AN14670

EdgeLock 2GO Provisioning via SPSDK for MCUs

Rev. 1.1 — 25 March 2026 Application note

Document information

Information	Content
Keywords	AN14670, EdgeLock 2GO, EdgeLock
Abstract	This document offers an outline of the EdgeLock 2GO platform and discusses the “Device provisioning via proxy” flow. In this case, the SPSDK command line tool is the proxy being used to provision an MCU.

1 Introduction

EdgeLock 2GO is a fully managed cloud platform operated by NXP that provides secure provisioning services for easy deployment and maintenance of IoT devices that integrate NXP MCU, MPU, and EdgeLock SE05x secure elements.

EdgeLock 2GO Managed is a service that allows you to create and manage secure objects, such as symmetric roots of trusts, key-pairs, and certificates, which are then securely provisioned by EdgeLock 2GO Managed into the MPU or MCU based IoT devices.

Secure objects and certificates created through EdgeLock 2GO Managed can be used for a wide variety of use cases, including for secure cloud onboarding of IoT devices to your preferred cloud service (for example, AWS IoT Core or Azure IoT Hub), data encryption or decryption, access control, and so on.

EdgeLock 2GO Managed can be easily used and configured through the EdgeLock 2GO web portal or the EdgeLock 2GO REST API.

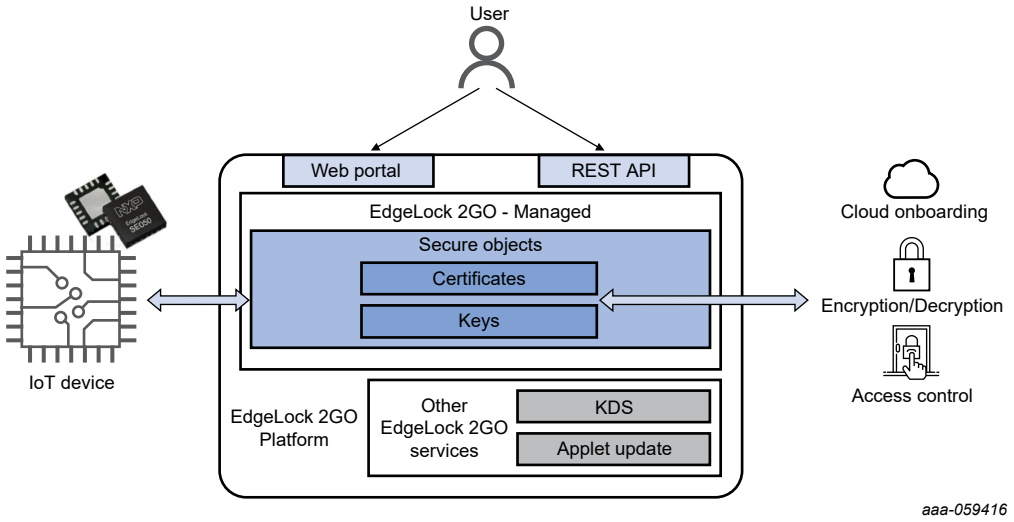

Figure 1. EdgeLock 2GO platform overview

For more details about EdgeLock 2GO, see EdgeLock 2GO homepage and the documentation available at EdgeLock 2GO Portal (account/sign-in required).

2 Scope and objective

This document offers an outline of the EdgeLock 2GO platform and discusses the “Device provisioning via proxy” flow. In this case, the SPSDK command line tool is the proxy being used to provision an MCU.

The document focuses on the initial device provisioning using secure objects from the EdgeLock 2GO cloud server.

Note: *The screenshots used in this application note show a variety of product families and part numbers. Always make sure to select the device family and device part numbers according to the actual device you are using.*

3 Prerequisites

3.1 Hardware procurement

During manufacturing, the NXP SoC are injected with EdgeLock 2GO credential and registered in the EdgeLock 2GO pool of devices. Such NXP parts can be procured at nxp.com.

Procure the device among the ones listed under **Supported Products** at EdgeLock 2GO homepage at nxp.com.

3.2 EdgeLock 2GO account creation

To access the EdgeLock 2GO cloud server, create an account by requesting access.

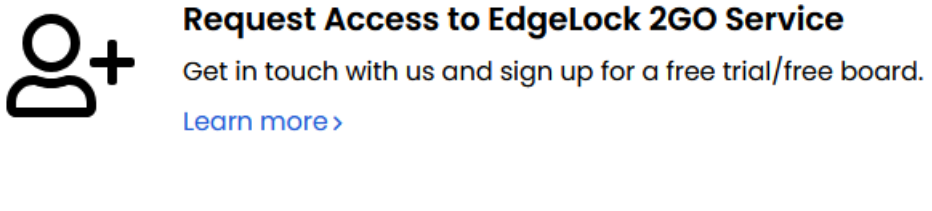

Figure 2. Requesting access to the EdgeLock 2GO cloud server

For more details, see the **Logging in the EdgeLock 2GO – Managed account** section in the *EdgeLock 2GO – Managed* (document AN12691) mentioned in Section 8 "References".

3.3 EdgeLock 2GO account login

Once an account is created in the EdgeLock 2GO Managed server using the email address you provided, you can use an NXP registered account associated with that email to log in to EdgeLock 2GO Portal.

Figure 3. Account login

3.4 Detect EdgeLock 2GO enabled hardware

After procuring the Evaluation Kit/Board or Part, the device can also be verified to be EdgeLock 2GO enabled by performing the following steps:

1. Determine the UUID as described in Section 9 "Appendix A UUID retrieval".
2. Log in to the EdgeLock 2GO Portal.
3. Create a device group.
   Click **Devices**; **MCU & MPU**; **New Device Group**, as shown in Figure 4.

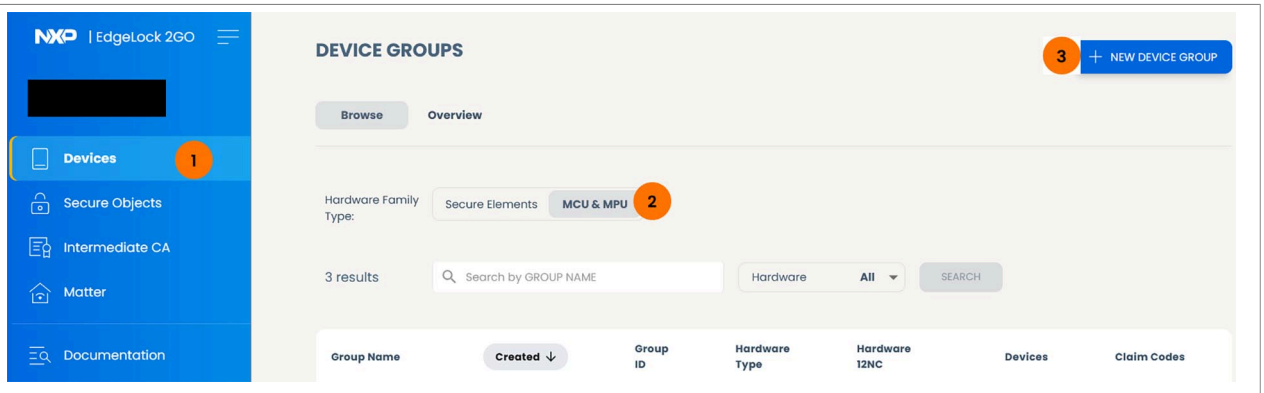

Figure 4. Create a device group

4. Verify the device.

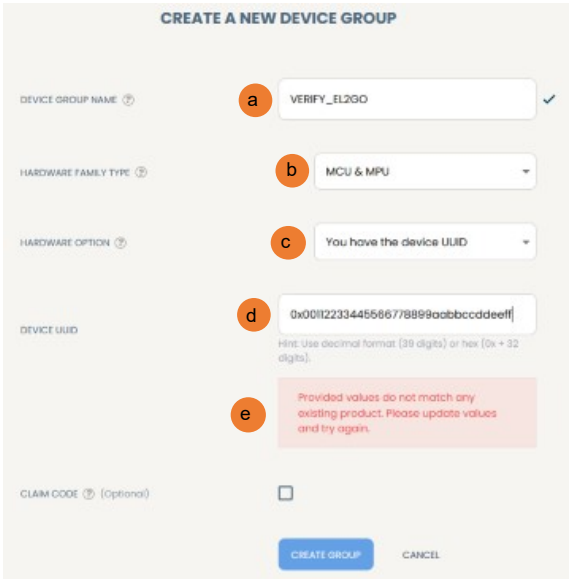

Figure 5. Verify the device

For the detailed information to input, see Figure 5.

a. Give a sample name to the **Device group name** field.
b. Choose **MCU & MPU** in the **Hardware Family Type** field.
c. Choose **You have the device UUID** in the **Hardware Option** field.
d. Paste the device UUID in the **Device UUID** field and prepend with `0x`.
e. If the device is registered in the EdgeLock 2GO Managed server, then the Error in red is not shown.

3.5 Install SPSDK and restricted data

Secure Provisioning SDK (SPSDK) is an open-source development kit with its source code released on Github and PyPI. It contains a set of API modules for custom production tool development which requires more advanced secure provisioning flow. The SPSDK also supports provisioning using EdgeLock 2GO. To follow the steps shown in this document, install the latest version of SPSDK.

After SPSDK is installed, the restricted data package is needed for the tool, because it contains device-specific provisioning firmware that the SPSDK tool must provision EdgeLock 2GO secure objects. You must be approved to download the restricted data, so make this request before getting started.

3.6 Kleopatra install and setup

To create an EdgeLock 2GO secure object for provisioning the SB3 key (`CUST_MK_SK`), encrypt the **SB3** key before it is uploaded to the EdgeLock 2GO server.

Different tools can be used for generating the encryption keys used for this purpose, but Kleopatra is the recommended Windows tool. It can be installed as part of the Gpg4win tool.

3.6.1 Kleopatra key setup

After installing the tools, open Kleopatra. When you open Kleopatra for the first time, it prompts you to create or import a key pair. Pick the **New Key Pair** option.

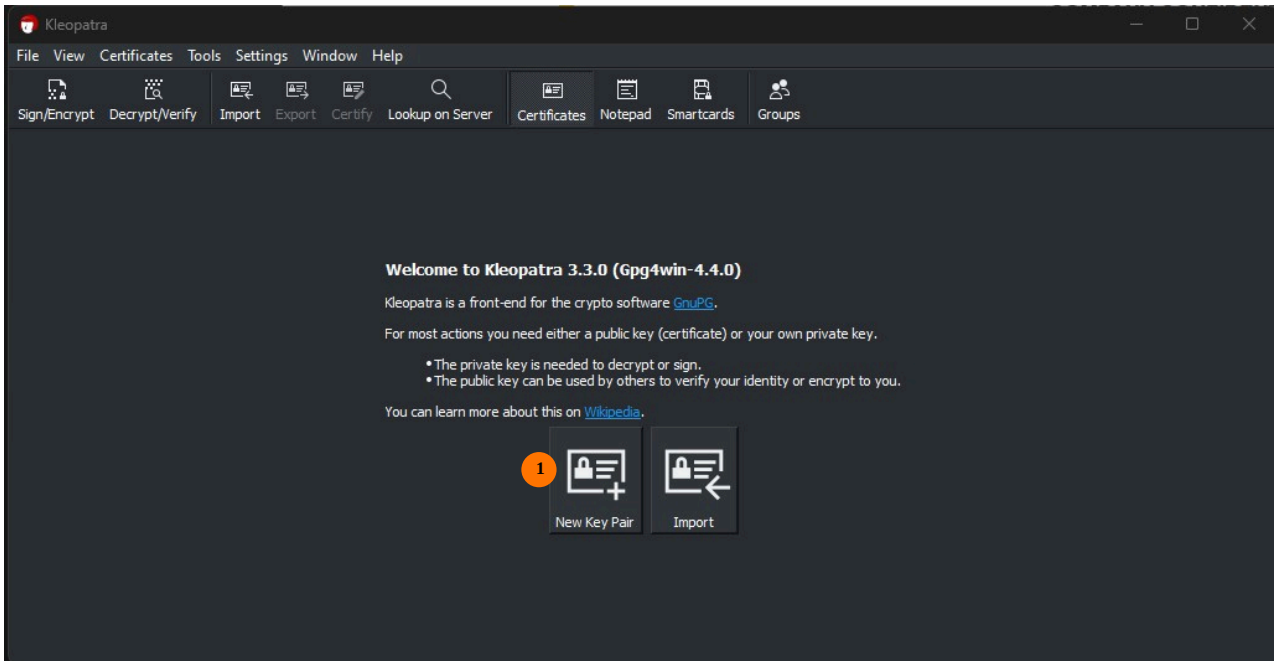

Figure 6. New Key Pair option

If regenerating keys or performing this step later, use the **File**; **New OpenPGP key pair** menu option.

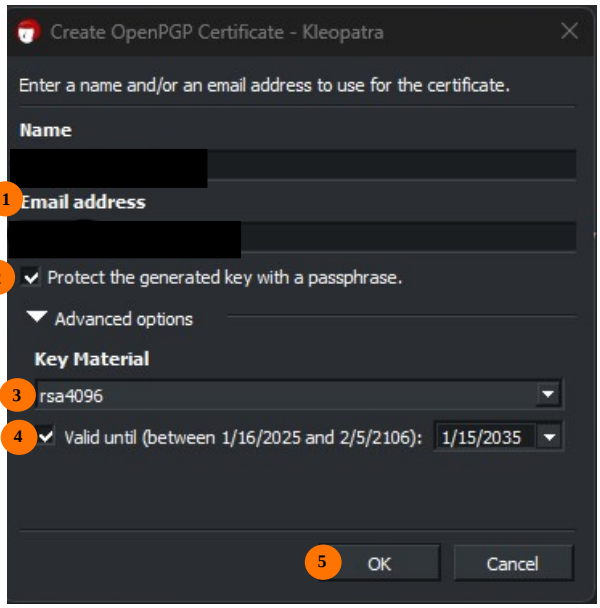

Figure 7. Create OpenPGP Certificate - Kleopatra dialog box

For the detailed information to input, see Figure 7:

1. Name and email address are autofilled. Fill them in or change them if needed.
2. Check the **passphrase** option. It is optional, but recommended.
3. Expand the advanced options and pick **rsa2048** or **rsa4096**. These are the only key options supported for EdgeLock 2GO key signing.
4. Pick an expiration date for the key pair. Recommend to select a date beyond the default expiration.
5. Click **OK** to generate the key pair.

After you click **OK**, a new window pops up, where you specify the passphrase for the key, as shown in Figure 8. Remember the passphrase, which you will use in the following steps.

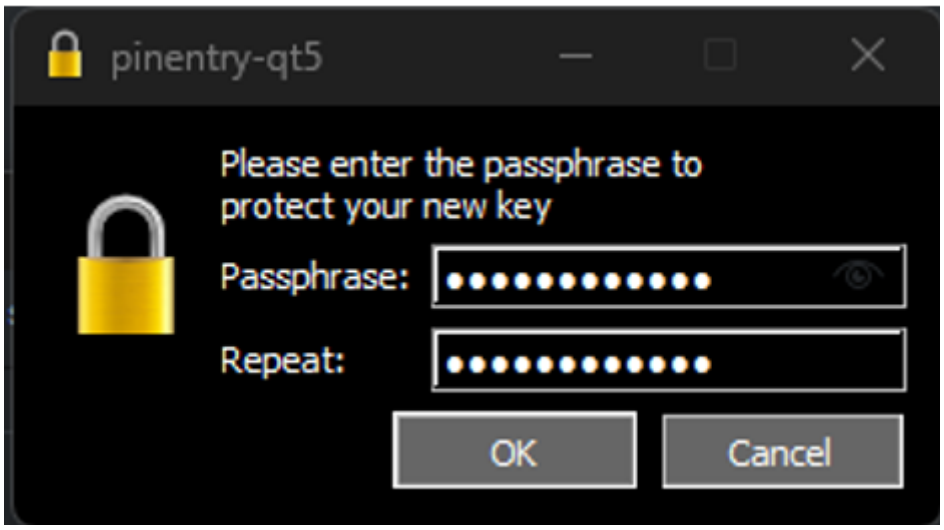

Figure 8. Passphrase window

Open a command prompt window. If Kleopatra is installed for all users, then the directory you use to open the window does not matter.

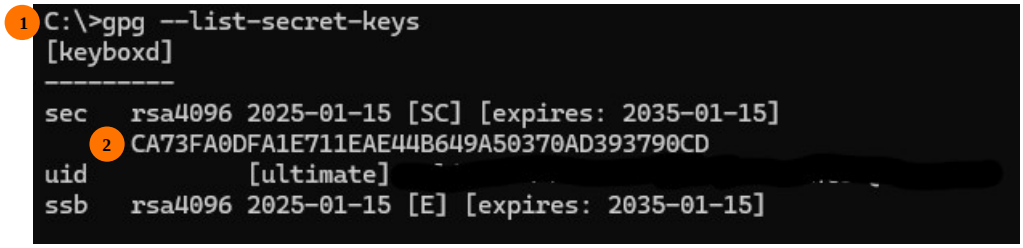

Figure 9. Get key hash

To obtain the hash of the key, which is needed for future steps, perform the following steps, as shown in Figure 9:

1. At the command prompt, type **gpg --list-secret-keys**.
2. Note the hash value which is returned. You'll use this value in the following steps.

The EdgeLock 2GO public key must be imported so that it can be used within GPG. Download the public key from the EdgeLock 2GO environment.

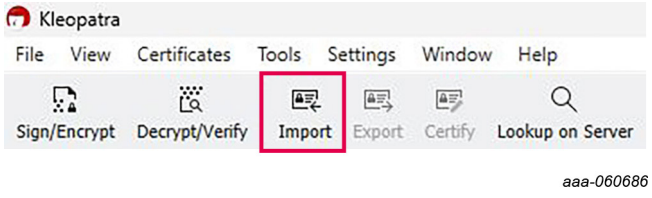

Figure 10. Import EdgeLock 2GO public key

To import the EdgeLock 2GO public key, perform the following steps:

1. Click import at the top of the tool options inside Kleopatra.
2. Select the **EL2GO_public_key.asc** file from the location on your host PC.

4 EdgeLock 2GO provisioning flows

For an overview on different EdgeLock 2GO services supported on MCUs and MPUs, see the *EdgeLock 2GO Services for MPU and MCU* (document AN14544) mentioned in Section 8 "References".

5 Device provisioning via proxy

When using the Provisioning via Proxy flow, a host running SPSDK or SEC is used to facilitate provisioning. The secure objects to be provisioned are loaded from the EdgeLock 2GO server to the host. Then they are moved from the host to the device.

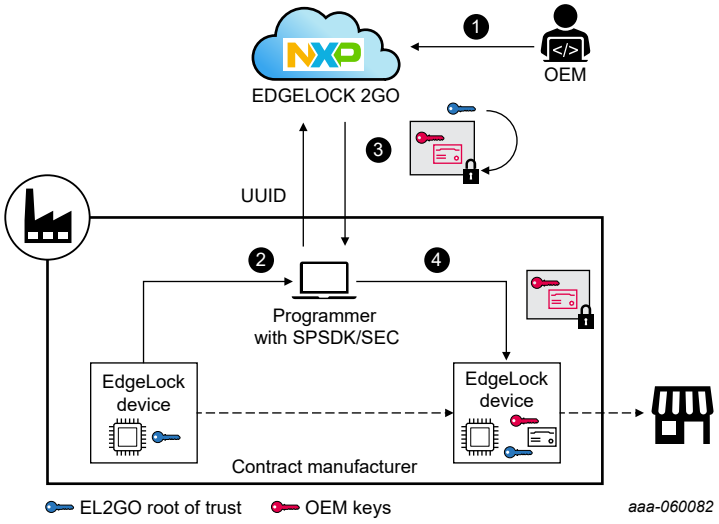

Figure 11. Provisioning via Proxy flow

To provision via the proxy, perform the following steps, as shown in Figure 11:

1. OEM configures secure objects via the EdgeLock 2GO portal or API.
2. At device manufacturing, readout device UUIDs (one by one or by batch) and send the UUID or list of UUIDs to EdgeLock 2GO over API.
3. EdgeLock 2GO generates the secure objects configured in step 1 and encrypts with EdgeLock 2GO root of trust. The encrypted secure objects are downloaded to the host.
4. The host loads the encrypted secure objects to the device where EdgeLock 2GO provisioning firmware passes the encrypted credentials to the EdgeLock secure enclave to unwrap them. Internal secure objects are installed into fuses/IFR. External secure objects are rewrapped and written to flash.

Note: *EdgeLock 2GO supports REST API with which user can interact with EdgeLock 2GO and perform same steps as below. Using REST API is out of the scope of this document. For detailed information, see the EdgeLock 2GO Developer Guide (document AN12642) mentioned in Section 8 "References".*

6 Using SPSDK and EdgeLock 2GO to provision a device

This section describes the steps to configure the EdgeLock 2GO server and SPSDK command line tool for device provisioning.

6.1 Create a device group

The EdgeLock 2GO server uses device groups to organize data. The device group includes information on the part number for the devices that are provisioned as part of the group. Later, add the secure objects to the group.

To create a device group in EdgeLock 2GO cloud server, perform the following steps:

1. Click **Devices**; **MCU & MPU**; **New Device Group**, as shown in Figure 12.

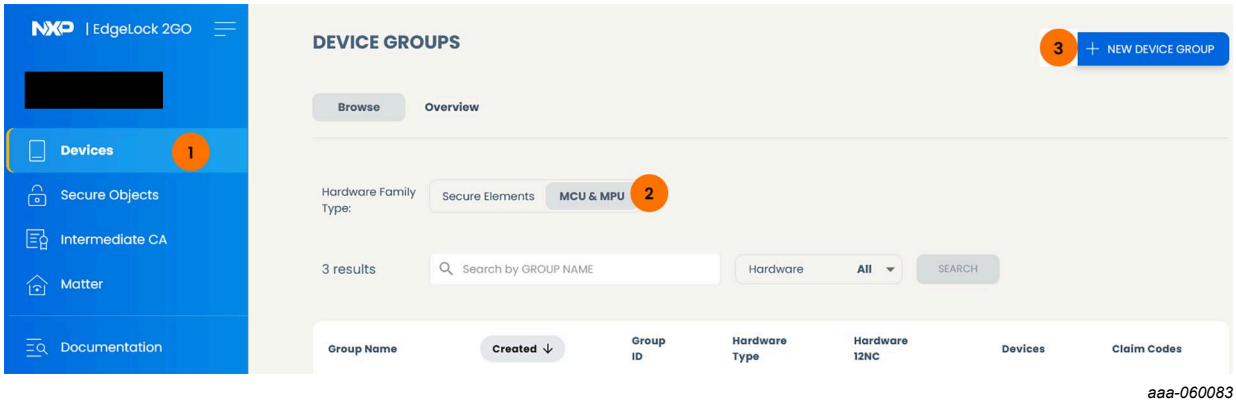

Figure 12. Create a device group

2. A dialog box pops up, as shown in Figure 13.

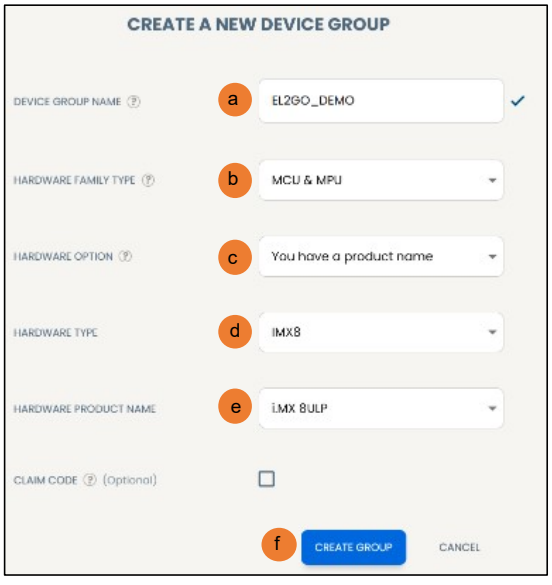

Figure 13. Create A new Device Group dialog box

For the detailed information to input, see Figure 13:

a. Input a custom **Device Group Name**.
b. Select **Hardware Family Type** as **MCU & MPU**.
c. The **Hardware Option** emerges, and select **You have a product name**.

d. Select the appropriate **Hardware Type**.
e. Select **Hardware Product Name**.
f. Click **Create Group** to create the device group, in which all new devices have been registered.

Some of the values from the device group are needed later when configuring the SPSDK tool to communicate with the EdgeLock 2GO server. After creating the device group, the Device Groups screen is opened. If not, or to go back to the device group, click **Devices**; **MCU & MPU**.

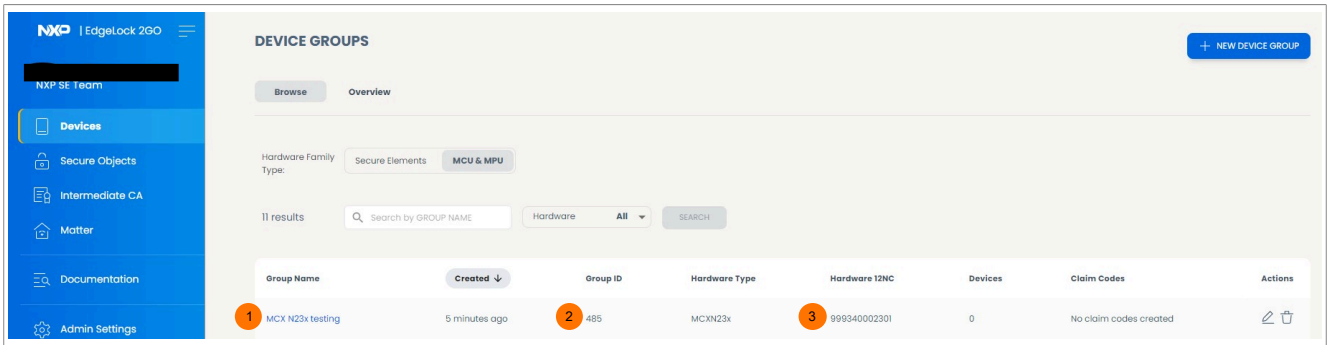

Figure 14. Device Groups interface

To configure the SPSDK tool to communicate with the EdgeLock 2GO server, perform the following steps, as shown in Figure 14:

1. Find the name of your device group in the list.
2. Copy the **Group ID** number for use later.
3. Copy the **Hardware 12NC** number for use later.

6.2 Create an API key

The SPSDK communicates with the EdgeLock 2GO server using REST APIs. The API key is a random sequence of characters that uniquely identify an EdgeLock 2GO customer. The API key is sent as a parameter with the REST APIs for authorization.

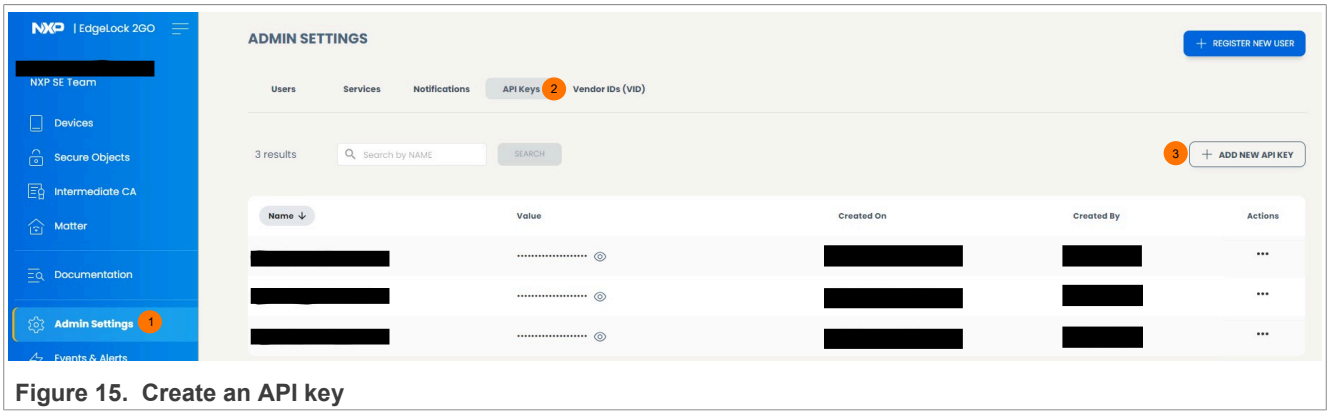

Figure 15. Create an API key

To create an API key, perform the following steps, as shown in Figure 15.

1. Click **Admin Settings**.
2. Select **API Keys**.
3. Click **Add New API Key**.

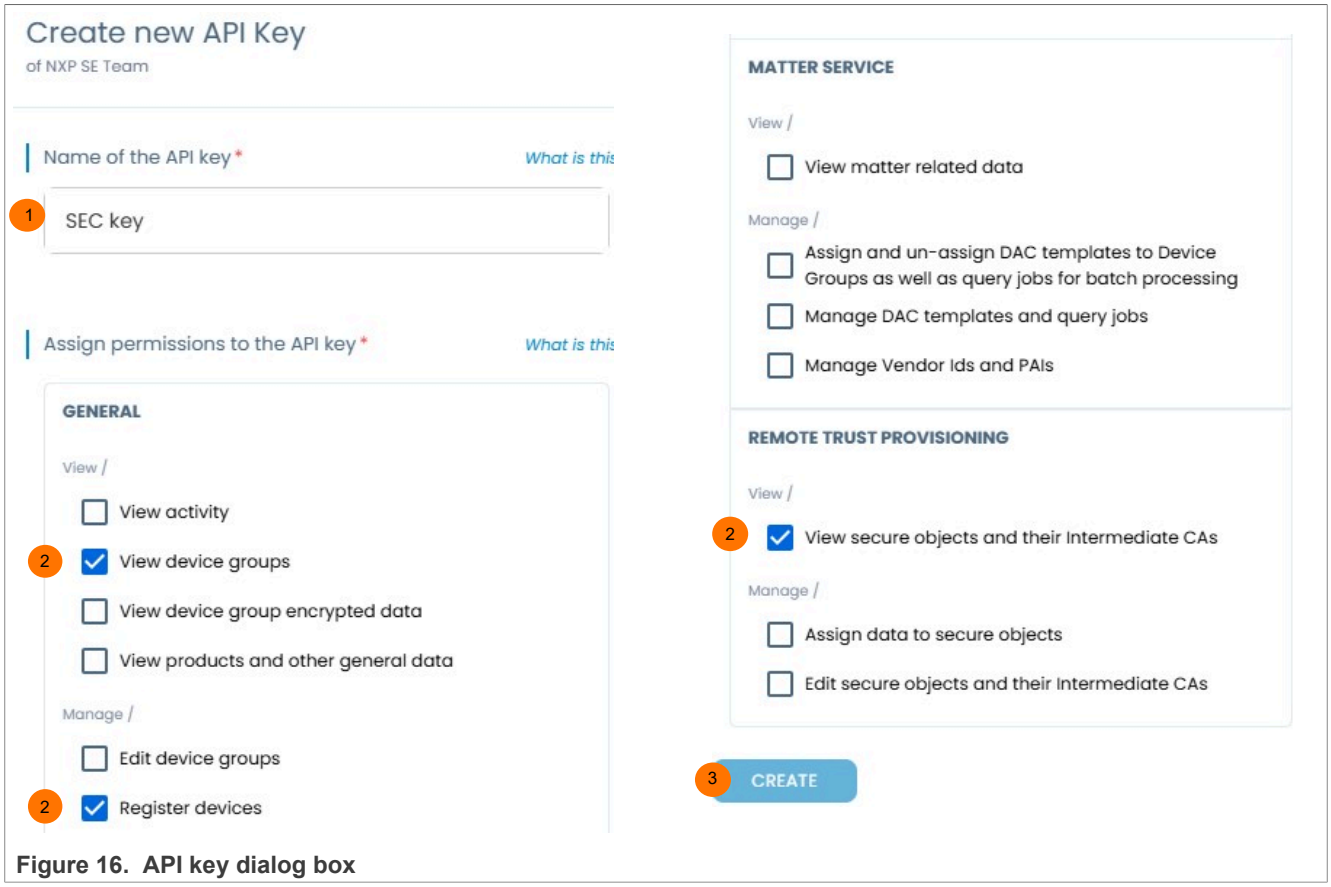

Figure 16. API key dialog box

For the detailed information to input, see Figure 16.

1. Input a name for the API key
2. Set the permissions for the API key:
   - View device groups
   - Register devices
   - View secure objects and their intermediate CAs

   The values must be selected at minimum for the rest of the steps shown in the document to complete successfully.
   Note: *The **View device groups** permission is required for testing the EdgeLock 2GO connection. It is optional for manufacturing packages.*
3. Click **Create** to generate the API key.

The API key value is needed later when configuring the SPSDK tool to communicate with the EdgeLock 2GO server. After creating the API key, you are in the API keys screen. If not, or to get the API key value later, click **Admin Settings**; **API Keys**.

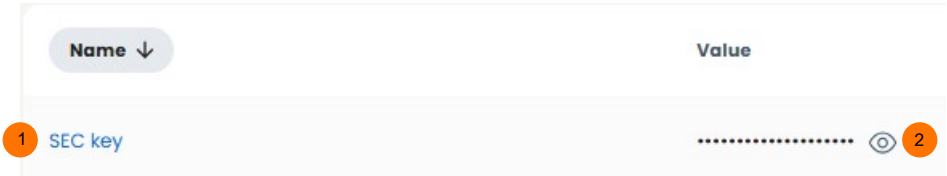

Figure 17. API key value

For the detailed information to input, see Figure 17.

1. Find the name of your API key in the list.
2. Click the eye icon to show the API key value. After clicking the eye icon, a copy to clipboard button pops up, which can be used to save the value.

6.3 Configure SPSDK for EdgeLock 2GO provisioning

The following steps describe how to configure the SPSDK tool to communicate with the Device Group you created on the EdgeLock 2GO server using the API key value you also created.

```
#                                          == General Options ==
# ============================================================================================================
# ------------------------------------------------===== MCU family [Required] =====----------------------------------------
# Description: Family identifier including the chip revision. If revision is not present, latest revision is used as
# default.
family: kw45b41z8
# ------------------------------------------------===== MCU revision [Optional] =====----------------------------------------
# Description: Revision of silicon
revision: latest
# ------------------------------------------===== EdgeLock 2GO API URL [Optional] =====-------------------------------------
# Description: Base URL of the EdgeLock 2GO backend API without an / at the end
url: https://api.edgelock2go.com
# ------------------------------------------===== EdgeLock 2GO API Key [Required] =====-------------------------------------
# Description: REST API key used for user authentication in EdgeLock 2GO. You may use: 1) path to a file with the key on
# the first line: ~/.el2go/wpc_token.txt (~ is interpreted as your HOME) 2) environment variable with the key:
# $MY_EL2GO_KEY 3) environment variable containing a path to the key file: $MY_KEY_FILE 4) directly your API key in
# plain text (not recommended)
api_key: abcxyz123
# ------------------------------------------------===== Device Group ID [Required] =====----------------------------------------
# Description: Device Group ID in which the connected secure element needs to be assigned
device_group_id: 49
# ------------------------------------------------===== Product 12NC [Required] =====----------------------------------------
# Description: Product 12NC registered in EdgeLock 2GO
nc12: 123456789012
# ------------------------------------------===== Secure Objects Domains [Optional] =====-------------------------------------
# Description: List of the EdgeLock 2GO Secure Objects Domains to download.
# Possible options: <RTP, MATTER>
domains:
  - RTP
  - MATTER
# ----------------------------===== Delay between EdgeLock 2GO's API calls [Optional] =====-------------------------------
# Description: Delay in seconds between EdgeLock 2GO API requests regarding Secure Objects generation status. Default
# value is 5 seconds
delay: 5
# ------------------------------------===== Timeout for EdgeLock 2GO API [Optional] =====-----------------------------------
# Description: Timeout in seconds of downloading Secure Objects through EdgeLock 2GO API operation. Default is 60
# seconds
timeout: 60
# ------------------------------------------===== Download Timeout [Optional] =====----------------------------------------
# Description: Timeout for overall download process including waiting for Secure Objects creation
download_timeout: 300
# -------------------------===== Address for External (user-defined) Secure Objects [Required] =====-----------------------
# Description: Address where to store External (user-defined) Secure Objects obtained from EL2GO. External Secure
# Objects refers to Secure Objects other than OEM FW Authentication Key Hash, and OEM FW Decryption Key. Please note
# that the FW will always erase 8K memory block (flash page size) in the destination address, so any data previously
# stored at that flash location will be gone. If you don't use any External Secure Objects, this setting has no effect
secure_objects_address: 123456
# ------------------------------------===== NXP EL2GO Provisioning FW path [Required] =====-------------------------------
# Description: Path to NXP EL2GO Provisioning Firmware SB file. If you don't want to use the Provisioning FW, please
# comment out the line below
prov_fw_path: oem_tp_fw.sb
```


aaa-060685

Figure 18. Configure SPSDK for EdgeLock 2GO provisioning

For the detailed information to input, see Figure 18:

1. Use the **el2go-host get-template** command to generate a configuration file for the processor used. Example: **el2go-host get-template -f kw45b41z8 -o kw45_el2go_config.yaml**
2. Use the values previously saved in the various steps to fill the relevant information in the file.
3. Fill in the API key value that you have saved. For details, see Section 6.2 "Create an API key".
4. Fill in the Device Group ID number that you have saved. For details, see Section 6.1 "Create a device group".
5. Fill in the 12NC number that you have saved. For details, see Section 6.1 "Create a device group".
6. Select a flash address to use for storing secure objects. This address is only used for external secure objects (secure objects that are not directly written to fuses or IFR).

7. Select the path to the trust provisioning firmware, located inside the restricted data package.
8. Save the file with new changes.

6.4 Set up provisioning credentials

The EdgeLock 2GO Managed remote trust provisioning service can securely configure, provision, and revoke secure objects in the IoT device. During the initial provisioning of the device, the EdgeLock 2GO Managed service requires the secure objects to be tied to the OEM firmware authentication key hash (`OEM_ROTKH`/`OEM_SRKH`), so it is necessary for the device to be registered and the `OEM_ROTKH`/`OEM_SRKH` recorded to the newly created device group.

6.4.1 Generate OEM firmware authentication key hash

ELE-based NXP MCU and MPU devices require four[1] root keys to be used for performing secure boot by the OEM. The public keys out of these OEM generated key pairs must be hashed to create the **OEM FW Authentication Key Hash**.

Root keys can be generated using the SPSDK tool.

For the SPSDK tool instructions, see Section 10 "Appendix B Steps to generate OEM FW authentication key hash (OEM_RTKH/OEM_SRKH)".

6.4.2 Install root key hash to device group (mandatory)

The following steps describe recording of the **OEM FW Authentication Key Hash** to the device group.

1 Some devices require up to four root keys. See the device-specific Security Reference Manual.

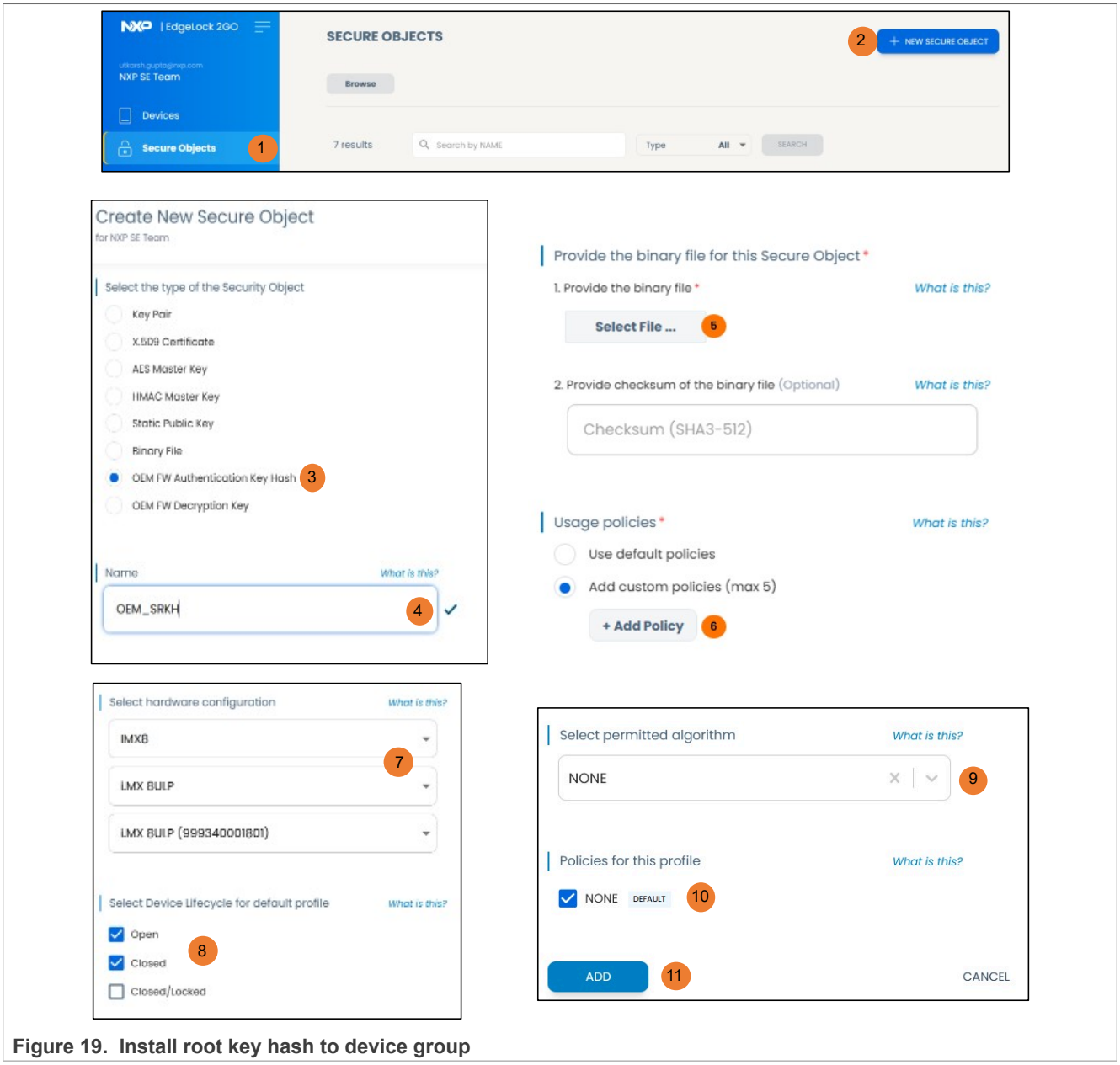

Figure 19. Install root key hash to device group

To install the root key hash to a device group, perform the following steps, as shown in Figure 19:

1. Click **Secure Objects**.
2. Click **New Secure Object**.
3. Select **OEM FW Authentication Key Hash**.
4. Enter the custom name in the **Name** field.
5. In the **Select File** field, choose the generated `OEM_ROTKH/OEM_SRKH` binary file from Step 4.
6. Select **Add custom policy** and click **+Add Policy**. This policy is to attach this secure object to a specific hardware.
7. Choose the appropriate hardware configurations.
8. Select the device lifecycle as **Open** or **Closed**, where this secure object can be imported.
9. Select **NONE** for the permitted algorithm.
10. Select **NONE** for the profile policy.

11. Click **Add** to create the secure object.

6.4.3 Generate OEM firmware decryption key

Most ELE based NXP MCU devices use SB3 files for secure firmware loading. For these devices, configuring an OEM firmware decryption key secure object is required.

For the SPSDK tool instructions, see Section 11 "Appendix C Steps to generate OEM FW decryption key(CUST_MK_SK/SB3KDK)".

6.4.4 Install SB3 key to device group (mandatory for devices that use SB3 files)

The following steps describe recording of the **OEM FW Decryption Key** to the device group.

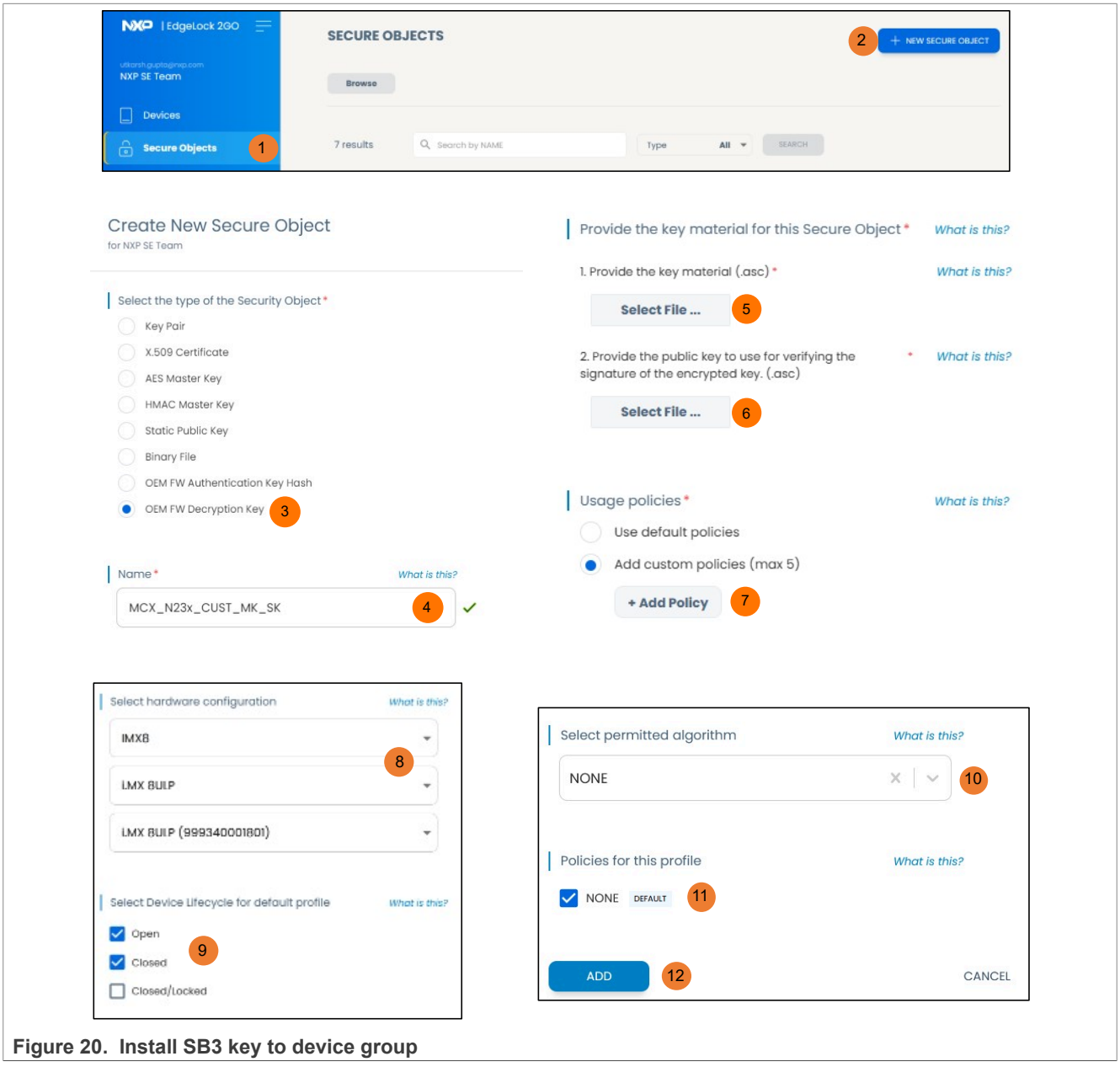

Figure 20. Install SB3 key to device group

To install the SB3 key to a device group, perform the following steps, as shown in Figure 20:

1. Click **Secure Objects**.
2. Select **New Secure Object**.
3. Select **OEM FW Decryption Key**.
4. Enter the custom name in the **Name** field
5. In the **Select File** field, choose the *cust_mk_sk.asc* file from Section 11 "Appendix C Steps to generate OEM FW decryption key(CUST_MK_SK/SB3KDK)".
6. In the **Select File** field, choose the *publick_signing_key.asc* file from Section 11 "Appendix C Steps to generate OEM FW decryption key(CUST_MK_SK/SB3KDK)".
7. Select **Add custom policy** and click **+Add Policy**. This policy is to attach this secure object to a specific hardware.
8. Choose the appropriate hardware configurations.
9. Select the device lifecycle as Open or Closed, where this secure object can be imported.
10. Select **NONE** for the permitted algorithm.
11. Select **NONE** for the profile policy.
12. Click **Add** to create the secure object.

6.4.5 Create and configure secure objects

Using EdgeLock 2GO service, it is possible to dynamically provision RSA and ECC key-pairs, AES keys, HMAC keys, X.509 certificates, binary files, and static public keys in the flash of the NXP MCU and MPU device. First, it is necessary to install the OEM FW Authentication Key Hash to the device group. The secure objects generated, thereafter, are encapsulated using the OEM FW Authentication Key Hash and thus can only be accessed in the devices with the same OEM FW Authentication Key Hash programmed in the eFuses.

For steps to set up different secure objects as needed, see the *Create a new secure object in EdgeLock 2GO – Managed* section in the *EdgeLock 2GO – Managed* (document AN12691).

6.4.6 Assign secure objects to the device group

Once the secure objects are created, they can be assigned to a device group. In this way, all devices belonging to the device group is provisioned with the secure object when they connect to the EdgeLock 2GO service. If a device is added to the device group later, it is provisioned with the secure object once it connects to the EdgeLock 2GO service.

For further details, see the **Assign a device group to a secure object** section in the *EdgeLock 2GO – Managed* (document AN12691).

6.5 Use SPSDK to provision a device

Now that the EdgeLock 2GO server is set with all the secure objects that you want to provision and the SPSDK tool has the information to access the device group, it is time to provision a device.

Note: *When you perform the EdgeLock 2GO provisioning, the lifecycle state of the device is written according to the lifecycle selected in the usage policy for the RKTH/SRKH and* `CUST_MK_SK` *secure objects. If your RKTH/SRKH and* `CUST_MK_SK` *secure objects are configured as* ***Closed****, then the lifecycle state of the device is advanced to* ***In Field****. The lifecycle state advancement burns fuses and is irreversible, so it is important that the desired lifecycle state was selected when creating the secure objects. Select only one lifecycle state option. If you're using a device that supports SB3 files, then the lifecycle state in the usage policy for the RKTH and* `CUST_MK_SK` *secure objects must also be the same.*

To use the SPSDK to provision a device, perform the following steps:

1. In your SPSDK command line, use the **el2go-host provision-device** command and connect a device in the ISP mode. Example: **el2go-host provision-device -p COM11 --config kw45_el2go_config.yaml**
2. The step above does the following steps in one command:
   a. Reads the UUID from the device
   b. Downloads secure objects from the EdgeLock 2GO server
   c. Writes the secure objects to a temporary location on the device
   d. Loads the OEM trust provisioning firmware into the device
   e. Executes the trust provisioning firmware which processes the secure objects and provisions the device

After a successful provisioning:

- The lifecycle fuse is written according to the lifecycle of the OEM FW authentication key hash (RKTH) and OEM FW decryption key (CUST_MK_SK) secure object lifecycles where Open = Develop, Closed = In Field, and Closed/Locked = In Field Locked.
- OEM FW authentication key hash value (RKTH) is written to CMPA for flash based devices. For external flash devices, the RKTH is written to shadow registers. If the life cycle is **In Field** or **In Field Locked**, then RKTH is written to fuses.
- OEM FW decryption key (`CUST_MK_SK`) is written to CMPA (flash based devices) or shadow registers (external flash devices). If the life cycle is **In Field** or **In Field Locked**, the `CUST_MK_SK` fuses are burned for external flash devices.
- External secure object blobs are written to the secure objects address where they can be accessed by an application.

7 Acronyms/Definitions

Table 1 lists the acronyms and definitions in this document.

Table 1. Acronyms and definitions

Acronym	Meaning
ELE	EdgeLock Enclave
SEC	MCUXpresso secure provisioning tool
SPSDK	Secure provisioning SDK
UUID	Universally Unique Identifier

8 References

Table 2. References

Document	Description
AN12642	EdgeLock 2GO Developer Guide
AN12691	EdgeLock 2GO - Managed
AN13255	EdgeLock 2GO Quick Start Guide
EdgeLock 2GO -Managed Service Description	EdgeLock 2GO - Managed Service Description
AN14544	EdgeLock 2GO Services for MPU and MCU
SPSDK	Secure Provisioning SDK
SEC	MCUXpresso Secure Provisioning Tool and restricted data

9 Appendix A UUID retrieval

UUID is the unique identifier fused in the NXP SoCs used to validate against the registered devices in the EdgeLock 2GO backend server and also to generate the secure objects unique to a device.

9.1 Read using SPSDK

Make sure that your board is configured for the ISP mode.

To read the UUID using SPSDK, perform the following steps, as shown in Figure 21:

1. In the SPSDK tool, use the **blhost get-property** command to read back the UUID.
   Example: **blhost -p COM11 get-property 18**

At the command line, paste the `blhost` command line.

```
C:\                                                    >blhost -u 0x1FC9,0x0158 -- get-property 18
Response status = 0 (0x0) Success.
Response word 1 = 3800551930 (0xe287d1fa)
Response word 2 = 1365066310 (0x515d4246)
Response word 3 = 3665152409 (0xda75c999)
Response word 4 = 1404528451 (0x53b76743)
Unique Device Identification = FA D1 87 E2 46 42 5D 51 99 C9 75 DA 43 67 B7 53
```

Figure 21. Paste the `blhost` command line

10 Appendix B Steps to generate OEM FW authentication key hash (OEM_RTKH/OEM_SRKH)

The SPSDK crypto backend provides unified API for all crypto operations needed for secure and trust provisioning like key and certificate generation, signing, encryption, hashing, and others.

```
(venv) C:\git_spsdk>nxpcrypto -v key generate -k secp256r1 -o workspace/ROT0_secp256r1.pem --force
The key pair has been created: C:\git_spsdk\workspace\ROT0_secp256r1.pub, C:\git_spsdk\workspace\ROT0_secp256r1.pem

(venv) C:\git_spsdk>nxpcrypto -v key generate -k secp256r1 -o workspace/ROT1_secp256r1.pem --force
The key pair has been created: C:\git_spsdk\workspace\ROT1_secp256r1.pub, C:\git_spsdk\workspace\ROT1_secp256r1.pem

(venv) C:\git_spsdk>nxpcrypto -v key generate -k secp256r1 -o workspace/ROT2_secp256r1.pem --force
The key pair has been created: C:\git_spsdk\workspace\ROT2_secp256r1.pub, C:\git_spsdk\workspace\ROT2_secp256r1.pem

(venv) C:\git_spsdk>nxpcrypto -v key generate -k secp256r1 -o workspace/ROT3_secp256r1.pem --force
The key pair has been created: C:\git_spsdk\workspace\ROT3_secp256r1.pub, C:\git_spsdk\workspace\ROT3_secp256r1.pem

(venv) C:\git_spsdk>nxpcrypto rot calculate-hash -f mcxn947 -k workspace\ROT0_secp256r1.pub -k workspace\ROT1_secp256r1.
pub -k workspace\ROT2_secp256r1.pub -k workspace\ROT3_secp256r1.pub
RoT hash: '963df49a1f6ae905b6a967015209cbf648ea2e5e7df618771405c4beef92df4d'
```

Figure 22. Keys and hash of keys

To generate the keys and calculate the hash, perform the following steps, as shown in Figure 22:

1. Use the **nxpcrypto key generate** command to generate four root-of-trust key pairs.
2. Use the **nxpcrypto rot calculate-hash** command to generate the hash of the four public keys for the device used.
3. Copy the hash and save it as the *rkth.bin* file.

The *rkth.bin* file is the binary file that you should upload to the EdgeLock 2GO server in the steps listed in Section 6.4.2 "Install root key hash to device group (mandatory)".

11 Appendix C Steps to generate OEM FW decryption key(CUST_MK_SK/ SB3KDK)

The *cust_mk_sk.asc* and *public_signing_key.asc* files in this folder are uploaded to the EdgeLock 2GO server as shown in the steps in Section 6.4.4 "Install SB3 key to device group (mandatory for devices that use SB3 files)".

The OEM FW decryption key is a 256-bit symmetric key that is user defined. There are many tools that can generate this value, such as a random number generator. Once the key is generated, it is necessary to encrypt it before loading it to the EdgeLock 2GO environment.

Follow the steps shown in the next figures to encrypt the OEM FW decryption key using Kleopatra.

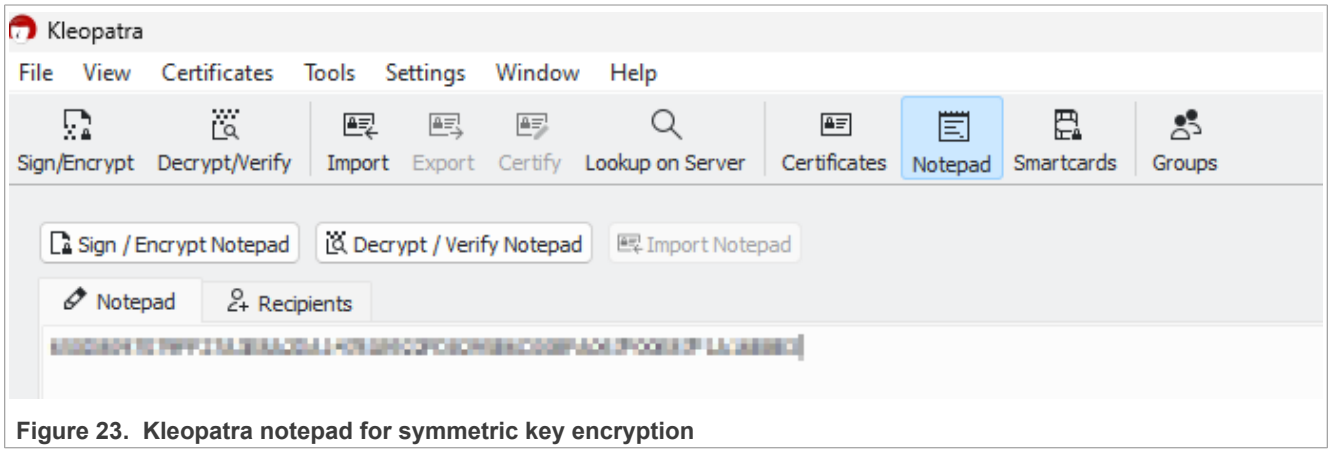

Figure 23. Kleopatra notepad for symmetric key encryption

1. Press the **Notepad** button in the Kleopatra tool.
2. Paste the value of the symmetric key directly to the notepad.

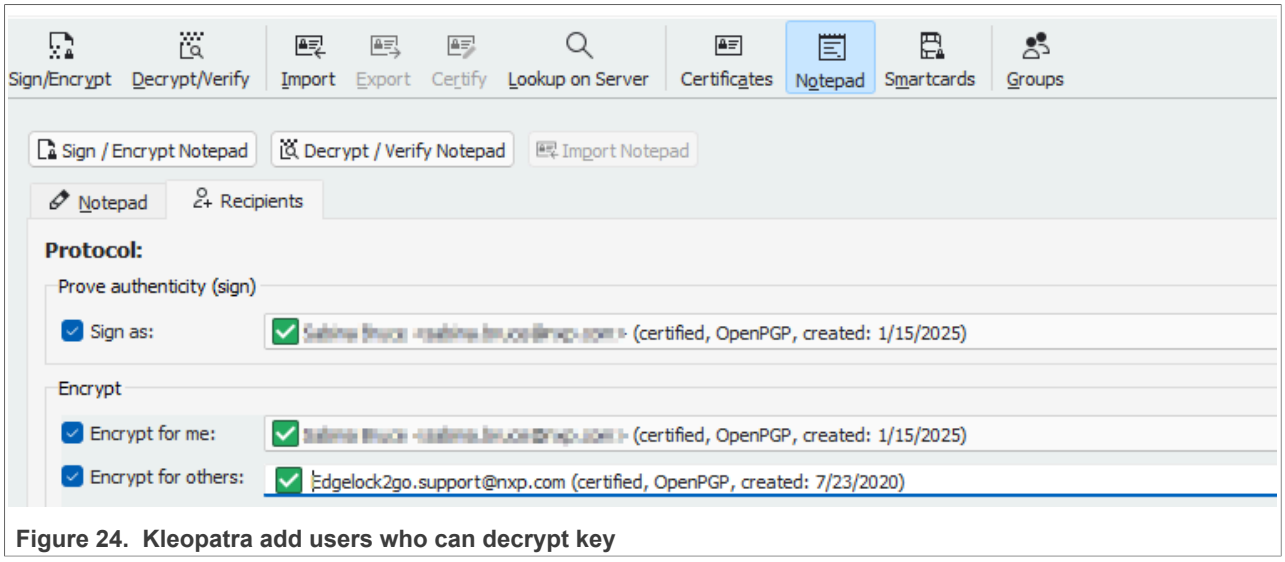

Figure 24. Kleopatra add users who can decrypt key

3. Click the **Recipients** tab.
4. Your credentials should already be listed in the **Sign** and **Encrypt** windows. In the **Encrypt** window, add a line for others and include the previously imported EdgeLock 2GO public key.

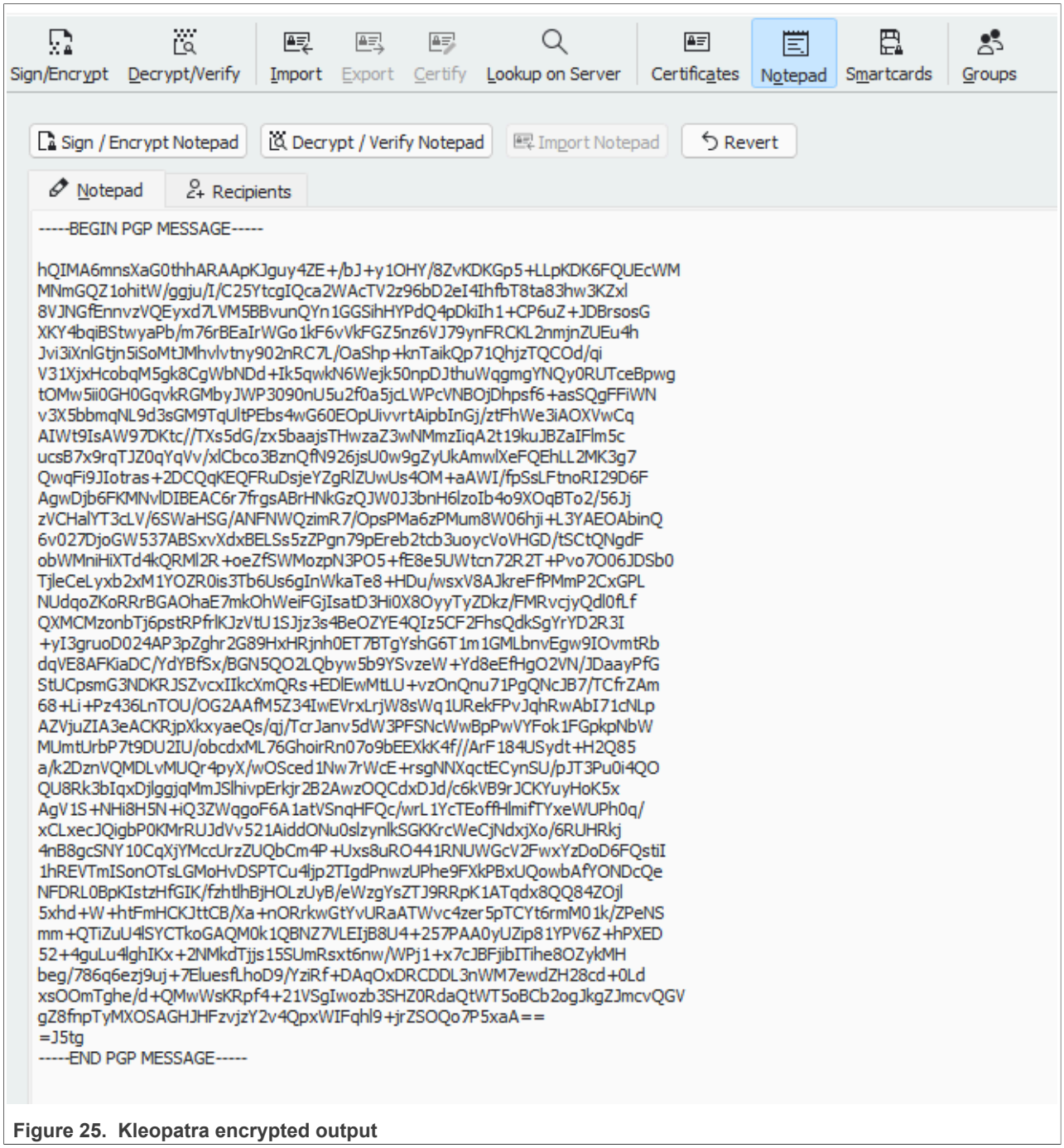

Figure 25. Kleopatra encrypted output

5. Click the **Sign/Encrypt Notepad** button to encrypt the symmetric key value.
6. Copy the contents to a file and save it as *cust_mk_sk..asc*.

12 Revision history

Table 3 summarizes the revisions to this document.

Table 3. Revision history

Document ID	Release date	Description
AN14670 v1.1	25 March 2026	Minor typo corrections
AN14670 v1.0	5 May 2025	Initial public release

Legal information

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.

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.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

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.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that 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 vulnerabilities or may support established security standards or specifications with known limitations. 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.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

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.

Amazon Web Services, AWS, the Powered by AWS logo, and FreeRTOS — are trademarks of Amazon.com, Inc. or its affiliates.

EdgeLock — is a trademark of NXP B.V.

Microsoft, Azure, and ThreadX — are trademarks of the Microsoft group of companies.

Contents