AN13478

Implementing FOTA on i.MX RT600

Rev. 0 — 06 June 2022

Application Note

Contents

1 Introduction

This application note introduces the implementation of FOTA on i.MX RT600. The Secure Bootloader (SBL) and Secure Firmware (SFW) are an Open Source Project (OTA) for MCU, launched by NXP. This project supports most chips of the i.MX RT series and LPC55S69.

2 SBL and SFW overview

The SBL and SFW are a secure firmware upgrade project for MCU, launched by NXP. The SBL is a second bootloader used with the FOTA-capable firmware. It manages the upgrade period by verifying and writing the new firmware image to the designated area of internal or external storage devices.

The SFW is based on FreeRTOS and is designed to implement a complete FOTA process together with the SBL. The SFW supports to obtain new firmware image through U-Disk and SD card locally, and remotely through the AWS cloud or Alibaba Cloud. After getting a new firmware image, the SFW itself writes the image to the storage device and set the corresponding flag, then reboot the device. After entering the SBL, it checks the new firmware image and completes the upgrade. See Figure 1 for the structural block diagram of SBL and SFW.

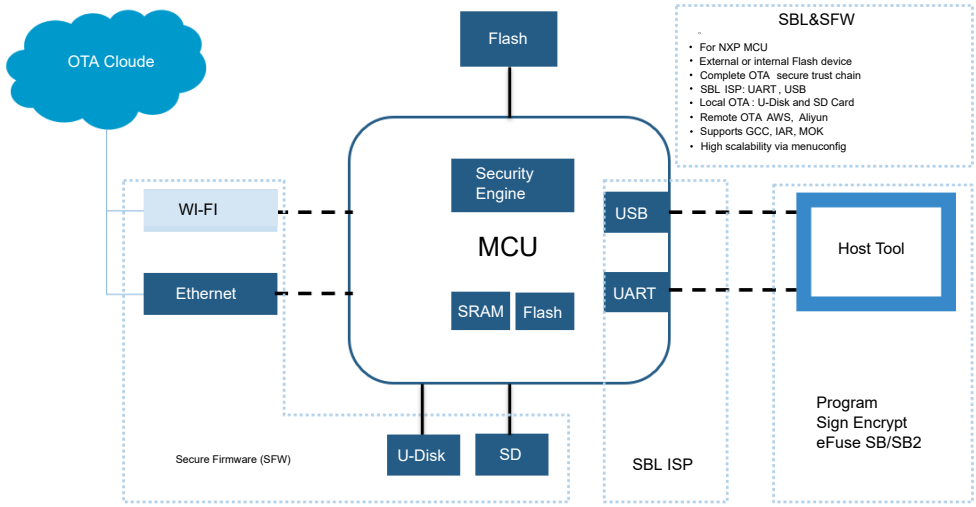

Figure 1. SBL and SFW diagram

3 FOTA implementation

This section demonstrates step-by-step procedure on how to use the SBL and SFW to perform OTA functions by SD Card or U-Disk with the example of i.MX RT600. Table 1 lists the NXP MCU boards supported by SBL and SFW. For details on SBL and SFW architecture, refer to *FOTA Design for SBL and SFW* (document AN13460).

Table 1. Supported NXP MCU boards

Board	Architecture	Boot device	Security		SBL			SFW OTA			
			Signature	Encryption	ISP	Swap	Remap	U-Disk	SD card	AWS	Aliyun
evkmimxrt1010	CM7	QSPI Flash	•	•	•		•	•			
evkmimxrt1020	CM7	QSPI Flash	•	•	•	•		•	•	•	•
evkbmimxrt1050	CM7	Hyper Flash	•	•	•	•		•	•	•	•
evkmimxrt1060	CM7	QSPI Flash	•	•	•		•	•	•	•	•
evkmimxrt1064	CM7	QSPI Flash	•	•	•		•	•	•	•	•
evkmimxrt1170	CM7+CM4	QSPI Flash	•	•	•		•	•	•	•	•
evkmimxrt500	CM33+F1	Octal Flash	•	•	•		•	•	•		
evkmimxrt600	CM33+HiFi4	Octal Flash	•	•	•		•	•	•		
lpc55s69	CM33+CM33	Internal Flash	•	•	•	•		•	•		

3.1 Signed + Non-encrypted OTA

FOTA includes signature and encryption functions, this chapter first introduces the combination of signature + no encryption.

1. Find the SFW path: `sfw\target\evkmimxrt600` in the SFW package.
2. Double-click `env.bat`.
3. Run the cmd `scons -menuconfig` to SFW configuration menu, see Figure 2.

Figure 2. SFW configuration menu

4. Select **MCU SFW core** > **Enable OTA** > **OTA from sdcard** > **OTA from u-disk**, see Figure 3.

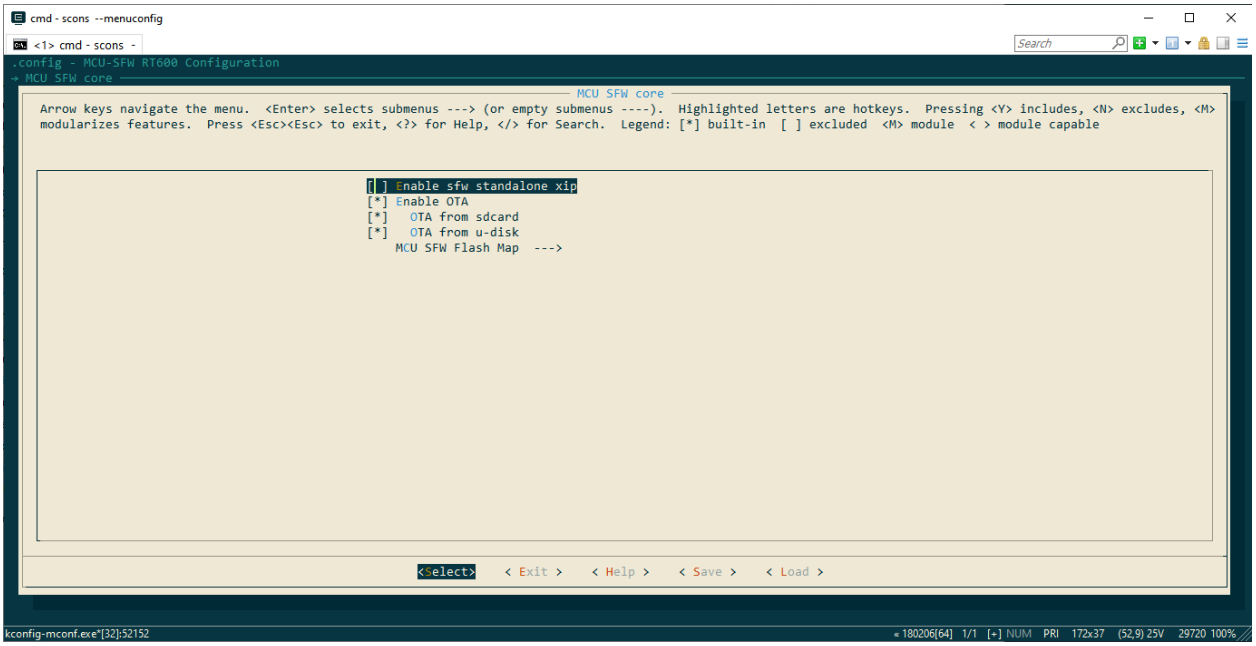

Figure 3. SFW OTA configuration

5. Select **MCU SFW Component** > **secure**.

NOTE

Do not select the **Encrypted XIP function**.

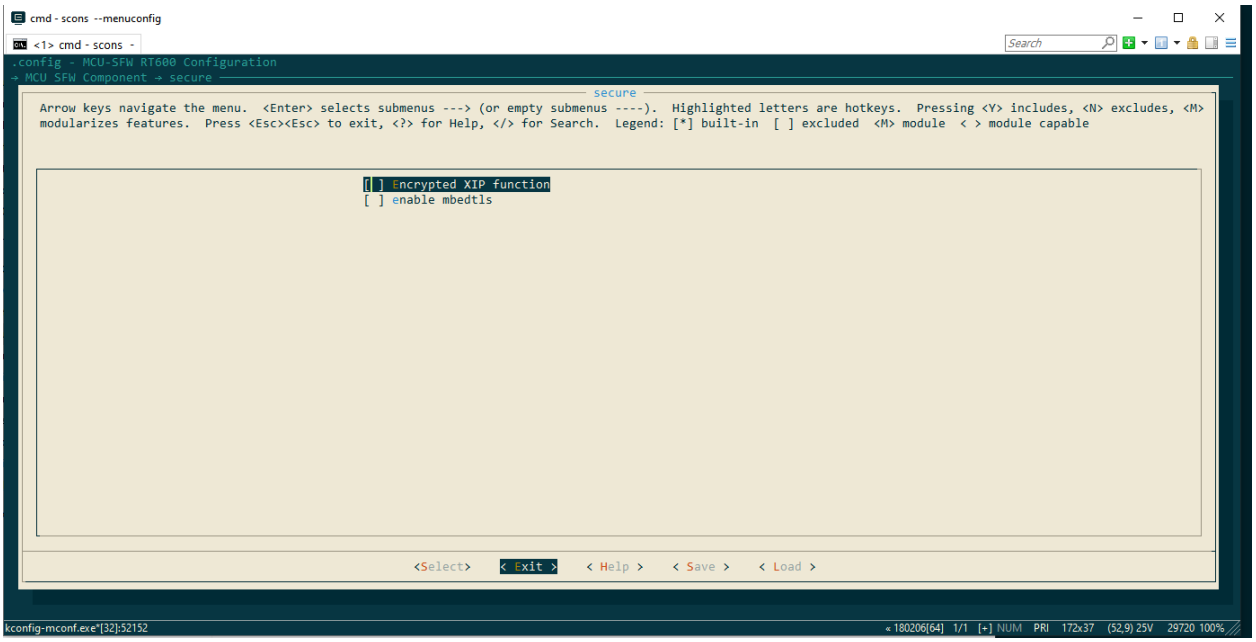

Figure 4. SFW component configuration

6. To save the configuration, select **Save** > **Modified and exit**, see Figure 4.

The SFW project supports three compilation tool-chains:

- IAR
- KEIL
- GCC

Here, use IAR to generate the `sfw.bin file`.

7. Run `scons --ide=iar` in the scons window and generate the SFW IAR project, see Figure 5.

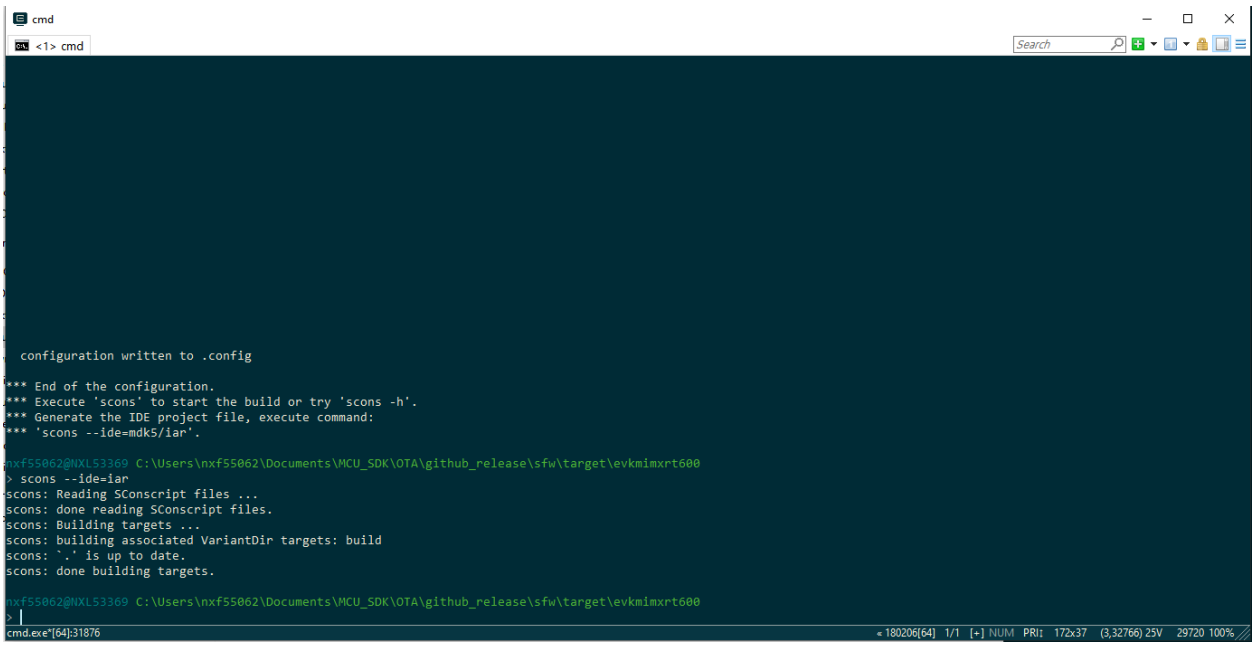

Figure 5. Generate IAR project

8. Find the path `sfw\target\evkmimxrt600\iar` in SFW IAR project.
9. Open the SFW IAR project,
   a. Change **hello sfw** to **hello sfw image1**, compile, and generate `sfw.bin`.
   b. Change **hello sfw** to **hello sfw image2**, generate sfw.bin.

   Rename the `sfw.bin` to `sfw2.bin`. Now, two `SFW.bin` files are ready, see Figure 6.

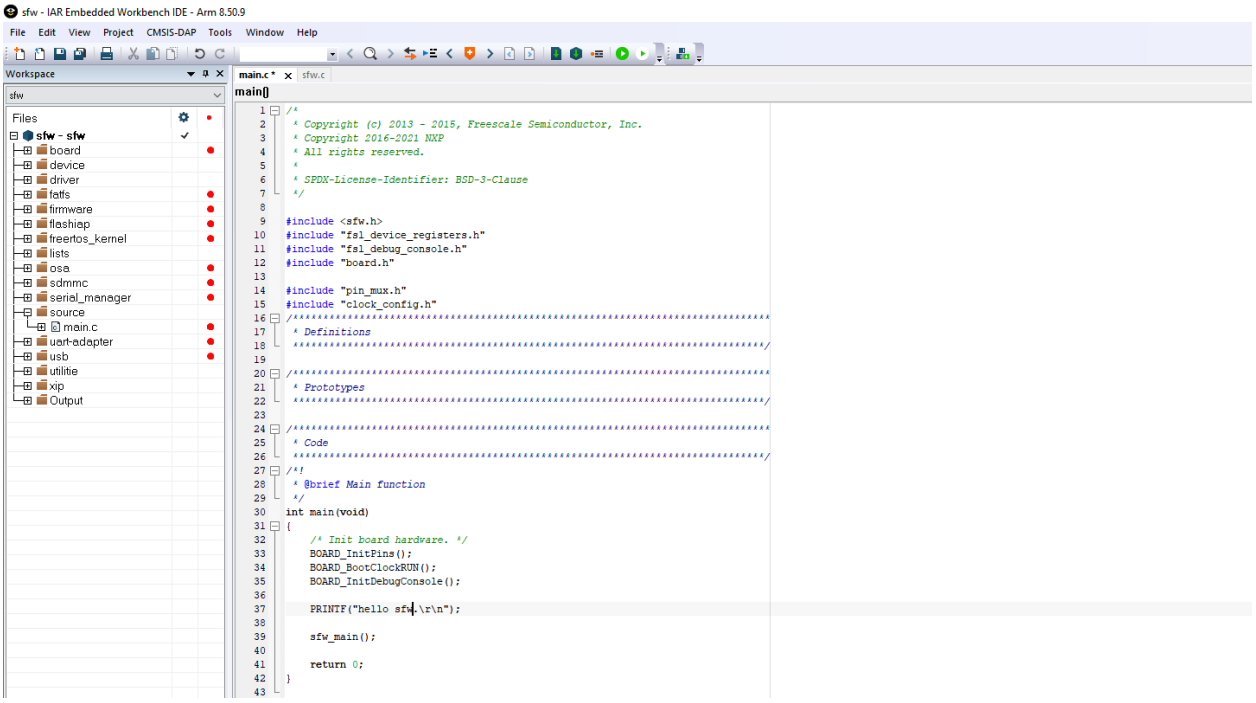

Figure 6. sfw.bin generate

10. Find the path of SBL: `sbl\target\evkmimxrt600` in SFW IAR project.
11. Double-click `env.bat` file, you get the window as shown in Figure 7.

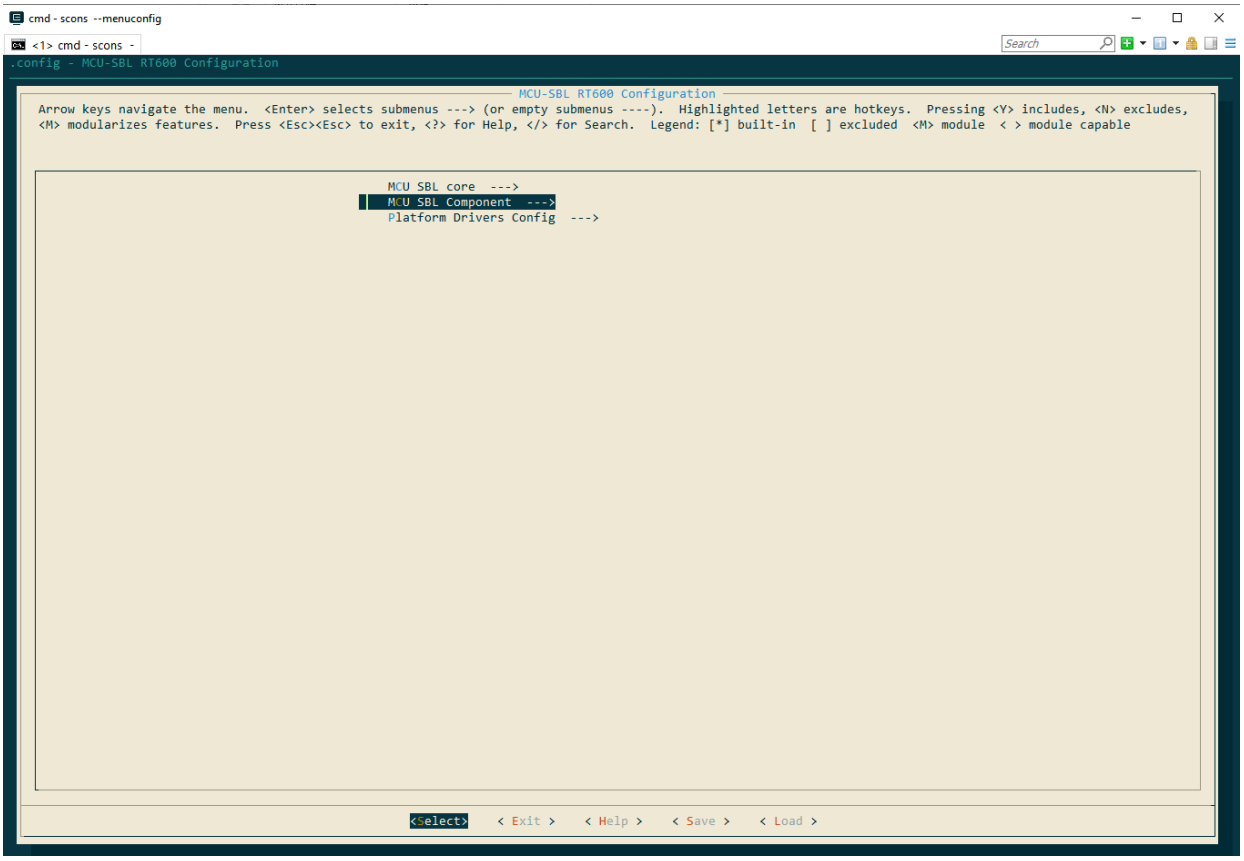

Figure 7. SBL component configuration

12. Select **MCU SBL Component > secure > signature function**.
13. Select signing method as **Select signature type ROM use**. Here, take **ROM use** as an example, two other signature methods are also supported, see Figure 8.

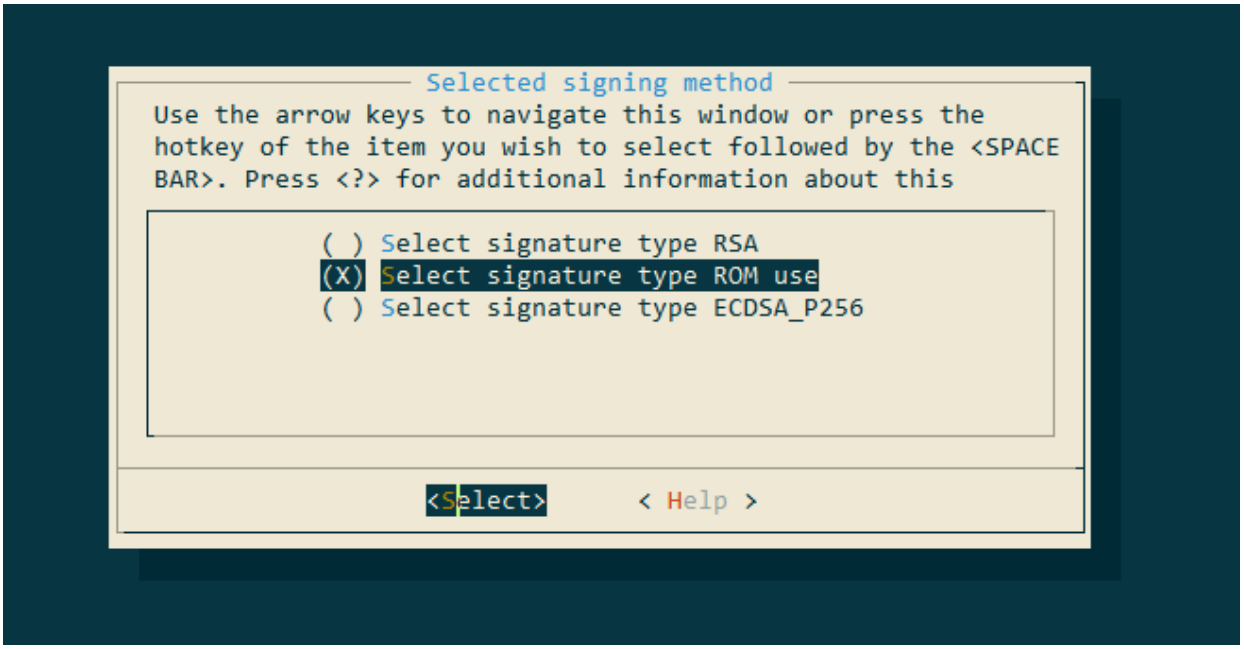

Figure 8. SBL secure configuration

14. Return to the previous window, because the signature + non-encryption is demonstrated first.

NOTE

Do not select **Encrypted XIP function**.

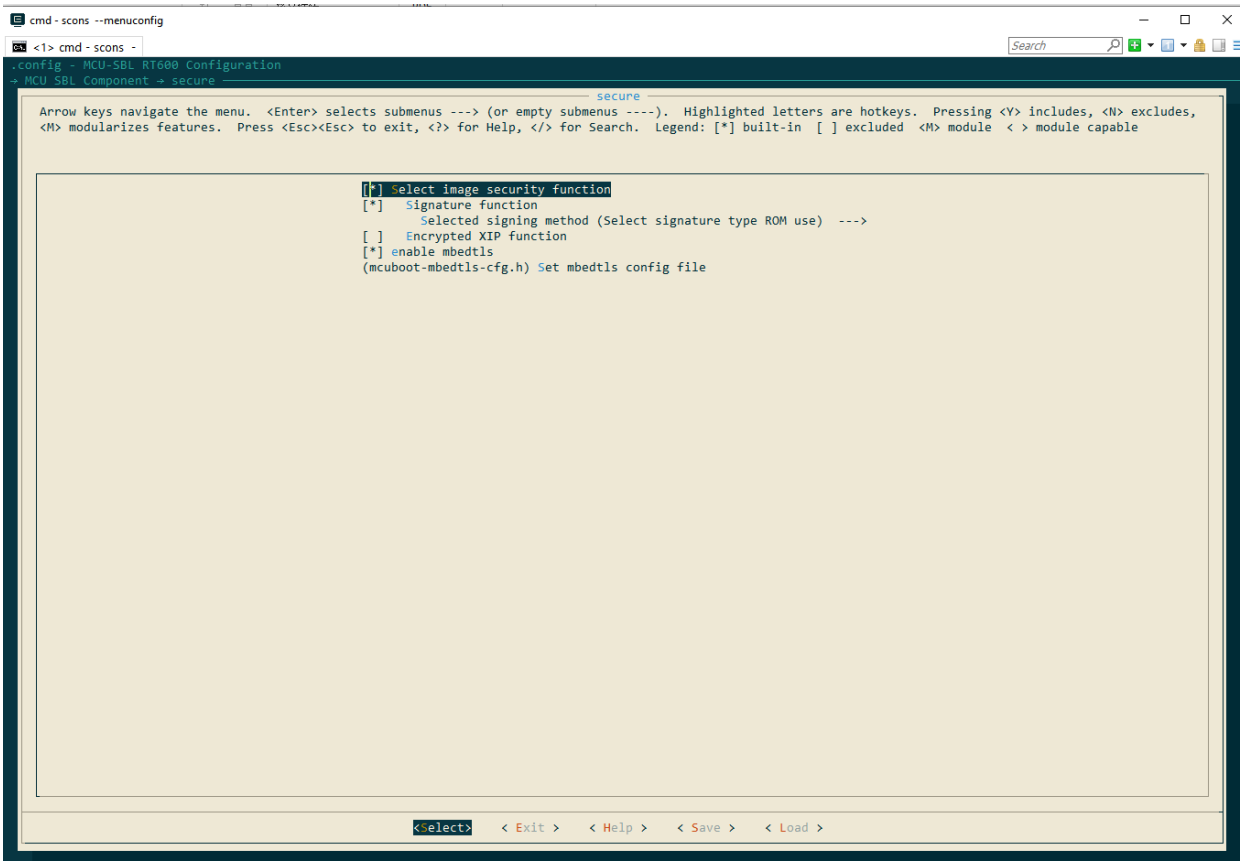

Figure 9. SBL signature configuration

15. To save the configuration, select **Save** > **Modified and exit**, see Figure 9.

    The SBL project supports three compilation tool-chains:

    - IAR
    - KEIL
    - GCC

    Here, use IAR to generate the `sbl.bin` file.

16. Run `scons --ide=iar` in the scons window and generate the SBL IAR project, compile SBL project, and generate `sbl.bin`.
17. Place the `sbl.bin`, `sfw.bin`, and `sfw2.bin` files at the path `sbl\target\evkmimxrt600\secure`.
18. To enable ROM secure boot:
    a. Generate keys and certificates, refer to chapter "7.4.4.1, Generating Keys and Certificates" of *MCU-OTA SBL and SFW User Guide* (document MCUOTASBLSFWUG).
    b. Copy folder **keys** and **crts** folder path `sbl/target/evkmimxrt600/secure`.
19. Use scripts to generate signed SBL and SFW, and download them to the RT600 EVK board. Since OTP can only be burned once, so use only shadow instead of burning OTP.
20. Put the attached scripts `otfad_enable.jlink` and `rkth_otpmaster.jlink` at the path `sbl\target\evkmimxrt600\secure`.

21. Open the `sign_sbl_app.bat`, add Jlink related scripts are shown in Figure 10.
22. Modify the JLink installation directory, serial number, and com port to be currently used.
23. Set the `signing_type` to `ROM_API`, see Figure 11.

```
@echo off
SET "PATH=C:\nxp\MCUX_Provi_v3.1\bin\tools\elftosb\win;%PATH%"
SET "PATH=C:\nxp\MCUX_Provi_v3.1\bin\tools\blhost\win;%PATH%"
SET "PATH=C:\nxp\MCUX_Provi_v3.1\bin\tools\cst\mingw32\bin;%PATH%"
SET imgtool_path=..\..\..\component\secure\mcuboot\scripts

SET Jlink="C:\Program Files (x86)\SEGGER\JLink\JLink.exe"
SET jlink_serial_number=600113866

SET com_port=COM25

@echo on

::*****************************************************************************
:: Configure signing method RSA2048, ECDSAP256 or ROM_API
::*****************************************************************************
set signing_type=ROM_API

set mcu_header_size=0x400
```

Figure 10. SBL Jlink script modification

```
::*****************************************************************************
:: Configure efuse to enable secure boot
::*****************************************************************************
%Jlink% -SelectEmuBySN %jlink_serial_number% -Device MIMXRT685S_M33 -IF SWD -Speed auto -ExitOnError -CommanderScript rkth_otpmaster.jlink

pause
```

Figure 11. SBL Jlink script configuration

24. Open the `sign_enc_sfw.bat`.
25. Set the `signing_type` to `ROM_API`.
26. Modified the `sfw2_otfad_arg`, see Figure 12.

```
@echo off
SET "PATH=C:\nxp\MCUX_Provi_v3.1\bin\tools\elftosb\win;%PATH%"
SET "PATH=C:\nxp\MCUX_Provi_v3.1\bin\tools\blhost\win;%PATH%"
SET "PATH=C:\nxp\MCUX_Provi_v3.1\bin\tools\cst\mingw32\bin;%PATH%"
SET "PATH=C:\nxp\MCUX_Provi_v3.1\bin\tools\image_enc\win;%PATH%"
SET imgtool_path=..\..\..\component\secure\mcuboot\scripts

SET user_kek=kek=0102030405060708090a0b0c0d0e0f00
SET sfw2_otfad_arg=otfad_arg=[00112233445566778899aabbccddeeff,0020406001030507,0x08101000,0x0000]

@echo on

::*****************************************************************************
:: Configure signing method RSA2048, ECDSAP256 or ROM_API
::*****************************************************************************
set signing_type=ROM_API

set mcu_header_size=0x400
```

Figure 12. SFW Jlink script configuration

27. Run `sign_enc_sfw.bat` in scons window and generate the `sfw_2_enc.bin` file, see Figure 13.

```
nxf55062@NXL53369 C:\Users\nxf55062\Documents\MCU_SDK\OTA\github_release\sbl\target\evkmimxrt600
> cd secure

nxf55062@NXL53369 C:\Users\nxf55062\Documents\MCU_SDK\OTA\github_release\sbl\target\evkmimxrt600\secure
> sign_enc_sfw2.bat

set signing_type=ROM_API

set mcu_header_size=0x400

if not exist ".\sfw2.bin" (
echo Can't find file sfw2.bin
 pause
 exit
)

if ROM_API == ROM_API (
elftosb -V -f rt6xx -J .\signed_sfw2_xip.json
 python img_helper.py paddingimage --pad-size 0x400 --input .\sfw_2_signed.bin --output .\sfw_2_padding.bin
 image_enc.exe hw_eng=otfad ifile=.\sfw_2_padding.bin ofile=.\sfw_2_enc.bin base_addr=0x08100000 kek=0102030405060708090a0b0c0d0e0f00 otfad_arg=[00112233445566
 python img_helper.py extract-keycontext --type otfad --enc_image .\sfw_2_enc.bin --output .\sfw_2_keyblob.bin
 python ..\..\..\component\secure\mcuboot\scripts\imgtool.py create --align 4  --version "1.1"  --header-size 0x400 --pad-header --slot-size 0x100000 --key-info
 python img_helper.py merge --header-size 0x400 --sign-image .\sfw_2_bootheader.bin --enc-image .\sfw_2_enc.bin
)  else (
python img_helper.py paddingimage --pad-size 0x400 --input .\sfw2.bin --output .\sfw_2.bin
 image_enc hw_eng=otfad ifile=.\sfw_2.bin ofile=.\sfw_2_enc.bin base_addr=0x08100000 kek=0102030405060708090a0b0c0d0e0f00 otfad_arg=[00112233445566778899aabbcc
 python img_helper.py extract-keycontext --type otfad --enc_image .\sfw_2_enc.bin --output .\sfw_2_keyblob.bin
 if ROM_API == RSA2048 (python ..\..\..\component\secure\mcuboot\scripts\imgtool.py sign --key ..\..\..\component\secure\mcuboot\scripts\sign-rsa2048-priv.pem
-max-sectors 32 --key-info .\sfw_2_keyblob.bin  .\sfw2.bin .\sfw_2_sign.bin )  else (python ..\..\..\component\secure\mcuboot\scripts\imgtool.py sign --key ..\
n "1.1" --header-size 0x400 --pad-header --slot-size 0x100000 --max-sectors 32 --key-info .\sfw_2_keyblob.bin  .\sfw2.bin .\sfw_2_sign.bin )
 python img_helper.py merge --header-size 0x400 --sign-image .\sfw_2_sign.bin --enc-image .\sfw_2_enc.bin
)
Parsing configuration file: .\signed_sfw2_xip.json.
        No "multicoreImages" section present in configuration file: .\signed_sfw2_xip.json.
        Used "imageLinkAddress" value: 135270400.
        Used "imageBuildNumber" value: 1.
        Success.
Starting processing image....
1. Check of the image file.
        Success. (File ./sfw2.bin: Size = 143689 bytes, AlignedSize = 143692 bytes)
2. Checking multicore configuration.
        Image is not containing multicore data.
        Success.
3. Fetching of image configuration: execution target and security.
        Internal flash (XIP) - plain signed: image will be signed based on provided configuration.
        Success.
3.1 Checking image link address configuration.
        Image link address will be set to: 0x08101000
        Success.
3.2 Checking image trust zone configuration.
        Trust zone enabled image: configuration of TZM-M_Preset disabled -> TZM-M_PresetFile is ignored and not used.
3.3 Checking image HW user mode keys enablement for all security levels.
        HW user mode key disabled.
        Success.
Start to generate signed image!
4. Load the root certificates.
4.1 Load the count of root certificates.
```

Figure 13. Run sign_enc_sfw.bat

28. Prepare the RT600 EVK board:

    a. Connect the Jlink to the board at JTAG interface.

b. Make sure that the jump JP2 is open.

c. Connect the USB cable to the board at JP5.

d. Set SW5 to ON, OFF, OFF to make RT600 ISP enter into **serial ISP** mode.

e. Run `sign_sbl_app.bat` in the scons window and generate the signed `sbl` and `signed sfw` file, see Figure 14.

f. Download signed `sbl` and `signed sfw` file on the RT600 EVK board.

```
cmd - sign_sbl_app.bat
<1> cmd - sign_sbl
nxf55062@NXL53369 C:\Users\nxf55062\Documents\MCU_SDK\OTA\github_release\sbl\target\evkmimxrt600\secure
> sign_sbl_app.bat

set signing_type=ROM_API

set mcu_header_size=0x400

if not exist ".\sbl.bin" (
echo Can't find file sbl.bin
 pause
 exit
)

elftosb -V -f rt6xx -J .\signed_sbl_xip.json
Parsing configuration file: .\signed_sbl_xip.json.
        No "multicoreImages" section present in configuration file: .\signed_sbl_xip.json.
        Used "imageLinkAddress" value: 134221824.
        Used "imageBuildNumber" value: 1.
        Success.
Starting processing image....
1. Check of the image file.
        Success. (File ./sbl.bin: Size = 56363 bytes, AlignedSize = 56364 bytes)
2. Checking multicore configuration.
        Image is not containing multicore data.
        Success.
3. Fetching of image configuration: execution target and security.
        Internal flash (XIP) - plain signed: image will be signed based on provided configuration.
        Success.
3.1 Checking image link address configuration.
        Image link address will be set to: 0x08001000
        Success.
3.2 Checking image trust zone configuration.
        Trust zone enabled image: configuration of TZM-M_Preset disabled -> TZM-M_PresetFile is ignored and not used.
3.3 Checking image HW user mode keys enablement for all security levels.
        HW user mode key disabled.
        Success.
Start to generate signed image!
4. Load the root certificates.
4.1 Load the count of root certificates.
        Success. (Root Certificate Count = 4)
4.2 Load selected certificate chain id, used to sign this image.
        Success. (Selected certificatate chain index = 0)
4.3 Load all root certificates.
        Root certificate 0 is self signed.
        Success. (Root Certificate 0 = ./crts/ROT1_sha256_2048_65537_v3_ca_crt.der)
        Root certificate 1 is self signed.
        Success. (Root Certificate 1 = ./crts/ROT2_sha256_2048_65537_v3_ca_crt.der)
        Root certificate 2 is self signed.
        Success. (Root Certificate 2 = ./crts/ROT3_sha256_2048_65537_v3_ca_crt.der)
        Root certificate 3 is self signed.
        Success. (Root Certificate 3 = ./crts/ROT4_sha256_2048_65537_v3_ca_crt.der)
4.4 Calculate size of root certificates.
        Success. (Root Certificate Size = 819 bytes, Aligned Size = 820 bytes)
5. Load all certificates in selected certificate chain.
5.1 Load the count of chained certificates in selected certificate chain.
        Success. (Certificate count = 2)
5.2 Load and parse certificates in selected certificate chain.
```

Figure 14. Run sign_sbl_app.bat

29. Set SW5 to ON, OFF, ON to make RT600 ISP enter into **boot from flexspi port b** mode.

30. Press SW3, reset RT600.

31. Open the serial port terminal on the PC, and you see the log `hello sfw image1` as shown in Figure 15, which represents SBL and SFW has run successfully.

```
hello sbl.
Bootloader Version 1.1.0
Remap type: none

The image now in PRIMARY_SLOT slot

Bootloader chainload address offset: 0x100000
Reset_Handler address offset: 0x100400
Jumping to the image


hello sfw image1.
Current image verison: 1.0.0
U-Disk updating task enable.
SD Card updating task enable.
Hello world1.
Hello world2.
Please plug in a u-disk to board.
Hello world1.
Hello world2.
Hello world1.
Hello world2.
Hello world1.
Hello world2.
```

Figure 15. SBL and SFW successfully run logs

32. Put the `sfw_2_enc.bin` generated by `sign_enc_sfw.bat` into the SD card or U-Disk, and rename it to `newapp.bin`. Here, take SD as an example for demonstration insert the SD into the SD card slot of the RT600 EVK board.

```
Card inserted.
Hello world1.
Hello world2.
Hello world1.
Hello world2.
reading...
new img verison: 1.1.0
updating...
Hello world1.
finished
write update type = 0x1
write magic number offset = 0xffff0
Please remove the SD Card!
sys rst...
```

Figure 16. SBL and SFW successfully update logs

33. Remove the SD card when you see the log **Please remove the SD card** as shown in Figure 16.
34. Reset the RT600 and you see the log `hello sfw image2` as shown in Figure 17, which represents SFW has been updated successfully.

```
hello sbl.
Bootloader Version 1.1.0
Remap type: none

The image now in SECONDARY_SLOT slot

Bootloader chainload address offset: 0x100000
Reset_Handler address offset: 0x100400
Jumping to the image


hello sfw image2.
Current image verison: 1.1.0
U-Disk updating task enable.
SD Card updating task enable.
Hello world1.
Hello world2.
Please plug in a u-disk to board.
Hello world1.
Hello world2.
```

Figure 17. SBL and SFW successfully update and boot logs

So far, FOTA has demonstrated the signature + non-encryption function on the RT600 EVK board.

3.2 Signed + Encrypted OTA

FOTA includes signature and encryption functions, this chapter introduces the combination of signature + encryption.

1. Find the SFW path: `sfw\target\evkmimxrt600` in the SFW package.
2. Double-click `env.bat`.
3. Run the cmd `scons -menuconfig` to SFW configuration menu, see Figure 18.

Figure 18. SFW configuration menu

4. Select the **MCU SFW core** > **Enable OTA** > **OTA from sdcard** > **OTA from u-disk**, see Figure 19.

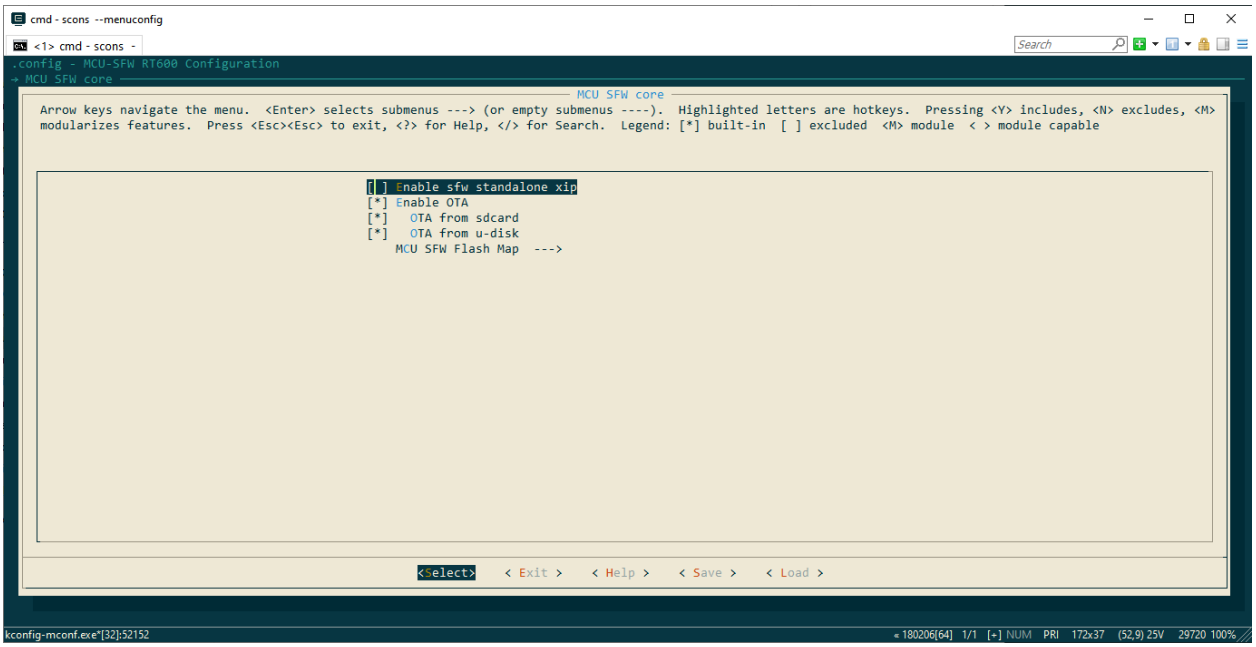

Figure 19. SFW OTA configuration

5. Select **MCU SFW Component > secure > Encrypted XIP function**.

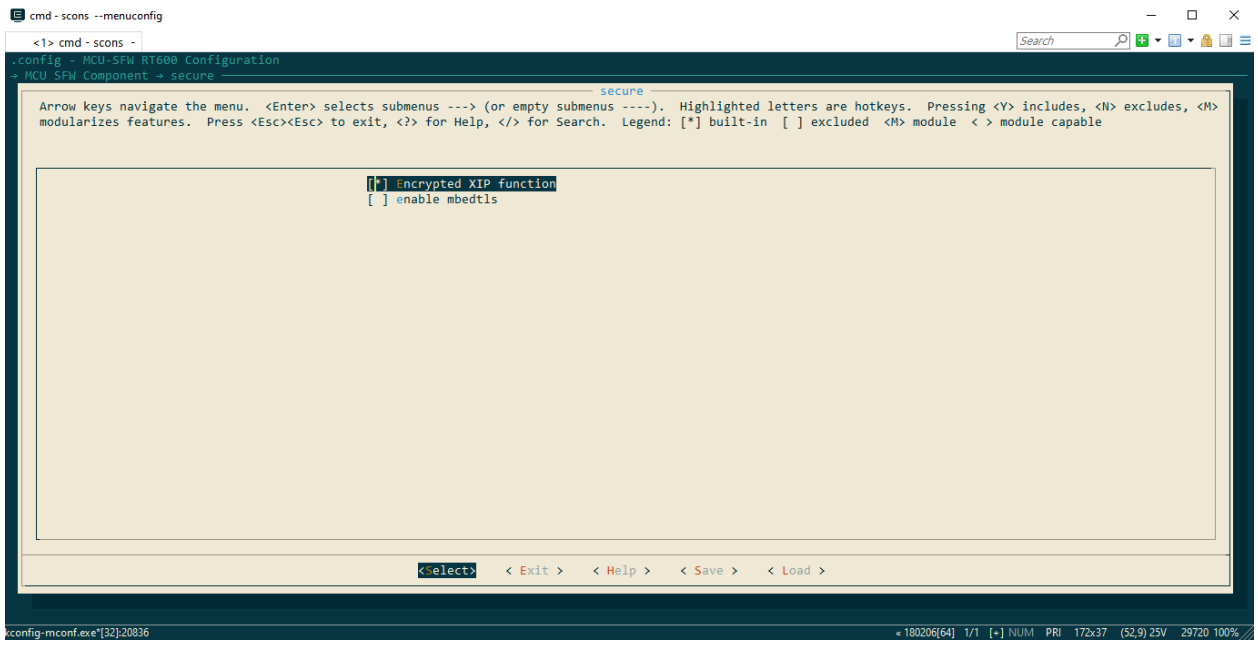

Figure 20. SFW component configuration

6. To save the configuration, select **Save>Modified and exit**, see Figure 20.

   The SFW project supports three compilation tool-chains:

   - IAR
   - KEIL
   - GCC

   Here, use IAR to generate the sfw.bin file.

7. Run `scons --ide=iar` in the scons window and generate the SFW IAR project, see Figure 21.

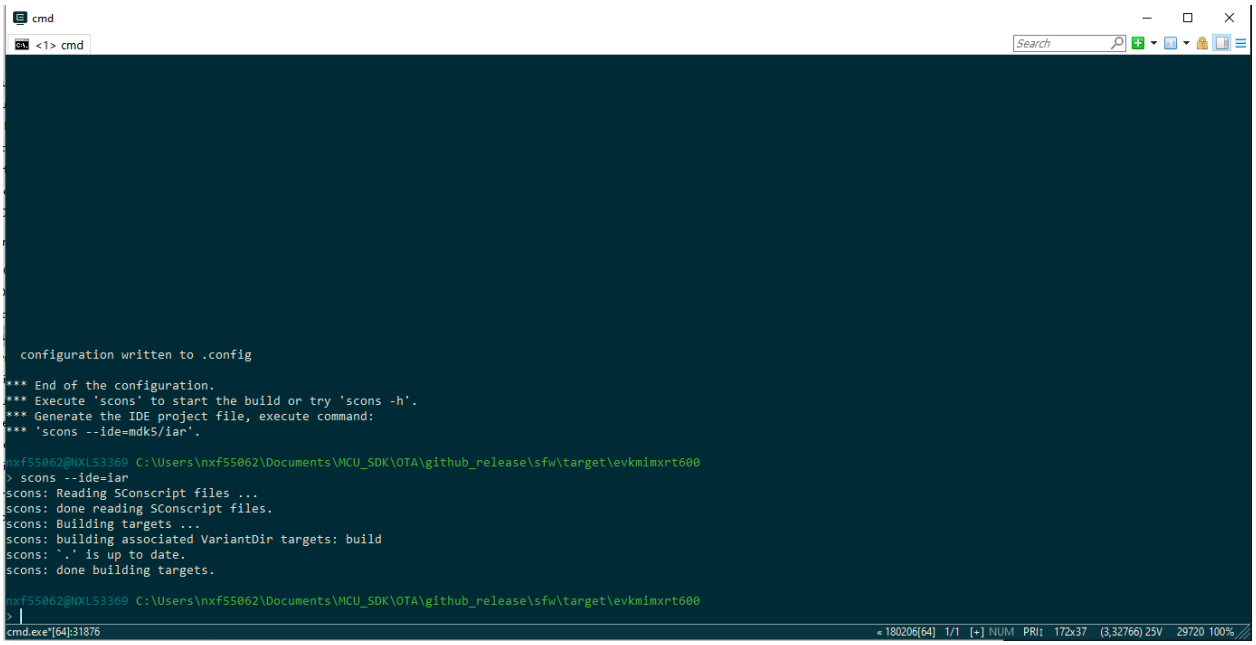

Figure 21. Generate IAR project

8. Find the path `sfw\target\evkmimxrt600\iar`, open the SFW IAR project:
   a. Change **hello sfw** to **hello sfw image1**, compile, and generate `sfw.bin`.
   b. Change **hello sfw** to **hello sfw image2**, generate `sfw.bin`.

   Rename the `sfw.bin` to `sfw2.bin`. Now, two `SFW.bin` files are ready, see Figure 22.

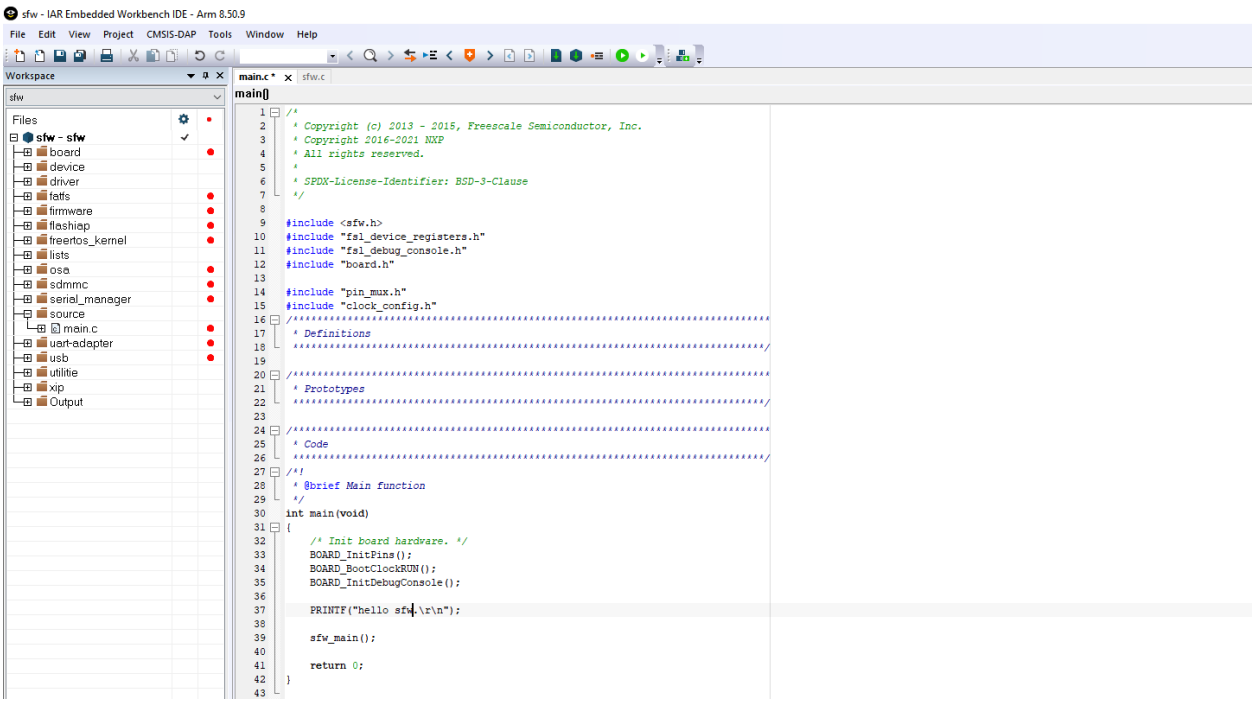

Figure 22. sfw.bin generate

9. Find the path of SBL: `sbl\target\evkmimxrt600`.

10. Double-click `env.bat` file, you get the window as shown in Figure 23.

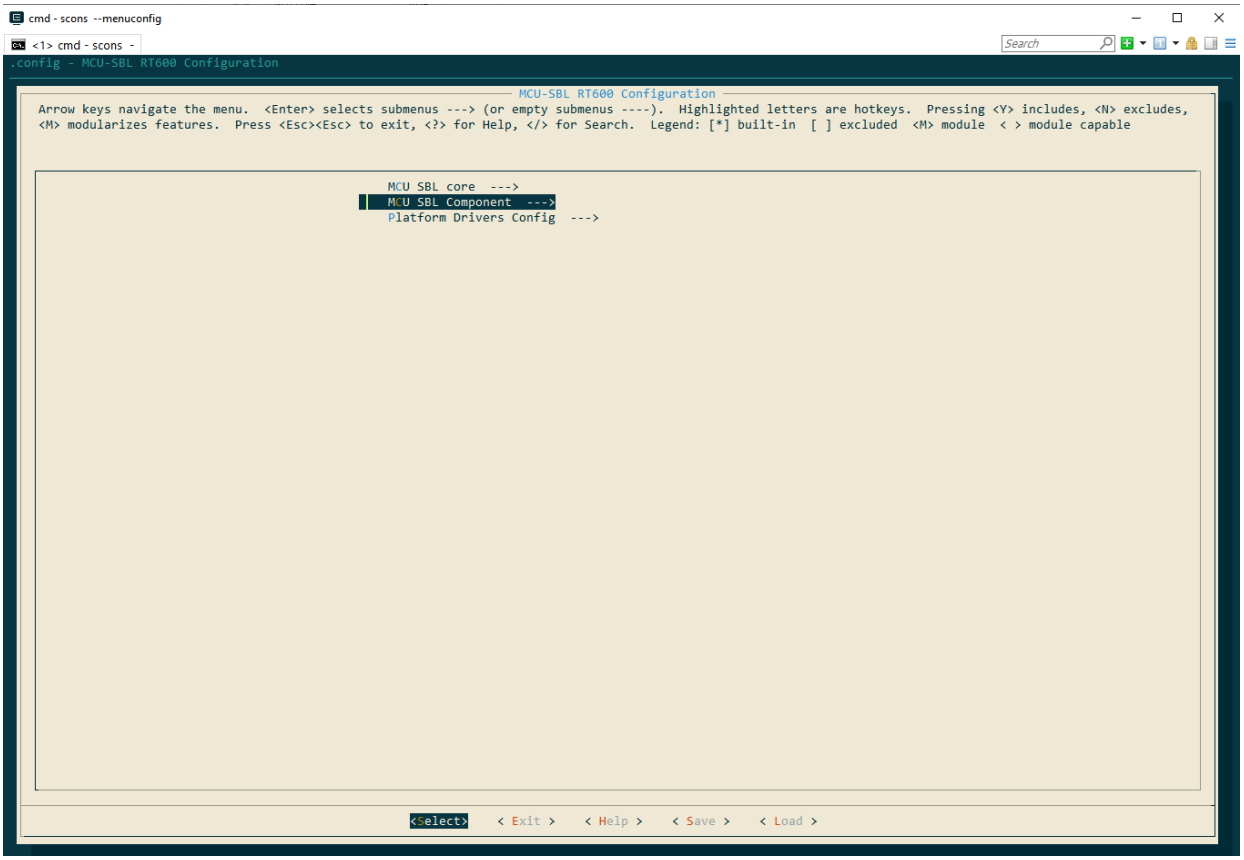

Figure 23. SBL component configuration

11. Select the **MCU SBL Component** > **secure** > **signature function** > **signing method** > **Select signature type RSA**. Here, take **RSA** as an example, two other signature methods are also supported, see Figure 24.

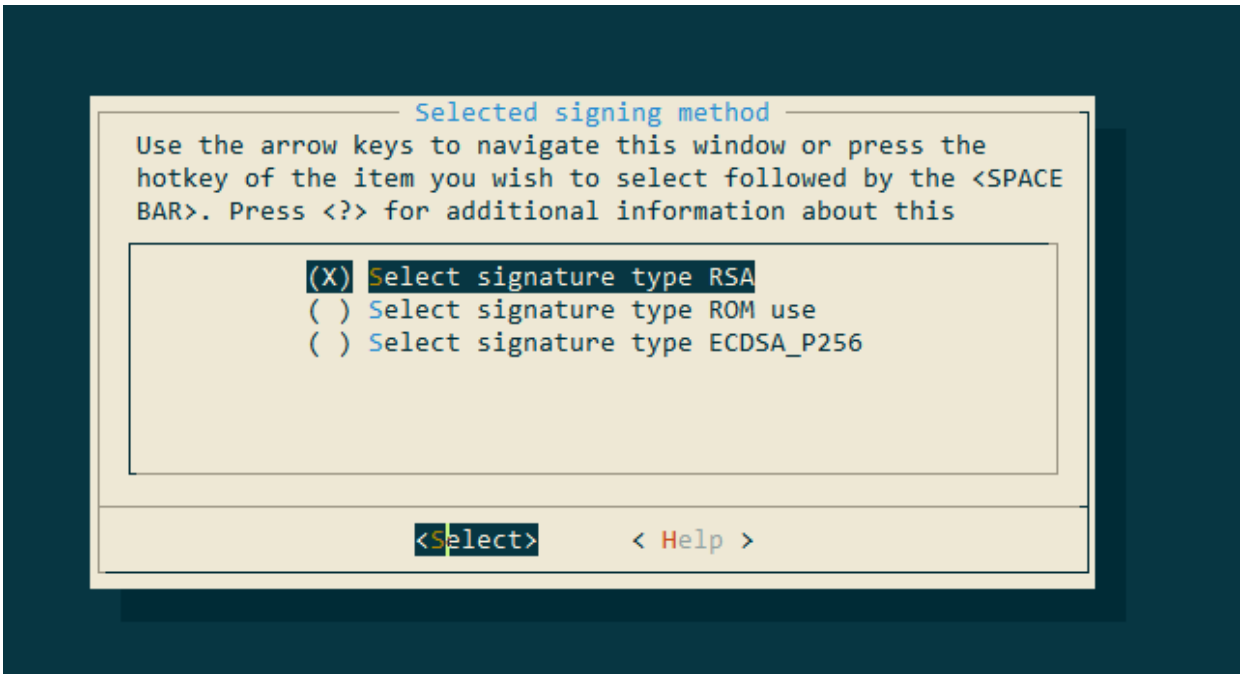

Figure 24. SBL secure configuration

12. Return to the previous window, because the signature + encryption is demonstrated.
13. Select **Encrypted XIP function**.

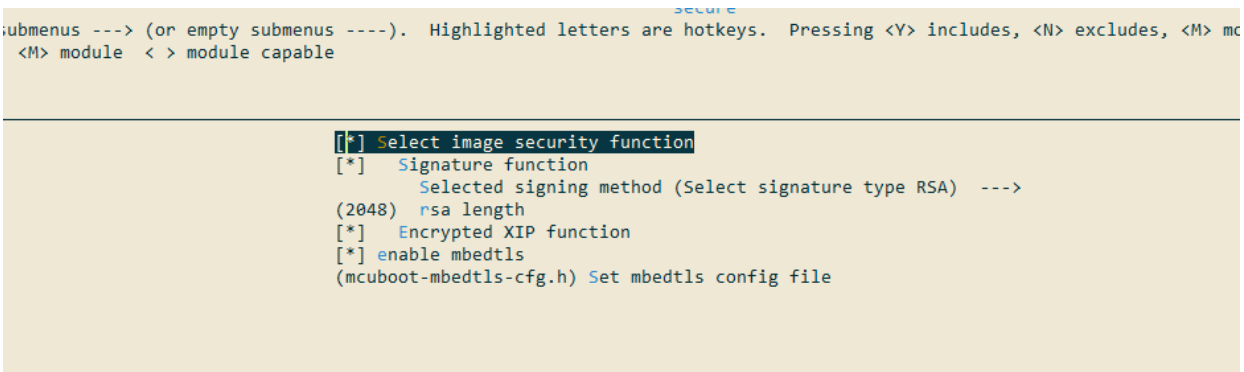

Figure 25. SBL signature configuration

14. To save the configuration, select **Save** > **Modified and exit**, see Figure 25.

    The SBL project supports three compilation tool chains:

    - IAR
    - KEIL
    - GCC

    Here, use IAR to generate the sbl.bin file.

15. Run `scons --ide=iar` in the scons window and generate the SBL IAR project.
16. Compile SBL project and generate `sbl.bin`.

17. Place the `sbl.bin`, `sfw.bin`, and `sfw2.bin` file at the path `sbl\target\evkmimxrt600\secure`.
18. To enable ROM secure boot:
    a. Generate keys and certificates, refer to chapter "7.4.4.1, Generating Keys and Certificates" of *MCU-OTA SBL and SFW User Guide* (document MCUOTASBLSFWUG).
    b. Copy folder **keys** and **crts** to folder path `sbl/target/evkmimxrt600/secure`.

NOTE

This step only must be done only once.

19. Use scripts to generate signed SBL and SFW, and download them to the RT600 EVK board. Since OTP can only be burned once, so only use shadow instead of burning OTP.
20. Place the attached scripts `otfad_enable.jlink` and `rkth_otpmaster.jlink` at the path `sbl\target\evkmimxrt600\secure`.
21. Open the `sign_enc_sbl_app.bat`.
22. Add Jlink related scripts as shown in Figure 26.
23. Modify the JLink installation directory, serial number, and com port to be currently used.
24. Set the `signing_type` to RSA2048, see Figure 27.

```
@echo off
SET "PATH=C:\nxp\MCUX_Provi_v3.1\bin\tools\elftosb\win;%PATH%"
SET "PATH=C:\nxp\MCUX_Provi_v3.1\bin\tools\blhost\win;%PATH%"
SET "PATH=C:\nxp\MCUX_Provi_v3.1\bin\tools\cst\mingw32\bin;%PATH%"
SET "PATH=C:\nxp\MCUX_Provi_v3.1\bin\tools\image_enc\win;%PATH%"
SET imgtool_path=..\..\..\component\secure\mcuboot\scripts

SET Jlink="C:\Program Files (x86)\SEGGER\JLink\JLink.exe"
::SET jlink_serial_number=724513722
SET jlink_serial_number=600113866

SET com_port=COM25

SET user_kek=kek=0102030405060708090a0b0c0d0e0f00
SET initial_otfad_arg=otfad_arg=[00112233445566778899aabbccddeeff,0020406001030507,0x08001000,0x1000],[00112233445566778899aabbccddeeff,0020406001030507,0x08101000,0x5000]
SET sfw1_otfad_arg=otfad_arg=[00112233445566778899aabbccddeeff,0020406001030507,0x08101000,0x5000]

@echo on

::*************************************************************************
:: Configure signing method RSA2048, ECDSAP256 or ROM_API
::*************************************************************************
set signing_type=RSA2048

set mcu_header_size=0x400
```

Figure 26. SBL Jlink script modification

```
::*************************************************************************
:: Configure efuse to enable secure boot
::*************************************************************************
%Jlink% -SelectEmuBySN %jlink_serial_number% -Device MIMXRT685S_M33 -IF SWD -Speed auto -ExitOnError -CommanderScript rkth_otpmaster.jlink

pause
```

Figure 27. SBL Jlink script configuration

18). Open the `sign_enc_sfw.bat`.

25. Set the `signing_type` to RSA2048.
26. Modified the `sfw2_otfad_arg` as shown in Figure 28.

```
@echo off
SET "PATH=C:\nxp\MCUX_Provi_v3.1\bin\tools\elftosb\win;%PATH%"
SET "PATH=C:\nxp\MCUX_Provi_v3.1\bin\tools\blhost\win;%PATH%"
SET "PATH=C:\nxp\MCUX_Provi_v3.1\bin\tools\cst\mingw32\bin;%PATH%"
SET "PATH=C:\nxp\MCUX_Provi_v3.1\bin\tools\image_enc\win;%PATH%"
SET imgtool_path=..\..\..\component\secure\mcuboot\scripts

SET user_kek=kek=0102030405060708090a0b0c0d0e0f00
SET sfw2_otfad_arg=otfad_arg=[00112233445566778899aabbccddeeff,0020406001030507,0x08101000,0x6000]

@echo on

::*****************************************************************************
:: Configure signing method RSA2048, ECDSAP256 or ROM_API
::*****************************************************************************
set signing_type=RSA2048

set mcu_header_size=0x400
```

Figure 28. SFW Jlink script configuration

27. Run `sign_enc_sfw.bat` in scons window and generate the `sfw_2_enc.bin` file, see Figure 29.

```
nxf55062@NXL53369 C:\Users\nxf55062\Documents\MCU_SDK\OTA\github_release\sbl\target\evkmimxrt600
> cd secure

nxf55062@NXL53369 C:\Users\nxf55062\Documents\MCU_SDK\OTA\github_release\sbl\target\evkmimxrt600\secure
> sign_enc_sfw2.bat

set signing_type=RSA2048

set mcu_header_size=0x400

if not exist ".\sfw2.bin" (
echo Can't find file sfw2.bin
 pause
 exit
)

if RSA2048 == ROM_API (
elftosb -V -f rt6xx -J .\signed_sfw2_xip.json
 python img_helper.py paddingimage --pad-size 0x400 --input .\sfw_2_signed.bin --output .\sfw_2_padding.bin
 image_enc.exe hw_eng=otfad ifile=.\sfw_2_padding.bin ofile=.\sfw_2_enc.bin base_addr=0x08100000 kek=0102030405060708090a0b0c0d0e0f00 otfad_arg=[00112233445566778899aabbccddeeff,002
 python img_helper.py extract-keycontext --type otfad --enc_image .\sfw_2_enc.bin --output .\sfw_2_keyblob.bin
 python ..\..\..\component\secure\mcuboot\scripts\imgtool.py create --align 4  --version "1.1"  --header-size 0x400 --pad-header --slot-size 0x100000 --key-info .\sfw_2_keyblob.bin
 python img_helper.py merge --header-size 0x400 --sign-image .\sfw_2_bootheader.bin --enc-image .\sfw_2_enc.bin
)  else (
python img_helper.py paddingimage --pad-size 0x400 --input .\sfw2.bin --output .\sfw_2.bin
 image_enc hw_eng=otfad ifile=.\sfw_2.bin ofile=.\sfw_2_enc.bin base_addr=0x08100000 kek=0102030405060708090a0b0c0d0e0f00 otfad_arg=[00112233445566778899aabbccddeeff,00204060010305
 python img_helper.py extract-keycontext --type otfad --enc_image .\sfw_2_enc.bin --output .\sfw_2_keyblob.bin
 if RSA2048 == RSA2048 (python ..\..\..\component\secure\mcuboot\scripts\imgtool.py sign --key ..\..\..\component\secure\mcuboot\scripts\sign-rsa2048-priv.pem --align 4 --version "1
-max-sectors 32 --key-info .\sfw_2_keyblob.bin  .\sfw2.bin .\sfw_2_sign.bin )  else (python ..\..\..\component\secure\mcuboot\scripts\imgtool.py sign --key ..\..\..\component\secure
n "1.1" --header-size 0x400 --pad-header --slot-size 0x100000 --max-sectors 32 --key-info .\sfw_2_keyblob.bin  .\sfw2.bin .\sfw_2_sign.bin )
 python img_helper.py merge --header-size 0x400 --sign-image .\sfw_2_sign.bin --enc-image .\sfw_2_enc.bin
)
Image for OTFAD has been generated succesfully, summary as below:
kek otp words  = 0x000f0e0d 0x0c0b0a09 0x08070605 0x04030201
        kek    = 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x00
        ctx[0]: start = 0x08101000, end = 0x08106ffb
        key    = 0x00 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x99 0xaa 0xbb 0xcc 0xdd 0xee 0xff
        counter= 0x00 0x20 0x40 0x60 0x01 0x03 0x05 0x07
        ctx[1]: start = 0x00000000, end = 0x00000000
        key    = 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
        counter= 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
        ctx[2]: start = 0x00000000, end = 0x00000000
        key    = 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
        counter= 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
        ctx[3]: start = 0x00000000, end = 0x00000000
        key    = 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
        counter= 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
sfw_2_enc.bin are the final image, press any key to download them
```

Figure 29. Run sign_enc_sfw.bat

28. Prepare the RT600 EVK board:
    a. Connect the Jlink to the board at JTAG interface.
    b. Make sure that the jump JP2 is open.
    c. Connect USB cable to the board at JP5.
    d. Set SW5 to ON, OFF, OFF to make RT600 ISP enter into **serial ISP** mode.
    e. Run sign_sbl_app.bat in the scons window and generate the signed and encrypted `sbl`, and `signed sfw` file, Figure 30.

f. Download `sbl`, and `signed sfw` file on the RT600 EVK board.

```
> sign_enc_sbl_app.bat

set signing_type=RSA2048

set mcu_header_size=0x400

if not exist ".\sbl.bin" (
echo Can't find file sbl.bin
 pause
)

elftosb -V -f rt6xx -J .\signed_sbl_xip.json
Parsing configuration file: .\signed_sbl_xip.json.
        No "multicoreImages" section present in configuration file: .\signed_sbl_xip.json.
        Used "imageLinkAddress" value: 134221824.
        Used "imageBuildNumber" value: 1.
        Success.
Starting processing image....
1. Check of the image file.
        Success. (File ./sbl.bin: Size = 70171 bytes, AlignedSize = 70172 bytes)
2. Checking multicore configuration.
        Image is not containing multicore data.
        Success.
3. Fetching of image configuration: execution target and security.
        Internal flash (XIP) - plain signed: image will be signed based on provided configuration.
        Success.
3.1 Checking image link address configuration.
        Image link address will be set to: 0x08001000
        Success.
3.2 Checking image trust zone configuration.
        Trust zone enabled image: configuration of TZM-M_Preset disabled -> TZM-M_PresetFile is ignored and not used.
3.3 Checking image HW user mode keys enablement for all security levels.
        HW user mode key disabled.
        Success.
Start to generate signed image!
4. Load the root certificates.
4.1 Load the count of root certificates.
        Success. (Root Certificate Count = 4)
4.2 Load selected certificate chain id, used to sign this image.
        Success. (Selected certificatate chain index = 0)
4.3 Load all root certificates.
        Root certificate 0 is self signed.
        Success. (Root Certificate 0 = ./crts/ROT1_sha256_2048_65537_v3_ca_crt.der)
        Root certificate 1 is self signed.
        Success. (Root Certificate 1 = ./crts/ROT2_sha256_2048_65537_v3_ca_crt.der)
        Root certificate 2 is self signed.
        Success. (Root Certificate 2 = ./crts/ROT3_sha256_2048_65537_v3_ca_crt.der)
        Root certificate 3 is self signed.
        Success. (Root Certificate 3 = ./crts/ROT4_sha256_2048_65537_v3_ca_crt.der)
4.4 Calculate size of root certificates.
        Success. (Root Certificate Size = 819 bytes, Aligned Size = 820 bytes)
5. Load all certificates in selected certificate chain.
5.1 Load the count of chained certificates in selected certificate chain.
        Success. (Certificate count = 2)
5.2 Load and parse certificates in selected certificate chain.
        Success. ( Chained certificate 0 (./crts/ROT1_sha256_2048_65537_v3_ca_crt.der))
        Success. ( Chained certificate 1 (./crts/IMG1_1_sha256_2048_65537_v3_usr_crt.der))
```

Figure 30. Run sign_sbl_app.bat

29. Set SW5 to ON, OFF, ON to make RT600 ISP enter into **boot from flexspi port b** mode.
30. Press SW3, reset RT600.
31. Open the serial port terminal on the PC, and you see the log `hello sfw image1` as shown in Figure 31, which represents SBL and SFW has run successfully.

```
hello sbl.
Bootloader Version 1.1.0
Remap type: none

The image now in PRIMARY_SLOT slot

Bootloader chainload address offset: 0x100000
Reset_Handler address offset: 0x100400
Jumping to the image


hello sfw image1.
Current image verison: 1.0.0
U-Disk updating task enable.
SD Card updating task enable.
Hello world1.
Hello world2.
Please plug in a u-disk to board.
Hello world1.
Hello world2.
Hello world1.
Hello world2.
Hello world1.
Hello world2.
Hello world1.
```

Figure 31. SBL and SFW successfully run logs

32. Put the `sfw_2_enc.bin` generated by `sign_enc_sfw.bat` into the SD card or U-disk, and rename it to `newapp.bin`. Here, take U-disk as an example for demonstration:

    a. Connect the USB cable to the J6 and ensure that the power supply is ON.

    b. Plug the U-disk to the RT600 EVK board and update the image.

    For more details, see Figure 32.

```
hello sbl.
Bootloader Version 1.1.0
Remap type: none

The image now in PRIMARY_SLOT slot

Bootloader chainload address offset: 0x100000
Reset_Handler address offset: 0x100400
Jumping to the image


hello sfw image1.
Current image verison: 1.0.0
U-Disk updating task enable.
SD Card updating task enable.
Hello world1.
Hello world2.
Please plug in a u-disk to board.
Hello world1.
Hello world2.
Hello world1.
Hello world2.
mass storage device attached:pid=0x1666vid=0x951 address=1
U-Disk OTA test
fatfs mount as logiacal driver 1......success
Hello world1.
Hello world2.
Hello world1.
Hello world2.
reading...
new img verison: 1.1.0
updating...
finished
write update type = 0x2
write magic number offset = 0xffff0
Please unplug the u-disk!
sys rst...
```

Figure 32. SBL and SFW successfully update logs

33. Unplug the U-disk when you see the log **Please unplug the u-disk** as shown in Figure 33.
34. Reset the RT600 and you see the log **hello sfw image2** as shown in Figure 33, which represents SFW has been updated successfully.

```
hello sbl.
Bootloader Version 1.1.0
Remap type: none

The image now in SECONDARY_SLOT slot

Bootloader chainload address offset: 0x100000
Reset_Handler address offset: 0x100400
Jumping to the image


hello sfw image2.
Current image verison: 1.1.0
U-Disk updating task enable.
SD Card updating task enable.
Hello world1.
Hello world2.
Please plug in a u-disk to board.
Hello world1.
Hello world2.
Hello world1.
Hello world2.
```

Figure 33. SBL and SFW successfully update and boot logs

So far, FOTA has demonstrated the signature + encryption function on the RT600 EVK board.

4 Conclusion

When using FOTA, some notes worth paying attention to:

1. This application note only introduces the security-related upgrade method based on remapping. For functions such as single image, and ISP, refer to *MCU-OTA SBL and SFW User Guide* (document MCUOTASBLSFWUG).
2. In the actual operation process, since OTP can only be burned once, the method of writing shadow through Jlink is used instead of burning some OTP. For the Jlink script involved in this article, see the related software released with this application note.
3. This application note does not describe the design architecture and specific content of SBL and SFW. For more details, refer to *FOTA Design for SBL and SFW* (document AN13460).

5 References

- SBL Repository Link: https://github.com/NXPmicro/sbl
- SFW Repository Link: https://github.com/NXPmicro/sfw
- *MCU-OTA SBL and SFW User Guide* (document MCUOTASBLSFWUG)
- *FOTA Design for SBL and SFW* (document AN13460)

6 Revision history

Table 2 summarizes the changes done to this document since the initial release.

Table 2. Revision history

Revision number	Date	Substantive changes
0	06 June 2022	Initial 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 http://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 data sheet 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.

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.

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.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Airfast — is a trademark of NXP B.V.

Bluetooth — the Bluetooth wordmark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NXP Semiconductors is under license.

Cadence — the Cadence logo, and the other Cadence marks found at www.cadence.com/go/trademarks are trademarks or registered trademarks of Cadence Design Systems, Inc. All rights reserved worldwide.

CodeWarrior — is a trademark of NXP B.V.

ColdFire — is a trademark of NXP B.V.

ColdFire+ — is a trademark of NXP B.V.

EdgeLock — is a trademark of NXP B.V.

EdgeScale — is a trademark of NXP B.V.

EdgeVerse — is a trademark of NXP B.V.

eIQ — is a trademark of NXP B.V.

FeliCa — is a trademark of Sony Corporation.

Freescale — is a trademark of NXP B.V.

HITAG — is a trademark of NXP B.V.

ICODE and I-CODE — are trademarks of NXP B.V.

Immersiv3D — is a trademark of NXP B.V.

I2C-bus — logo is a trademark of NXP B.V.

Kinetis — is a trademark of NXP B.V.

Layerscape — is a trademark of NXP B.V.

Mantis — is a trademark of NXP B.V.

MIFARE — is a trademark of NXP B.V.

MOBILEGT — is a trademark of NXP B.V.

NTAG — is a trademark of NXP B.V.

Processor Expert — is a trademark of NXP B.V.

QorIQ — is a trademark of NXP B.V.

SafeAssure — is a trademark of NXP B.V.

SafeAssure — logo is a trademark of NXP B.V.

StarCore — is a trademark of NXP B.V.

Synopsys — Portions Copyright © 2021 Synopsys, Inc. Used with permission. All rights reserved.

Tower — is a trademark of NXP B.V.

UCODE — is a trademark of NXP B.V.

VortiQa — is a trademark of NXP B.V.

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

Date of release: 06 June 2022
Document identifier: AN13478