+ core. It handles critical security tasks on behalf of the rest of the system, with which it communicates using MUs.

System Controller Unit (SCU) runs on a dedicated M4 core and configures the platform at start-up. It also manages the available resources for the Arm Cortex-A and Cortex-M cores.

3 HSM

3.1 Architecture

The architecture of HSM depends on the system on which it is deployed and the location where the user accesses the HSM services. HSM is composed of a SECO FW component and a series of additional OS-independent components that must be ported to the environment in which the user accesses the HSM services. Porting of these components has already been done for the NXP Linux distribution to demonstrate HSM usage. The architecture described in this document and schematized in Figure 1, refers to this HSM demonstrator on Linux running on i.MX 8QXP Rev C0.

The architecture of the HSM demonstrator on Linux comprises four main components:

- The HSM services provided by SECO. These additional services are offered alongside the baseline services inside the SECO FW.
- The HSM kernel driver integrated in the Linux BSP. It is used to access the MUs that allows the communication between the users core and SECO.
- The HSM storage manager. It provides storage services to SECO to preserve persistent data across power cycles. In this case, the storage manager provides the services through the Linux file system driver. The user can choose to modify the storage manager abstraction layer to support a different type of non-volatile storage and the methods to access it.
- The HSM Lib. It provides the HSM features API to the user application.

NOTE

During HSM communication between the user core and SECO, MUs support only the control channel. The data channel is represented by a memory partition shared between the user core and SECO, which the two can access independently.

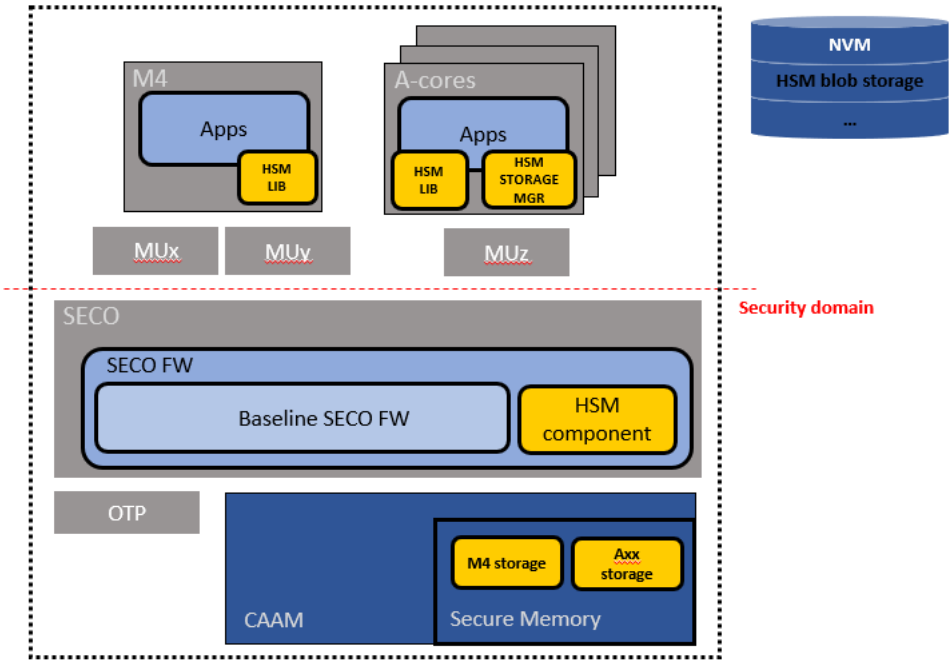

Figure 1. HSM architecture

3.2 Services

All the HSM operations (in yellow in Figure 2) are accessed using a handle (in blue in Figure 2) related to the services available. Services are organized hierarchically (as in Figure 2), that is, to open a new child service, the handle of the parent service is required. For example, to open a Cipher service to perform encryption/decryption, the user must provide the handle of an open Key Store service.

The first handle is required to identify an open session with HSM. The services for Signature Verification, Hashing, and RNG can be open by using a session handle without authentication. To open a Key Store service, authentication request is required that is based on the Domain ID (DID), MU ID, the user-provided Key Store identifier, and related nonce.

The hardware-level authentication is based on the DID and MU ID that allows only a user running on the expected core and using the correct MU to perform operations. The provided nonce assures that the SECO cannot access the key store content without a valid user request. Ensure to setup the platform partitioning by assigning a unique DID to each core and provide each domain the exclusive access to its MU. The authentication is performed only once, during the opening of the Key Store service. If the authorization is successful, the Key Store handle is returned and the user can access all the operations and child services of the Key Store.

NOTE

For any key store present inside HSM, only one Key Store handle can be provided to the user.

All the HSM operations that involve a secret, such as Key Management, Ciphering, Signature Generation, Secure Data Storage, and MAC are accessed through the Key Store.

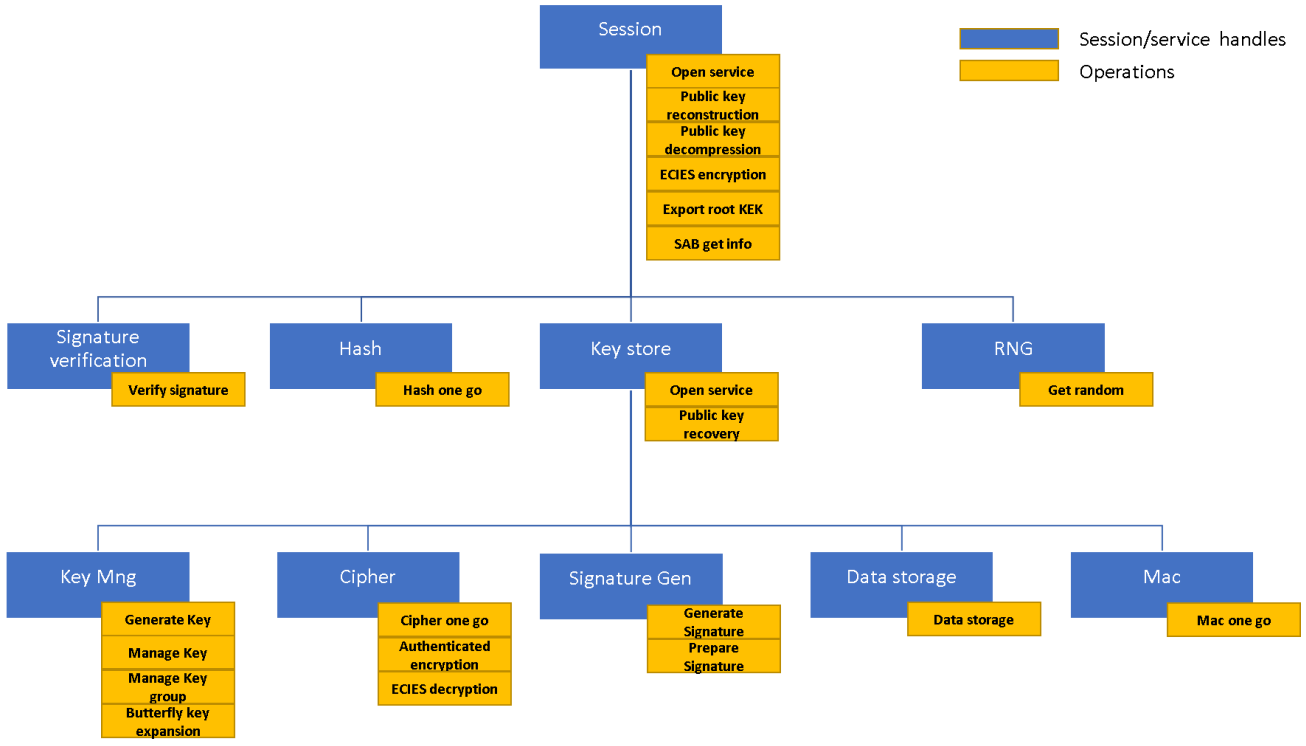

Figure 2. Services and related operation offered by HSM

3.3 Non-Volatile Memory (NVM) manager

The NVM manager is a user-space component of HSM used by the SECO HSM services to store permanent data across power cycles. SECO does not have direct access to permanent storage, therefore it relies on an external source to support this feature. Data is stored as encrypted blobs, in the non-volatile storage that the user decided to support in the abstraction layer of the NVM manager. The NVM manager must be only one on the system, it is subordinated to SECO requests and there is no specific domain in which it should run. In the provided example, the NVM manager is launched as a separate thread on the Cortex-A core having an underlying Linux distribution, which provide the file system driver used to store the encrypted blobs as files in the root partition.

3.4 Partitioning the domains on the board

For successful authentication during the opening of the Key Store service, the request must have the correct DID, MU ID, and TZ setting. The user must address the following characteristics to correctly partition the system resources.

1. Each core (for example, M4)/core cluster (for example, core-A cluster) in the system is associated with a unique DID
2. Each of the 4 MUs used by SECO, need to be associated to one domain (for example, MU0 to SCU, MUx to M4, MUy to the core-A cluster, and so on)

This setup is performed by the SCU and can be tuned by the user by modifying the `board_system_config` routine inside the board.c file of the SCU FW.

NOTE

To exchange the data during HSM operations it is required that a memory area (for example, DDR) is shared between the user and the SECO. This can also be done in the `board_system_config` routine inside the board.c file or by using the SCFW API.

4 SHE

4.1 Architecture

Similar to HSM, the architecture of SHE depends on the platform on which it is deployed and on the location where the user accesses the SHE services. SHE is composed of a SECO FW component and a series of additional OS-independent components that must be ported to the environment in which the user accesses the HSM services. A porting of these components has been already done for the NXP Linux distribution to demonstrate SHE usage. The architecture described in this document and schematized in Figure 3, refers to this SHE demonstrator on Linux running on i.MX 8QXP Rev C0.

The architecture of SHE Linux demonstrator includes four main components as shown in Figure 3:

- The SHE services provided by SECO. It is a part of the baseline SECO FW.
- The SHE kernel driver integrated in the Linux BSP. It is used to access the MU that allows the communication with SECO.
- The SHE storage manager. It provides storage services to SECO to preserve persistent data across power cycles. The storage manager provides such services through the file system driver of the OS on which it is deployed.
- The SHE Lib. It provides the user application access to SHE features.

NOTE

The MU is used only for communication between the User Domain and SECO for control, and the data is exchanged using a shared memory partition.

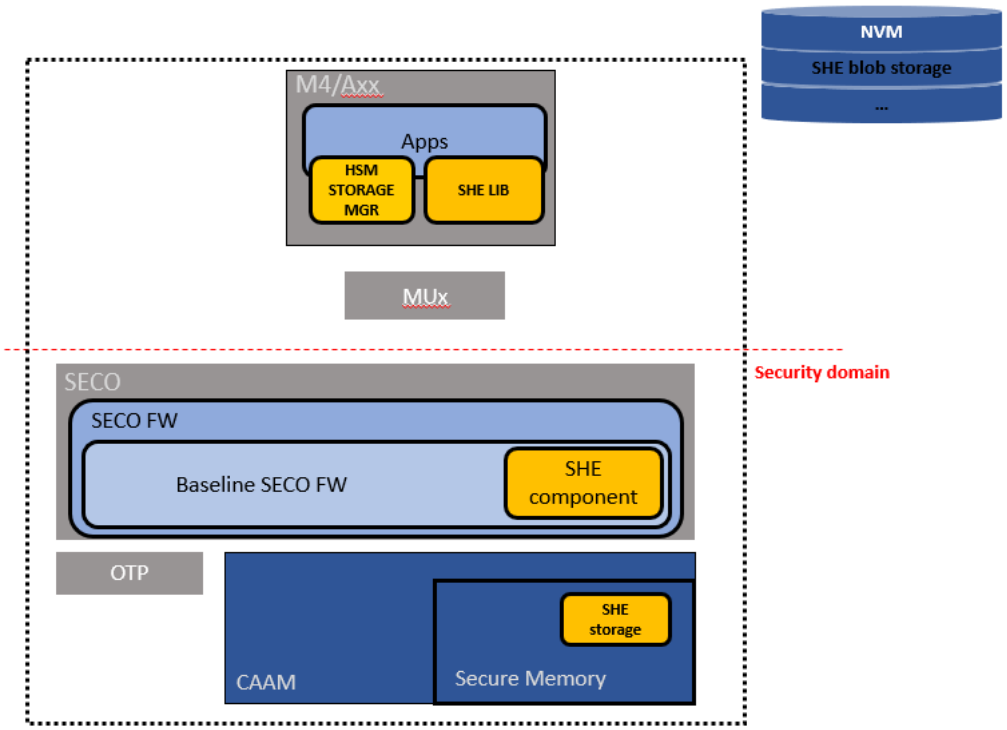

Figure 3. SHE architecture

NOTE

SECO releases prior to 2.6.x are not suitable to support production SHE implementations.

4.2 Services

SHE services are limited with respect to HSM and are accessed using the same session handle as shown in Figure 4. When opening a session, an authentication is performed based on the DID, MU ID, a SHE storage identifier and the related password. The SHE storage identifier and the related password are provided by the user during the creation of the SHE storage. The SHE storage is unique in the system and always must be created before calling any other SHE API.

Using a session handle it is possible to perform message authentication using AES-CMAC, key management (storage, deletion and update of internal keys), encryption and decryption using AES in ECB or CBC mode, and random number generation.

Message authentication and encryption/decryption are performed by using internal non-volatile keys. These keys, named KEY_<n> (where n is a number from 1 to 10, with possible extensions to support up to 50 keys), can be used for only one function, and can be selected at the time of loading between message authentication and encryption/decryption.

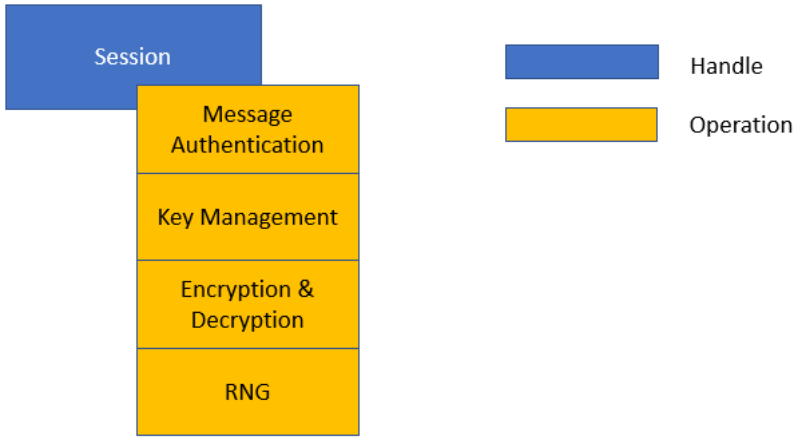

Figure 4. Handle and related services offered by SHE

4.3 SHE Storage manager

The SHE storage manager allows SECO to store permanent data across power cycles in the form of encrypted blobs. The storage manager is OS independent and depends on an abstraction layer that the user must modify to support the desired type of non-volatile storage. As the HSM NVM manager, it needs to be unique in the system and can be run in the user-preferred domain (A-core cluster, M4, or SCU). In the provided Linux demonstrator, the storage manager is run as a separate thread in the Linux OS running on the A-core cluster, using the file system API to support non-volatile storage inside the root partition.

4.4 Partitioning the domains on the board

SHE uses the same authentication mechanism of HSM to provide access to key storage. The authentication mechanism is based on identifying a user request, firstly by its DID and MU ID, and then on a provided storage ID and password. To set up the board to securely access SHE services, the user needs to:

- Assign to each core/core-cluster a unique DID
- Associate each of the four SECO MUs to one core/core-cluster

This setup is performed by the SCU and can be tuned by the user modifying the `board_system_config` routine inside the board.c file of the SCU FW.

NOTE

To exchange the data during SHE operations it is required that a DDR memory area is shared between the user and the SECO. This can also be done in the `board_system_config` routine inside the board.c file or by using the SCFW API.

5 Setup

The setup provided aims at creating a system supporting both HSM and SHE to run the two examples described in the following paragraphs. The HSM architecture is only compatible with i.MX 8QXP Rev C0 and i.MX 8DXL. The following steps describe setting up of the system with the correct Linux BSP containing the SECO HSM drivers and HSM-compatible SECO FW.

1. Use Yocto to create the bootable SDCARD with the Linux distribution based on 5.4.3-2.0.0:

```
$: mkdir <work>
$: cd <work>
$: mkdir <release>
$: cd <release>
$: repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-linux-zeus -m
imx-5.4.3-2.0.0.xml
$: repo sync
```

2. Setup the build directory selecting your <machine> between imx8qxpc0mek and imx8dxlevk:

```
$: MACHINE=<machine> DISTRO=fsl-imx-wayland source ./imx-setup-release.sh -b
<hsm_build_directory>
```

3. Add development features to the build, by inserting the following two lines in the file <release>/<hsm_build_directory>/conf/local.conf :

```
EXTRA_IMAGE_FEATURES_append = " dev-pkgs tools-sdk tools-debug "

IMAGE_INSTALL_append = " git "
```

4. Launch the build:

```
$: bitbake imx-image-core
```

5. Flash the bootbale SDCARD file, by substituting /dev/sdX with the device name of your SD CARD:

```
$: bzcat tmp/deploy/images/<machine>/imx-image-core-<machine>.wic.bz2 | sudo dd of=/dev/sdX
bs=1M conv=fsync && sync
```

6. Boot the board from SD card and execute the following steps from the target. Install seco_libs userspace libraries:

```
$: cd <work>
$: git clone https://github.com/NXP/imx-seco-libs.git
$: cd imx-seco-libs
$: git checkout imx_5.4.3_2.0.0
$: make install
```

5.1 HSM example application

An example program is provided with this application note to exercise the HSM features on a platform that meets the requirement of the setup described in the previous section. The example illustrates how to launch the NVM manager, create a key store, internally generate an AES key, and use it to encrypt and decrypt user-provided data.

Retrieve the example code inside the imx_sec_apps repository containing other security related projects:

```
$: git clone https://source.codeaurora.org/external/imxsupport/imx_sec_apps.git
$: cd imx-sec-apps/hsm_she_examples
```

In the Makefile, enter the location of seco_libs

```
SECO_LIBS_DIR = <work>/imx-seco-libs
```

Compile and launch the example:

```
$: make all DEBUG=y
$: ./hsm_test [-n or --no-create] [keystore_identifier]
```

Launch the application with `-n or --no-create` flag when a key store has already been created. Optionally, provide a key store identifier in the form of a 32bits hex, otherwise a default one will be used. If the user wants to clean the NVM state, then delete the related files used by the HSM Linux demonstrator (as below), followed by a board reset:

```
$: rm -rf /etc/seco_hsm
```

See the README available here, which contains further details on the available option and the operations performed during the example.

5.2 SHE example application

An example program is provided to show the usage of the SHE APIs. The example illustrate how to launch the SHE storage manager, create an empty SHE storage, load a key, and use it for encryption and decryption user-provided data.

Retrieve the example code from the same repository containing the HSM example described in the previous section. Compile and launch the example:

```
$: make all DEBUG=y
$: ./she_test [-n or --no-create]
```

Launch the application with `-n or --no-create` .

If the user wants to clean the NVM state, then delete the related files used by the SHE Linux demonstrator (as below), followed by a board reset:

```
$: rm -rf /etc/seco_she_nvm
```

The repository containing the SHE example is the same as that containing the HSM example. Therefore, see to the indications in HSM example application to clone and compile the example. Also, see the README in the same repository to discover further details on the performed operations.

6 Revision history

Table 2. Revision history

Revision number	Date	Substantive changes
0	06/2020	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: June 2020

Document identifier: AN12906

arm