

RX Family

Application Design Using the RX74M First Stage Bootloader

Introduction

The RX74M Group has an inalterable first stage bootloader (FSBL). The FSBL is executed after a reset, which verifies the OEM bootloader (OEM_BL) stored in MRAM and only passes control to the OEM_BL when the verification is successful. This allows building the boot chain starting from the inalterable FSBL.

The FSBL of the RX74M supports a CRC boot in which the integrity of the OEM_BL is verified by CRC32, and an HMAC boot in which the authenticity and integrity of the OEM_BL are verified by HMAC-SHA256 through the use of signature verification and a device-specific key. In HMAC boot, the OEM_BL is verified by using an ECDSA P-256 signature at the time of production programming.

This application note explains the settings for using the FSBL with an RX74M, the generation of keys and certificates required for verification, production programming that uses the Renesas flash programmer, and the procedures for checking the operations of a CRC boot and HMAC boot by using the included sample project.

Required Resources

Target Devices

- RX74M (with RSIP-E50D)

Software and development tools

- e² studio IDE v2026-07
- Renesas Flexible Software Package (FSP) v1.0.0
The links to download the above software are available at <https://github.com/renesas/rx-fsp>.
- Renesas Security Key Management Tool (SKMT) v1.12
- Renesas Flash Programmer (RFP) v3.24
- GNU Privacy Guard for Windows: [Gpg4win](#)
- Renesas Key Wrap Server

Hardware

- EK-RX74M, Evaluation Kit for the RX74M Group (renesas.com/rx/ek-rx74m)
- Workstation running Windows®11 and the Tera Term console or similar application
- One USB Type-C® cable (USB Type-C® male to USB Type-C® male)

The terms used in this document are explained below.

Table 1-1 Terminology

Term	Description
RSIP	Renesas Secure IP A security engine that is integrated into some Renesas MCUs. The RX74M is provided with RSIP-E50D.
HUK	Hardware Unique Key A device-specific encryption key that exists within the device and is used to protect key data.
UFPK	User Factory Programming Key A user-set keyring used to generate an encrypted key from a user key during key injection, which is set by users. The UFPK is not used on the device.
W-UFPK	Wrapped UFPK Key information that is generated by wrapping a UFPK in an HRK (a common encryption key, which only exists within the RSIP and the secure room of Renesas). This is used after being decrypted into the UFPK by the HRK within the RSIP.
MRAM	Magnetoresistive Random Access Memory This is used as non-volatile RAM (NVRAM) in place of flash memory in process nodes at certain levels. The RX74M is provided with code MRAM and data MRAM.
RFP	Renesas Flash Programmer A tool for writing programs to the NVRAM of Renesas MCUs.
SKMT	Security Key Management Tool A tool supporting secure key injection and firmware update.
OEM_BL	A program that the FSBL verifies before bootup. A second stage bootloader or application program developed by an OEM.
SSBL	Second Stage Bootloader A bootloader that is executed after having been verified by the FSBL.

Reference Information

The demo project included with this application note uses the Renesas e² studio IDE. In addition, knowledge of application boot loading and cryptographic algorithms is desirable. As required, refer to the following sections in the *RX74M Group User's Manual: Hardware*.

- Security Features
- Option-Setting Memory (OFSM)
- MRAM

This application note is intended for product developers, product manufacturers, product support personnel, and end users involved in application development using the first-stage bootloader on Renesas RX Family MCUs.

Contents

1. Overview of the First Stage Bootloader (FSBL)	5
1.1 The Purpose and Role of the FSBL	5
1.2 FSBL in the RX74M.....	6
1.3 Overall Workflow from Production Programming to Bootup	7
1.3.1 Workflow for Application of Authentication Information in a CRC Boot.....	7
1.3.2 Workflow for Application of Authentication Information in an HMAC Boot.....	8
2. Security Model and Operation of the RX74M FSBL	9
2.1 Root of Trust and Chain of Trust.....	9
2.2 Keys and Authentication Information for Use in an HMAC Boot.....	11
2.2.1 OEM_BL Verification Root Key and OEM_BL Verification Key	11
2.2.2 Key Certificate	12
2.2.3 Code Certificate.....	13
2.3 Verification at the Time of Bootup in Single-Chip Mode	15
3. Sample Project for a CRC Boot.....	18
3.1 CRC Boot Settings for the RX74M FSBL.....	18
3.2 Create the Code Certificate.....	20
3.3 Using the RFP for Programming and Checking Operations	22
4. Sample Project for an HMAC Boot	25
4.1 Preparing Keys.....	25
4.1.1 Generate ECC Key Pairs by Using OpenSSL.....	25
4.1.2 NIST CAVP Test Vectors as ECC Key Pairs	26
4.2 Implementing the Root of Trust.....	27
4.2.1 Generate the Wrapped User Factory Programming Key (W-UFPK)	27
4.2.2 Generate the Wrapped OEM_ROOT_PK by Using the SKMT CLI	29
4.2.3 Generate the Wrapped OEM_ROOT_PK by Using the SKMT GUI.....	30
4.3 HMAC Boot Settings for the RX74M FSBL	32
4.4 Anti-Rollback Counters.....	35
4.5 Generating Certificates.....	37
4.5.1 Using OpenSSL and SKMT Command Line Interface.....	37
4.5.2 Using NIST ECC Key Pairs and SKMT GUI	38
4.6 Authenticated Production Programming	41
5. Appendix.....	45
5.1 Using the FSBL with Renesas Secure Factory Programming	45
5.2 Register Settings for the FSBL.....	45
6. References	47
7. Website and Support	48
Revision History	49

1. Overview of the First Stage Bootloader (FSBL)

A secure embedded system requires checking, before starting its execution, that the software to be run has been created by an intended provider and that it has not been altered after being stored. When this checking is to be done by the bootloader, the bootloader itself must be in a reliable state.

The first stage bootloader (FSBL) is a bootloader that is first to be run after a reset and verifies the software image that is next to be run. In this document, the image that the FSBL directly verifies before passing control to it is called the OEM_BL. There are two configurations for the OEM_BL: one that directly places the product application and another that places a second stage bootloader like MCUboot.

In the RX74M, a user-inalterable FSBL is implemented. The FSBL of the RX74M provides CRC32-based integrity verification for the OEM_BL or HMAC boot combining signature authentication and HMAC-SHA256. This section describes the role of the FSBL and overall processing by the RX74M. The information for use in authentication and detailed processing for verification are explained in section 2.

1.1 The Purpose and Role of the FSBL

In a secure system, an application program must be executed only after confirming that it has not been altered maliciously or accidentally. This confirmation, typically performed by a bootloader, can be a simple integrity check, or it involves signature verification to guarantee reliability. However, it is important to note that the legitimacy of the bootloader itself must also be guaranteed.

The following is a high-level representation of the usage of the FSBL in an embedded system in MCU single-chip mode.

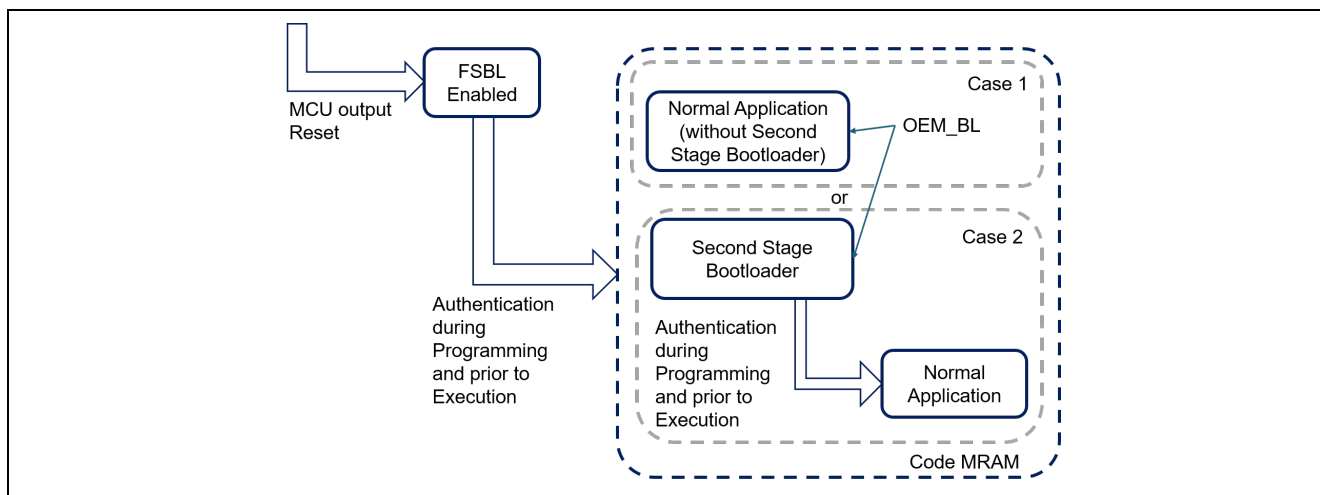


Figure 1-1 Usage of the FSBL in Single-Chip Mode

1.2 FSBL in the RX74M

When the FSBL is enabled, there are two options for verifying the OEM_BL.

- CRC boot: Integrity check using CRC32. This check will be performed at initial MCU programming and prior to execution.
- HMAC boot: Authenticity check. During initial MCU programming, the OEM_BL is authenticated by ECDSA using the secp256r1 (NIST P-256) ECC curve. Prior to execution, the OEM_BL is authenticated by HMAC-SHA256 with a HUK in the MCU.

To secure a strong root of trust, the HMAC boot sequence is implemented in stages starting from an invariant entity. Writing of an invariant FSBL to the RX74M provides the root of trust as the start of the HMAC boot sequence. The FSBL first verifies the OEM_BL, after which the OEM_BL verifies a normal application program to form a chain of trust (see Case 2 in Figure 1-1 for the authentication path) to boot the system.

The process of a CRC boot is simpler than that of an HMAC boot. The features of a CRC boot when compared with an HMAC boot are as follows.

Table 1-1 Comparison of CRC Boot and HMAC Boot

Item	CRC Boot	HMAC Boot
Root of trust	Not required	Required
Key certificate	Not required	Required
Code certificate	Required	Required
ECDSA signature	Not required	Required
Anti-rollback	Not required	Required
Integrity verification	CRC32	HMAC-SHA256
Authenticity verification	—	✓

1.3 Overall Workflow from Production Programming to Bootup

The root of trust, key certificate, and code certificate described up to the previous section do not exist individually, but these are used in interlinked ways at the time of production programming and in the authentication process at the time of bootup. The FSBL uses these items of authentication information to build a chain of trust to verify that the OEM_BL has not been altered and has been generated by an authorized developer.

The authentication information and verification procedure for use in authentication differ between a CRC boot and HMAC boot. In a CRC boot, the integrity of the OEM_BL is checked by using the CRC value stored in the code certificate. On the other hand, in an HMAC boot, the authenticity and integrity of the OEM_BL are verified by the combination of the root of trust, key certificate, code certificate, and HMAC digest.

The following describes how each item of authentication information is used at the time of production programming in the respective cases of a CRC boot and HMAC boot.

1.3.1 Workflow for Application of Authentication Information in a CRC Boot

During the programming stage of the OEM_BL with the RFP, the following items will be programmed in the MCU.

- OEM_BL (application image)
- Code certificate

The following figure summarizes the overall workflow using the SKMT and RFP during production programming when using a CRC boot. A step-by-step walk-through of using the SKMT and RFP is provided in section 3.

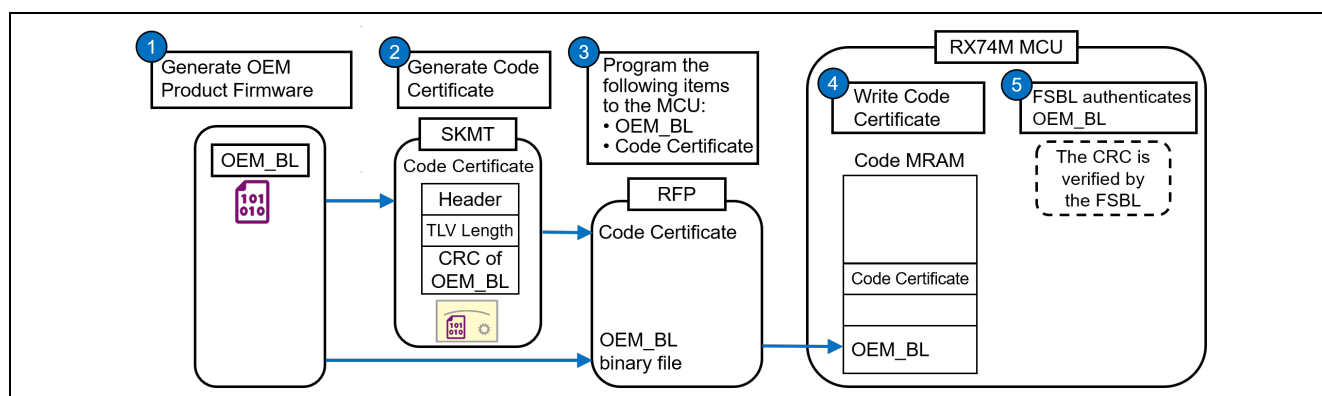


Figure 1-2 Summary of Authentication Information Usage during Production Programming for a CRC Boot

1.3.2 Workflow for Application of Authentication Information in an HMAC Boot

During the programming stage of the OEM_BL with the RFP, the following items will be programmed in the MCU.

- Hash of OEM_ROOT_PK
- OEM_BL (application image)
- Code certificate

The following figure summarizes the overall workflow using the SKMT and RFP during production programming when using an HMAC boot. A step-by-step walk-through of using the SKMT and RFP is provided in section 4.

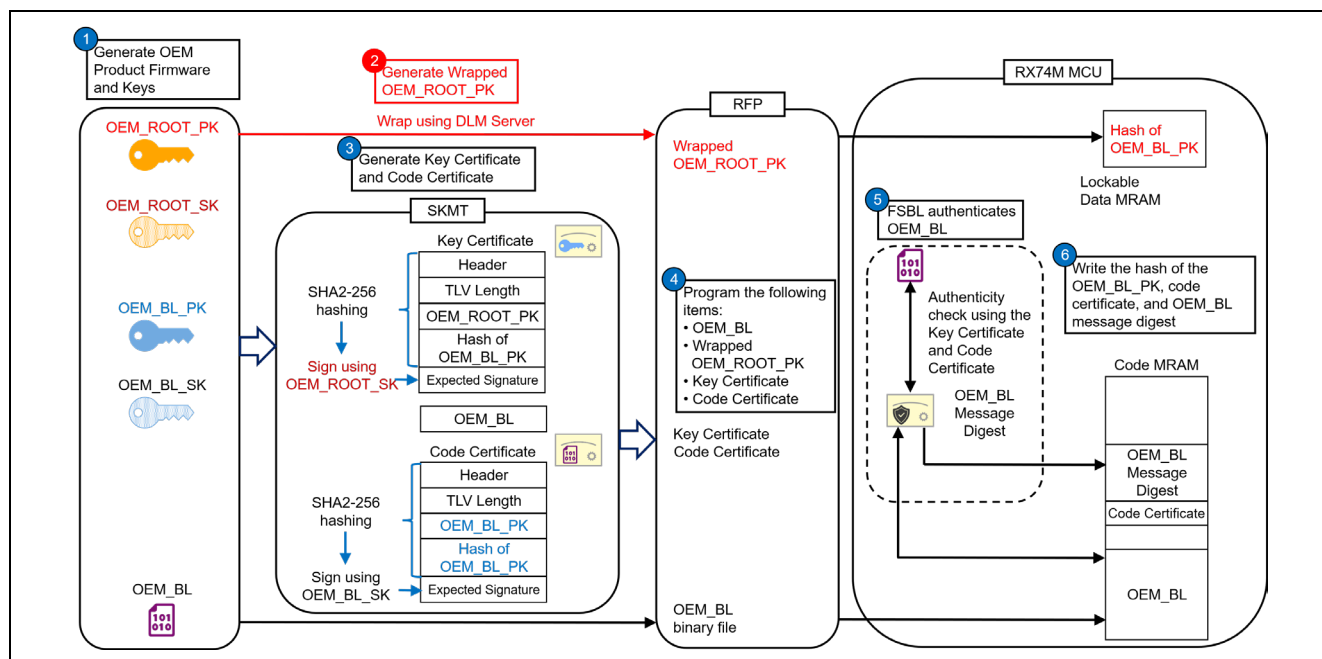


Figure 1-3 Summary of Authentication Information Usage during Production Programming for an HMAC Boot

2. Security Model and Operation of the RX74M FSBL

In an HMAC boot, a chain of trust is built by using several items of authentication information in order to guarantee the authenticity and integrity of the OEM_BL. At the time of system bootup, each component verifies the next component, which guarantees the legitimacy of the OEM_BL and application that are finally to be run.

This section explains the security model of an HMAC boot. First, it explains the root of trust which serves as the origin of confidence in the system, and then explains the method of protection by the key certificate, code certificate, and anti-rollback function. By understanding how these items work together to form a chain of trust, you can easily understand the subsequent certificate generation and production programming procedures.

A summary of the chain of trust in the case of an HMAC boot of the FSBL is described below. The chain of trust is built with the FSBL written to an inalterable area as a point of origin. The FSBL verifies the OEM_BL based on the public key for the OEM_BL verification root key pair, and only permits execution of the OEM_BL if the verification was successful.

A CRC boot is not for verifying the legitimacy of the OEM_BL, so the explanation in this section focuses on an HMAC boot.

2.1 Root of Trust and Chain of Trust

An HMAC boot requires checking before system bootup that the software to be run has been generated by an authorized developer and has not been altered. This scheme is called the chain of trust, which is realized by successive verification of each component by a trustworthy component.

In boot processing with the use of the FSBL for the RX74M, the authenticity and integrity of the OEM_BL are verified by utilizing the public key for the OEM_BL verification root key pair (OEM_ROOT_PK), key certificate, and code certificate with the FSBL written as the root of trust to an inalterable area. The OEM_BL is only executed if the verification was successful.

Figure 2-1 shows the chain of trust with the FSBL.

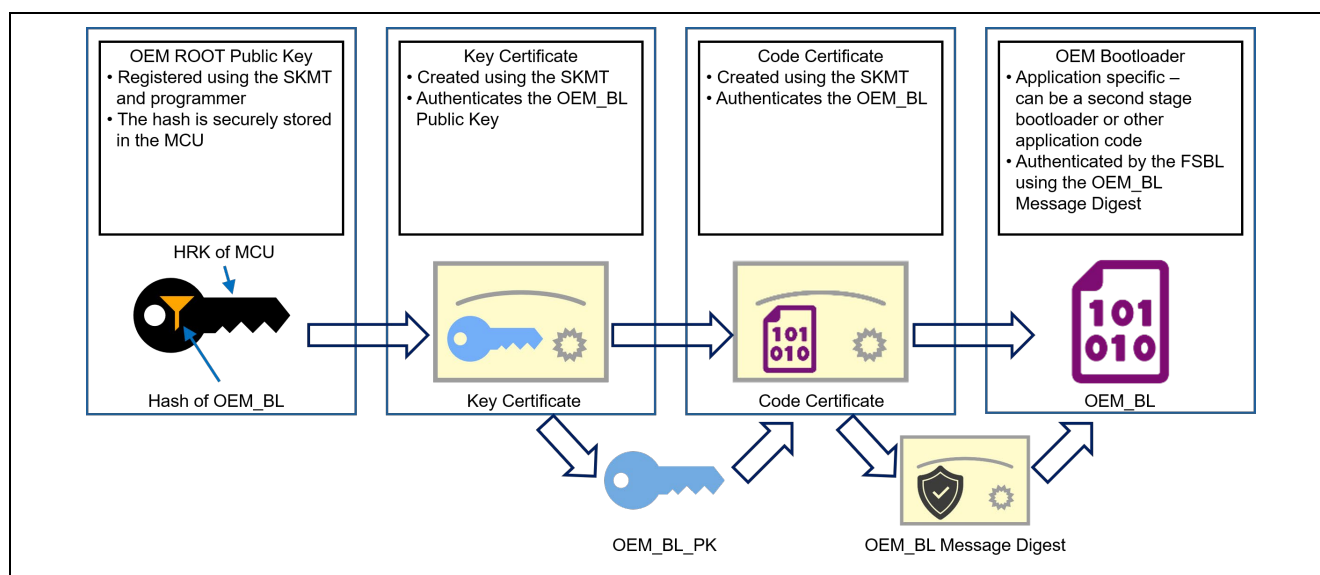


Figure 2-1 Chain of Trust

The main items of authentication information constituting a chain of trust are as follows.

- Root of Trust

An SHA-256 hash value of the OEM_ROOT_PK which serves as the origin of confidence in the system.

- Key Certificate

A certificate used to guarantee the legitimacy of the public key for the OEM_BL verification key pair (OEM_BL_PK).

- Code Certificate

A certificate that holds the verification information of the OEM_BL image.

- OEM_BL_Digest

An HMAC-SHA256 digest for use in authentication at the time of execution when an HMAC boot is enabled.

The following section explains the structure and role of these items of authentication information.

2.2 Keys and Authentication Information for Use in an HMAC Boot

2.2.1 OEM_BL Verification Root Key and OEM_BL Verification Key

Two sets of secp256r1 ECC key pairs must be generated when using an HMAC boot.

- OEM_BL verification root key pair
 - OEM_ROOT_PK: the public key for the OEM_BL verification root key pair
 - OEM_ROOT_SK: the secret key for the OEM_BL verification root key pair
- OEM_BL verification key pair
 - OEM_BL_PK: the public key for the OEM_BL verification key pair
 - OEM_BL_SK: the secret key for the OEM_BL verification key pair

The SHA256 hash of the OEM_ROOT_PK is stored in the option setting memory (HOEMRTPK) by using the process described in the “Secure Key Injection” section of the *RX74M Group User's Manual: Hardware*. The injection of the root of trust is done during production programming of the application.

2.2.2 Key Certificate

The key certificate is used to verify the legitimacy of the OEM_BL_PK during production programming for HMAC boot enabled applications.

Below is some important information about the creation and usage of the key certificate.

- The key certificate is signed by the OEM_ROOT_SK.
- The OEM_ROOT_PK is included in the signed key certificate. It is verified by the value stored in the HOEMRTPK.
- The hash of the OEM_BL_PK is included in the key certificate. It is used by the OEM_BL to verify the OEM_BL_PK.
- The key certificate is discarded after the OEM_BL PK has been verified.

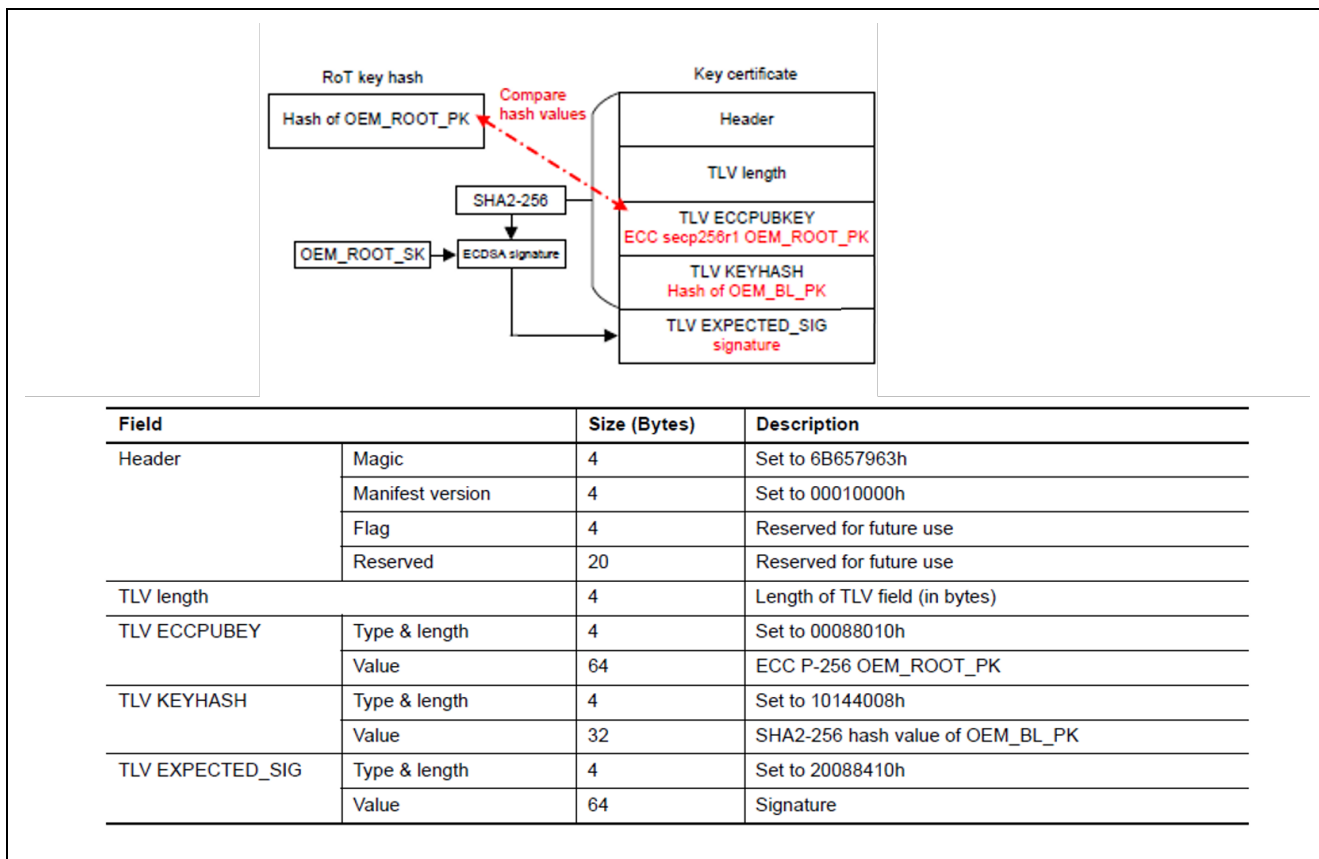


Figure 2-2 Key Certificate to Verify the OEM_BL_PK

2.2.3 Code Certificate

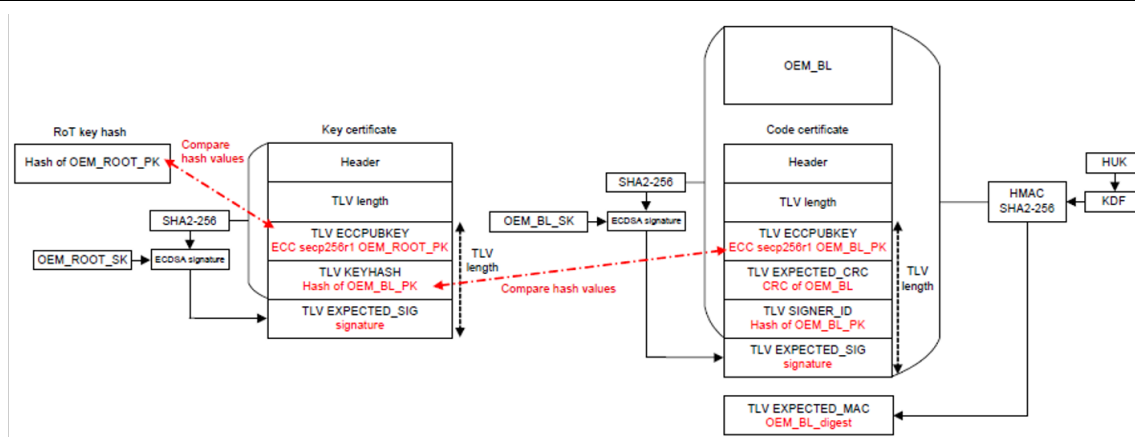
The code certificate holds information required for the verification of the OEM_BL. The code certificate is used for both a CRC boot and an HMAC boot and referenced during production programming and at the time of bootup in single-chip mode.

- The code certificate is signed by the OEM_BL_SK.
- The OEM_BL_PK is included in the code certificate. It is verified by the hash stored in the key certificate. The OEM_BL_PK is then used to verify the OEM_BL's signature included in the code certificate.
- The code certificate contains authentication information for use in the FSBL verification process during single-chip mode. It must be programmed into the MRAM (code MRAM or data MRAM) after the OEM_BL has been programmed.
- The application image version number is part of the code certificate when using an HMAC boot. The range of valid image version numbers depends on the target of verification by the FSBL. It is 1 to 64 when the target of FSBL verification is an SSBL and 1 to 1024 for an application. When the target of FSBL verification is an SSBL, the number can be updated 64 times. The FSBL implements an anti-rollback scheme. Therefore, to update to a new OEM_BL image, the next new image must have a version number that is higher than that of the previous application image. For more details on the operation and explanation of the usage of the image version number, refer to the "Anti-Rollback Counter" section of the *RX74M Group User's Manual: Hardware*.
- The code certificate location in MRAM must be specified by using the SACC0 or SACC1 register. The SACC0 register specifies the start address of the code certificate when the lower bank is Bank 0 in dual bank mode or when the startup area is the normal area in linear mode. The SACC1 register specifies the start address of the code certificate when the lower bank is Bank 1 in dual bank mode or when the startup area is the alternate area in linear mode. Configuration of SACCx registers can be set by the RX FSP Board Support Package (BSP).
- Figure 3-1 and Figure 4-6 are example configurations of the code certificate programming locations. Adjust them as required for your specific application.
- In an HMAC boot, the TLV EXPECTED_CRC field is not used. The structure of the code certificate, however, remains unchanged.
- After successful verification of the OEM_BL during production programming, the HMAC-SHA256 digest (OEM_BL_digest) of the OEM_BL plus the code certificate is calculated by the OEM_BL with the use of a HUK in the MCU. The OEM_BL will program the OEM_BL_digest in the memory area immediately following the code certificate.

Field		Size (Bytes)	Description
TLV EXPECTED_MAC	Type & length	4	Fixed value 30184008h
	Value	32	Unique OEM_BL_digest in each MCU

Figure 2-3 Details of the OEM_BL_digest

Figure 2-4 shows the structure of the code certificate for an HMAC boot.



Field		Size (Bytes)	Description
Header	Magic	4	Set to 636F6463h
	Manifest version	4	Set to 00010000h
	Flag	4	Set to 00000000h
	Load Addr	4	Set to the start address of the OEM_BL
	Dest Addr	4	Set to the start address of the OEM_BL
	Image size	4	Size in bytes of the OEM_BL. Specify a multiple of 16. The minimum size is 64 bytes.
	Image version	4	Version number of the OEM_BL. The setting depends on the setting of ARCMD.
	Build number	4	Set to 00000000h
TLV length		4	Byte length of TLV field
TLV ECCPUBKEY	Type & length	4	Set to 01088010h
	Value	64	ECC secp256r1 OEM_BL_PK
TLV EXPECTED_CRC	Type & length	4	Set to 40000001h
	Value	4	CRC32 of OEM_BL
TLV SIGNER_ID	Type & length	4	Set to 10144008h
	Value	32	SHA2-256 hash value of OEM_BL_PK
TLV EXPECTED_SIG	Type & length	4	Set to 25088410h
	Value	64	OEM_BL signed by signature OEM_BL_SK in [code certificate] When generating the signature, enter the code certificate and OEM_BL in the hash function in this order.

Figure 2-4 Code Certificate to Verify the OEM_BL Using an HMAC Boot

Figure 2-5 shows the code certificate for a CRC boot. The TLV ECCPUBKEY and TLV_EXPECTED_SIG fields are not present in the code certificate. For a CRC boot, there is no anti-rollback policy enforced, and the image version field is not used.

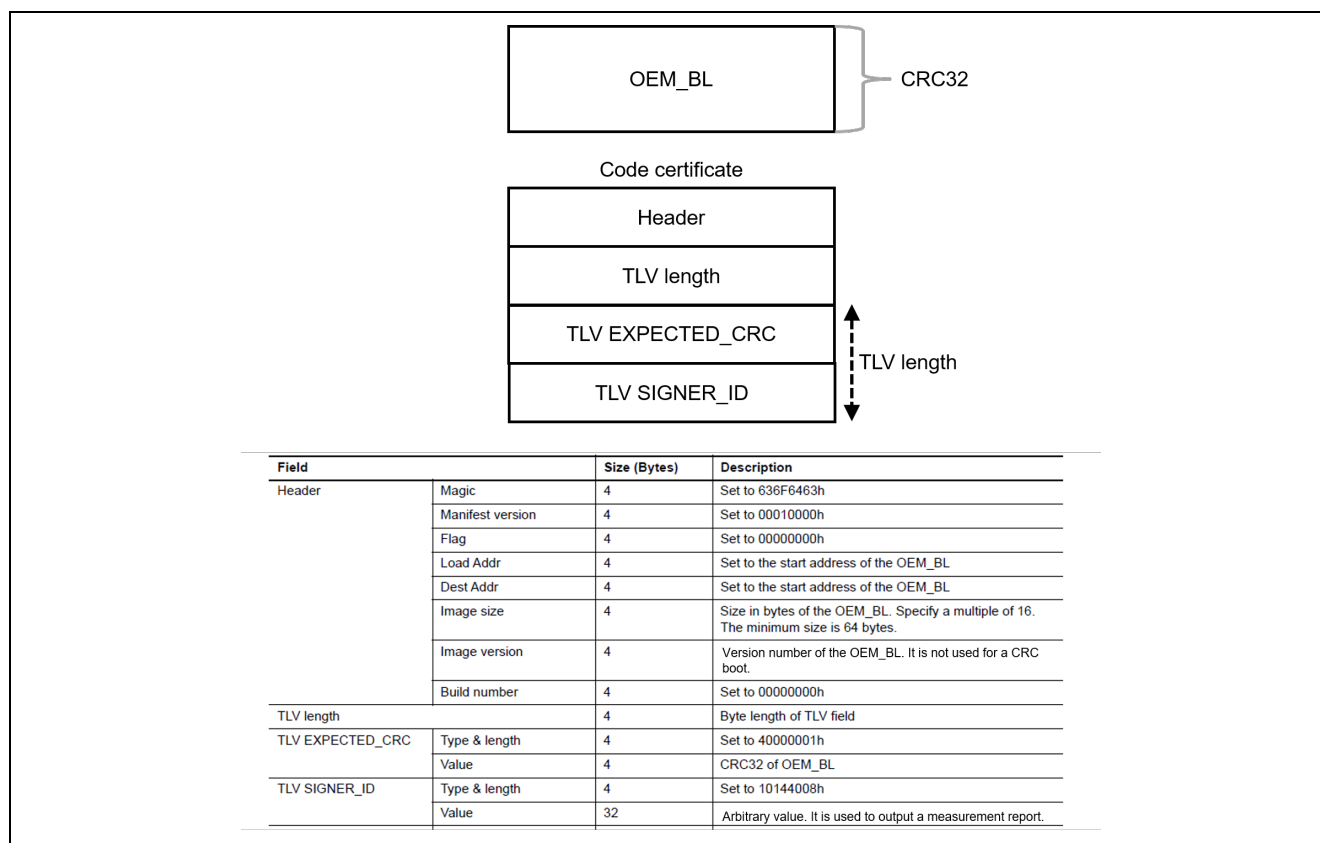


Figure 2-5 Code Certificate to Verify the OEM_BL Using a CRC Boot

2.3 Verification at the Time of Bootup in Single-Chip Mode

For an HMAC boot, the OEM_BL is verified by the boot firmware using an ECDSA during production programming. The boot mode of the RX74M is available from the JTAG, SCI and USB interfaces. For details of the boot mode and respective interfaces, refer to the “Operating Modes” section of the *RX74M Group User’s Manual: Hardware*.

Before booting the OEM_BL, the user programs the hash of the OEM_ROOT_PK to the area called HOEMRTPKn in the system by, for example, using the RFP. The boot firmware then reads the image version number in the code certificate when the MCU is programmed with the OEM_BL.

If the image version number is higher than that of the image which was previously programmed, the boot firmware calculates the hash of the OEM_ROOT_PK (which is stored in the key certificate) and compares it with the hash of the OEM_ROOT_PK that is programmed in the MCU.

Thus, the OEM_BL_PK in the code certificate is verified. Once the OEM_BL_PK is verified, it will be used by the boot firmware to verify the OEM_BL by using the signature in the code certificate.

If the OEM_BL is verified, the boot firmware increments the anti-rollback counter to the image version number written in the code certificate and calculates the OEM_BL_digest. The code certificate and OEM_BL_digest are then programmed into the MCU.

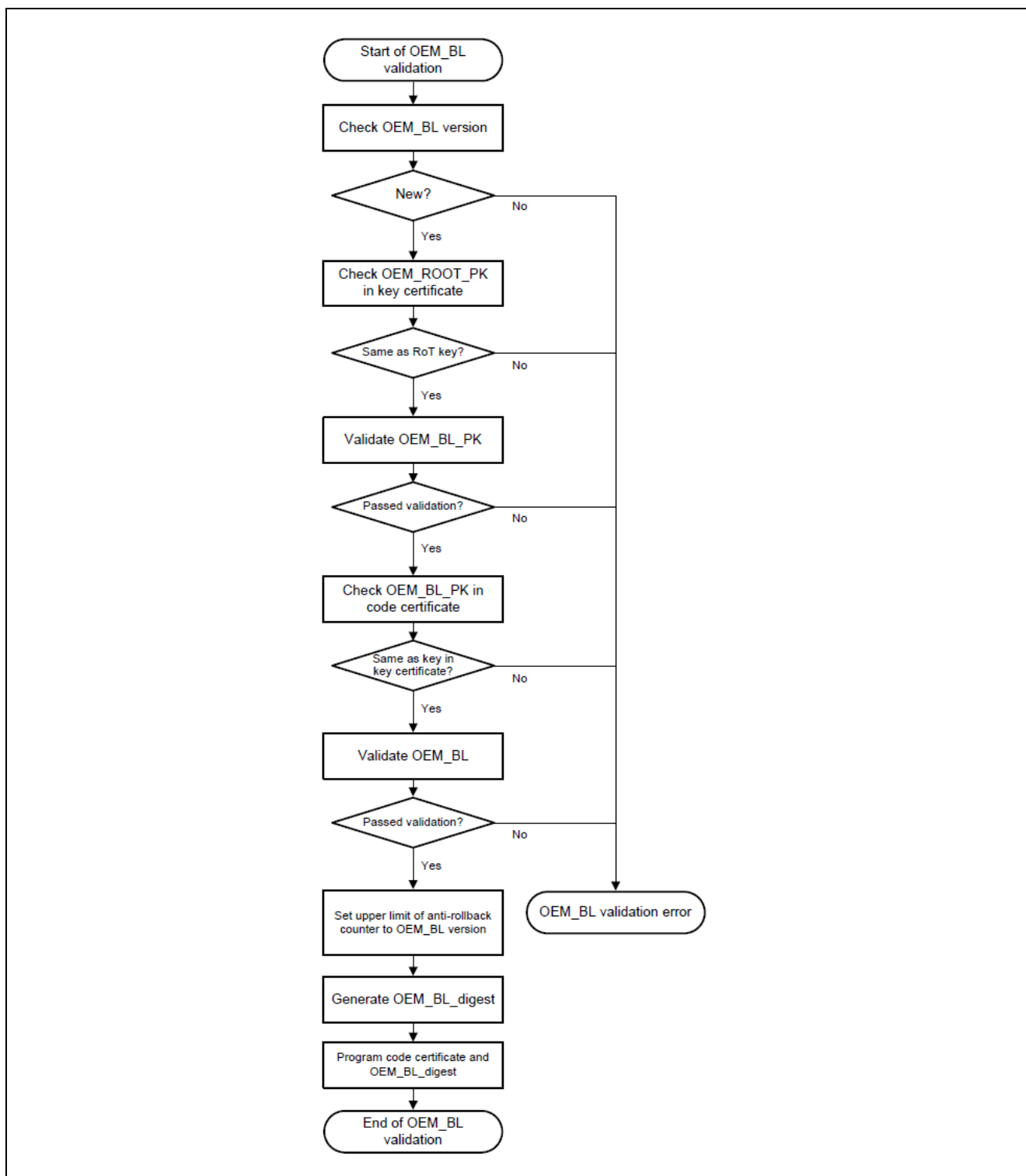


Figure 2-6 OEM_BL Verification Flow in Production Programming Mode

After successful completion of production programming, the RX74M is booted up in single-chip mode, and the FSBL is executed after a reset if the FSBL is enabled. The FSBL verifies the OEM_BL according to the selected boot mode.

When a CRC boot is used, the FSBL calculates the CRC32 value of the OEM_BL stored in the MRAM, and compares it with the expected value stored in the code certificate.

When an HMAC boot is used, the FSBL calculates HMAC-SHA256 for the OEM_BL and code certificate, and compares it with the OEM_BL_digest stored in the MRAM at the time of production programming. If the values from verification match, the FSBL passes control to the OEM_BL.

Figure 2-7 shows the OEM_BL verification flow in single-chip mode.

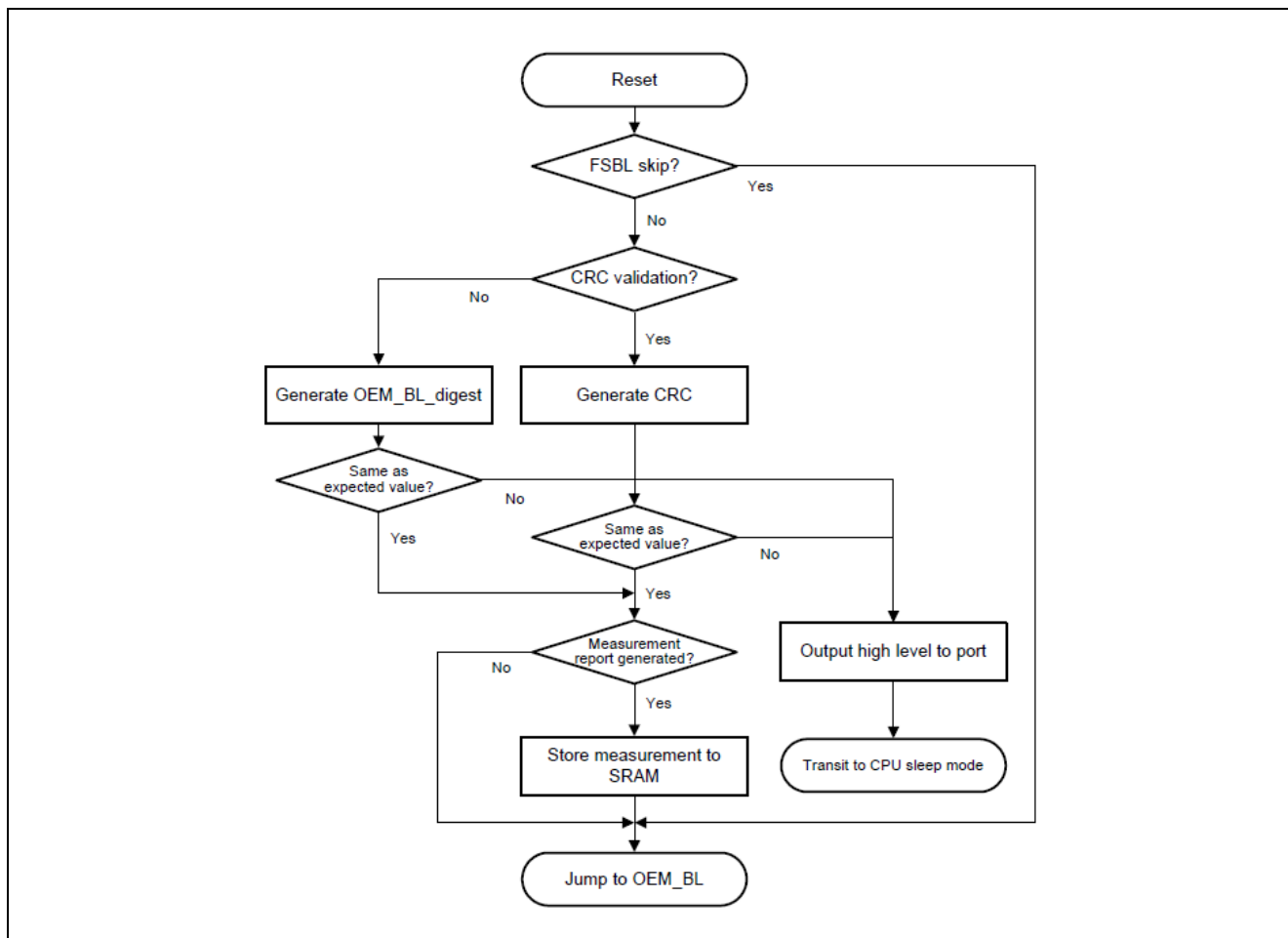


Figure 2-7 Verification of the OEM_BL in Single-Chip Mode

In each boot mode, a configuration for output of the verification results as a measurement report is also selectable. For the procedure for this setting and the report format, refer to section 5.2.

3. Sample Project for a CRC Boot

In this section, how to use the sample project for a CRC boot is explained.

Though it presents the case where the compiler in use is CC-RX, the details of the operations are the same for the GCC compiler.

3.1 CRC Boot Settings for the RX74M FSBL

The blinky project (`blinky_crc_boot_rx74m_ccrx`), which uses a CRC boot, was created on the basis of the blinky project of the FSP. To enable the FSBL, set FSBL Execution Mode to **CRC boot with report measurement** as shown in Figure 3-1. Additionally, pin P604 is set for use as an error report pin.

Import `blinky_crc_boot_rx74m_ccrx` to a workspace, run the `configuration.xml` file, and click **Generate Project Content**. Build and run `blinky_crc_boot_rx74m_ccrx` to verify its functionality.

- The generated Motorola hex file will be used for the creation of the code certificate in the section 3.2.
- By checking that the three LEDs are blinking, you can confirm that the sample project has been started.

Properties Problems Smart Browser Console		
EK-RX74M		
Settings	Property	Value
	▼ RX74M Device Options	
	▼ OFS Registers	
	▼ First Stage Bootloader (FSBL) [CAUTION OTP!]	
	▼ Code Certificate	
	▼ SACC00 (Start Address of Code Certificate 00) Settings	Enabled
	Address	0x86906000
	> SACC10 (Start Address of Code Certificate 10) Settings	Disabled
	> SACC01 (Start Address of Code Certificate 01) Settings	Disabled
	> SACC11 (Start Address of Code Certificate 11) Settings	Disabled
	> SACC02 (Start Address of Code Certificate 02) Settings	Disabled
	> SACC12 (Start Address of Code Certificate 12) Settings	Disabled
	> SACC03 (Start Address of Code Certificate 03) Settings	Disabled
	> SACC13 (Start Address of Code Certificate 13) Settings	Disabled
	▼ FSBLCTRL0 (FSBL Control Register 0) Settings	Enabled
	FSBLEN	Enabled
	FSBLSKIPSW	Enabled
	FSBLSKIPDS	Enabled
	FSBLCLK	300 MHz
	▼ FSBLCTRL1 (FSBL Control Register 1) Settings	Enabled
	FSBLEXMDFSBLEN	CRC boot with report measurement
	▼ FSBLCTRL2 (FSBL Control Register 2) Settings	Enabled
	PORTPN	PORTn04
	PORTGN	PORT6m
	▼ SAMR (Start Address of Measurement Report) Settings	Enabled
	Address	0x82001000
	> ZHUK (Zerorization HUK Register) Settings	Disabled

Figure 3-1 FSBL Settings for a CRC Boot

Open Tera Term and set **Speed** to **115200** and **Data** to **8 bit**.

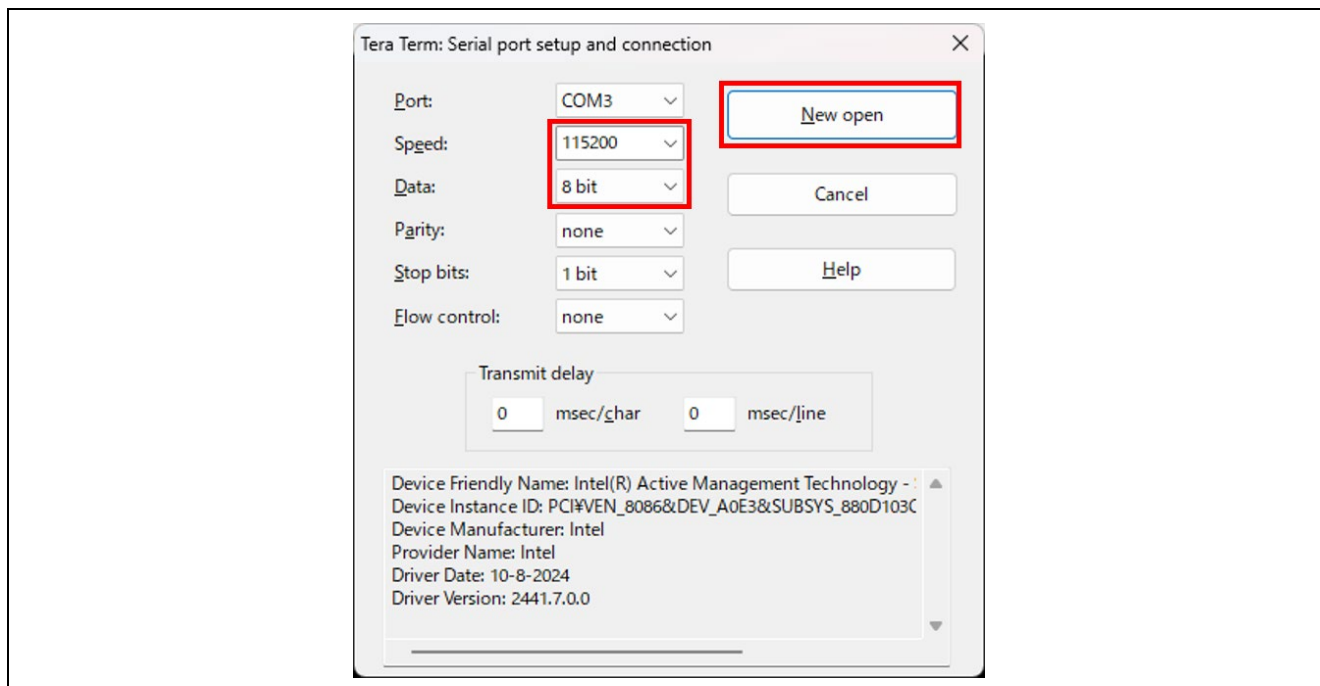


Figure 3-2 Tera Term Settings

Observe the following Tera Term output. Note that since the code certificate is not programmed yet, the output for the CRC is undefined. In addition, since the image version information is not used in a CRC boot, `version_num_oem_bl` output is all 0s.

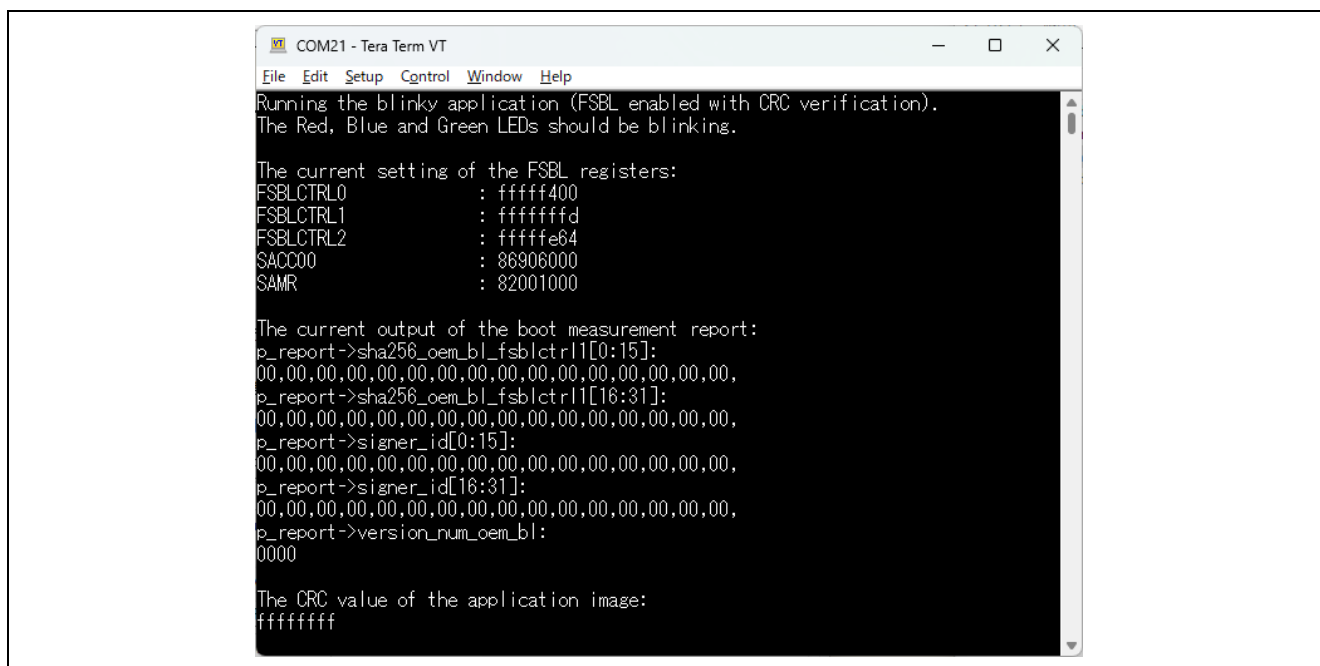


Figure 3-3 Console Output for a CRC Boot

3.2 Create the Code Certificate

The code certificate for a CRC boot must contain the expected CRC of the OEM_BL. If an integrity check has been performed at the time of an MCU reset, the FSBL calculates the CRC of the OEM_BL and compares it with the value stored in the code certificate.

The code certificate is created by using the SKMT CLI or SKMT GUI.

Creating the Code Certificate for a CRC Boot by Using the SKMT CLI

To generate the code certificate, there are several properties to configure.

- Input: /loadaddr: the load address of the OEM_BL
- Input: /cfsize: the code MRAM size of the device to be used
- Input: /oembl: the binary file of the OEM_BL
- Output: /output_codecert: the code certificate

Open a command window and navigate to the \SecurityKeyManagementTool\cli folder and run the following command to generate the code certificate based on the project created in section 3.1.

```
C:\Renesas\SecurityKeyManagementTool\cli>skmt.exe /gencert /mode "CRC" /loadaddr "FFF00000"  
/cfsize "100000" /oembl  
"C:\PARENT_LOC\blinkycrcboot_rx74m_ccrx\Debug\blinkycrcboot_rx74m_ccrx.mot"  
/output_codecert "C:\RX74M_FSBL\certificates\blinkycrcboot_rx74m_code_cert.bin"  
Output File: C:\RX74M_FSBL\certificates\blinkycrcboot_rx74m_code_cert.bin
```

Creating the Code Certificate for a CRC Boot by Using the SKMT GUI

Open GUI and navigate to the **FSBL** tab. Select **CRC** for **Programming Verification Method** and set the output file for **Code Certificate**. Clicking **Generate File(s)** generates a code certificate.

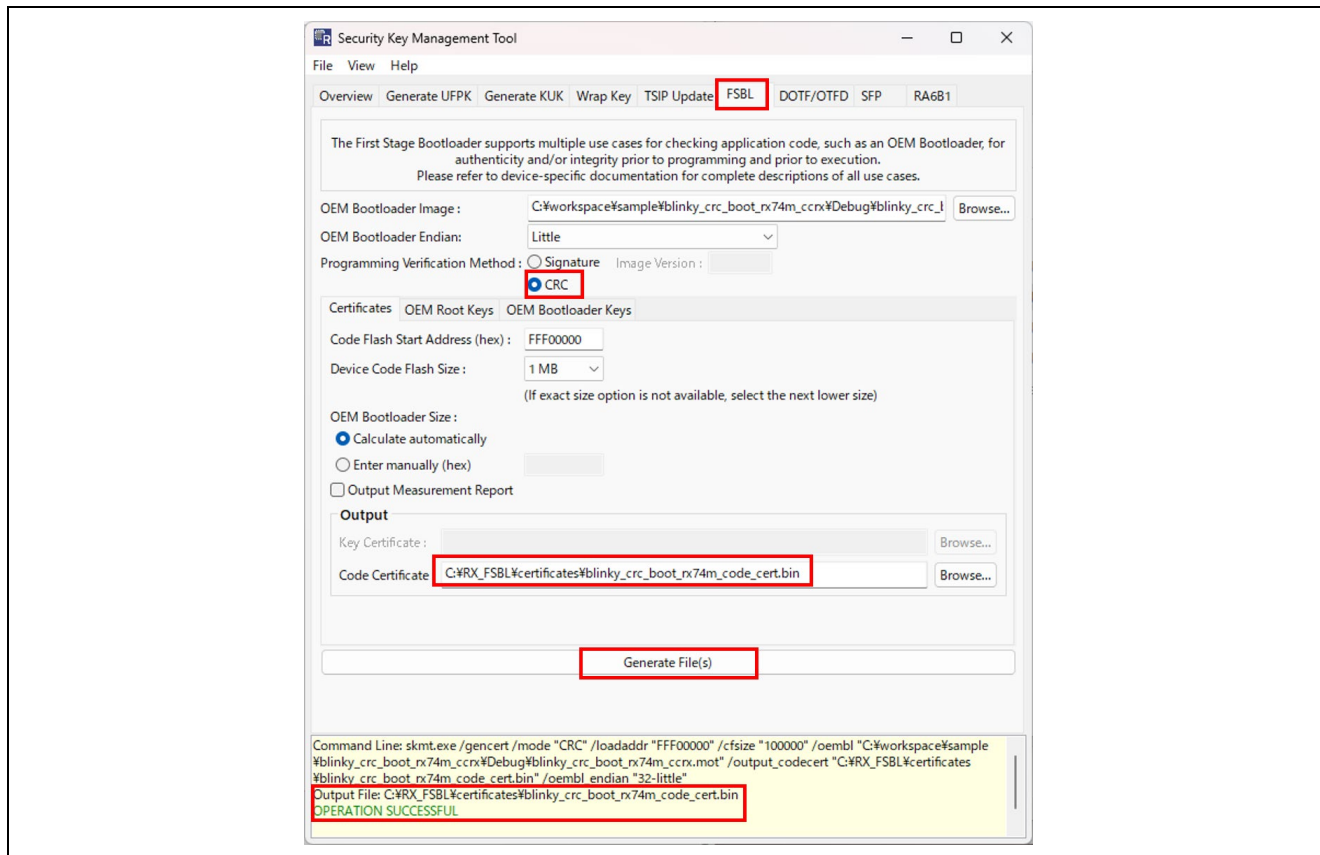


Figure 3-4 Create the Code Certificate for a CRC Boot

When the code certificate is generated by specifying the CRC, the CRC value is output as a dummy value for the Signer ID field in the code certificate. When a measurement report for CEC verification is created, the Signer ID is used eight times the length of the CRC value. CRC32 is 4 bytes (32 bits) and Signer ID (SHA256) is 32 bytes, so the CRC value is written eight times in the Signer ID field.

3.3 Using the RFP for Programming and Checking Operations

Launch the **RFP** and open the included `rfp_rx74m.rpj`. Provide the authentication information to download the application to the MCU and authenticate and boot the application through the MCU's FINE interface boot mode.

Ensure that the following **Operation Settings** are configured.

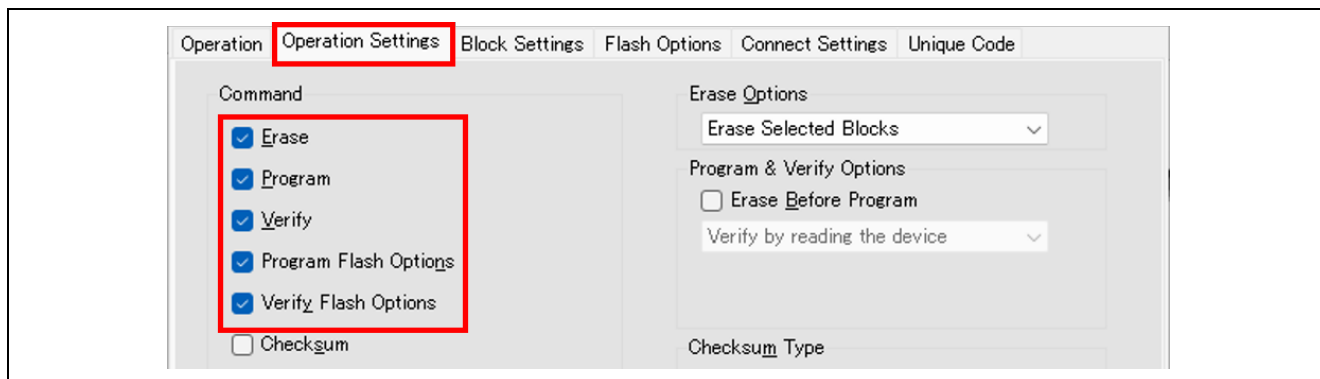


Figure 3-5 Configure the Operation Settings before Downloading the Application

On the **Operation** tab, select the application project to be programmed into the MCU.

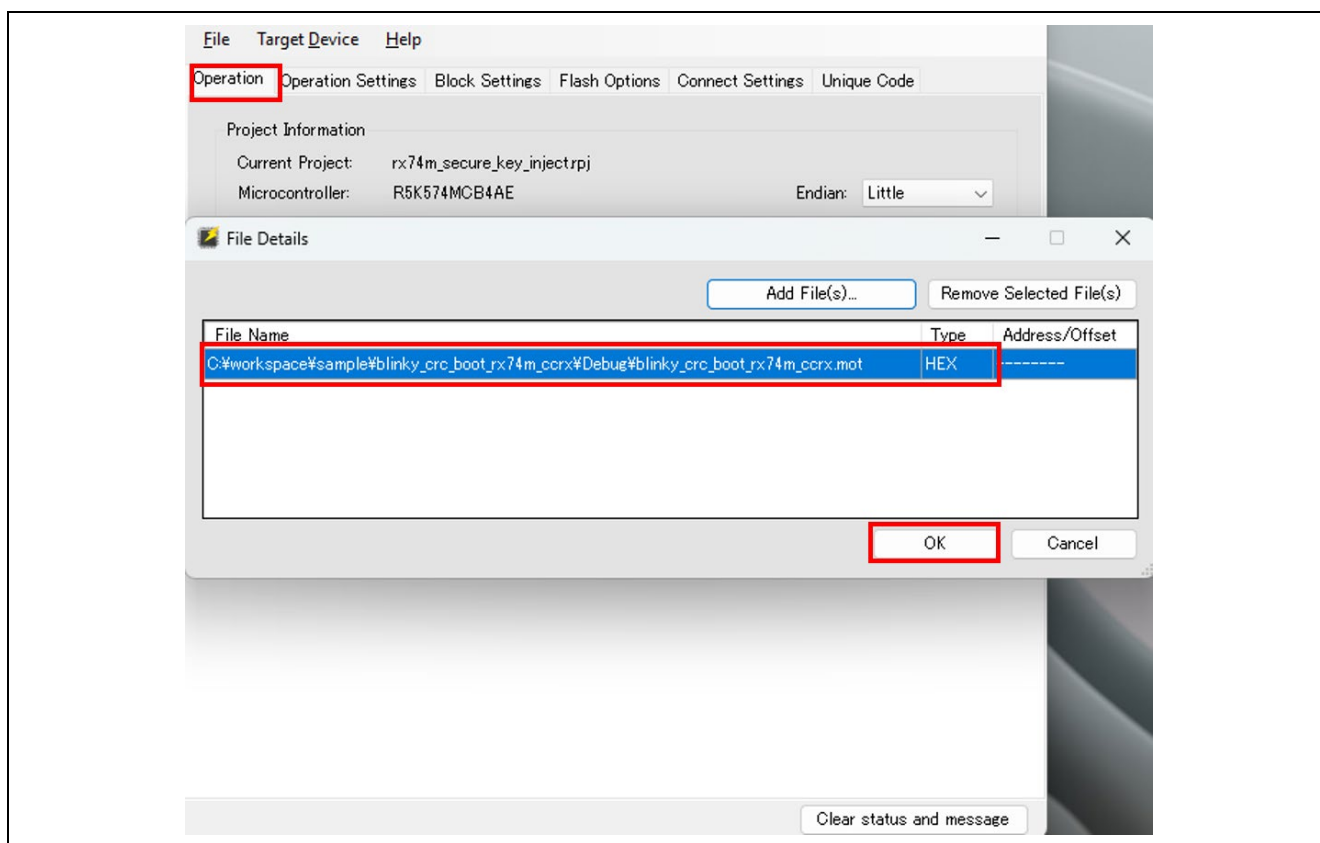


Figure 3-6 Select the Application Project

Click **OK** and navigate to the **Flash Options** tab to configure the items for **Certificate**.

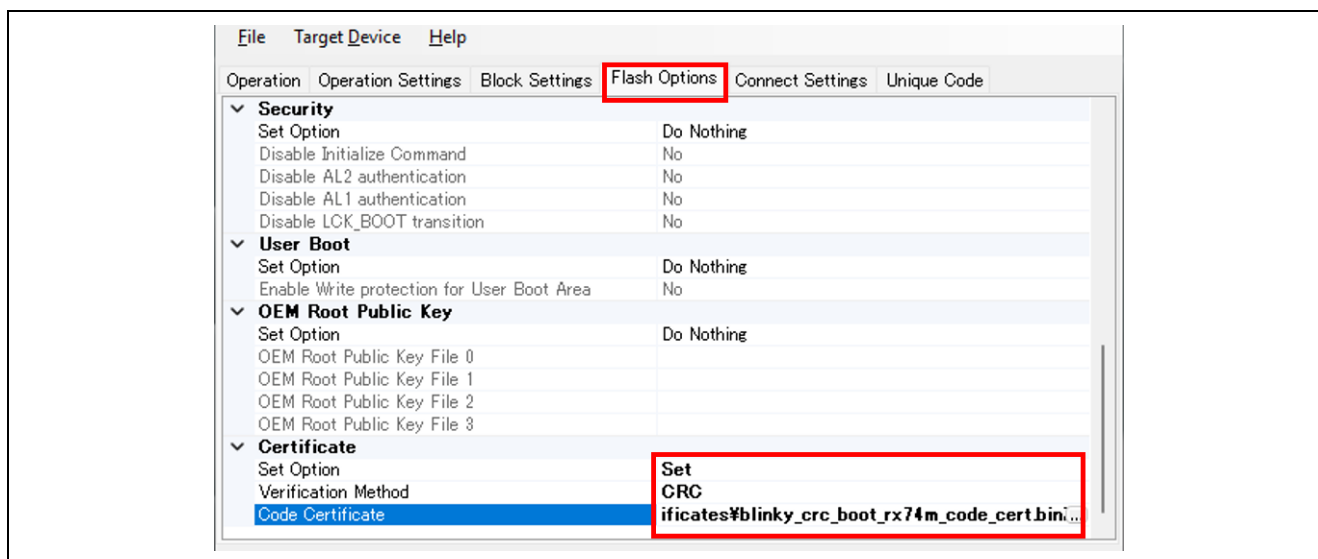


Figure 3-7 Configure the CRC Code Certificate

Next, navigate to the **Operation** tab and select **Start**. The application as well as the code certificate will be programmed. The three LEDs will blink, indicating normal operation.

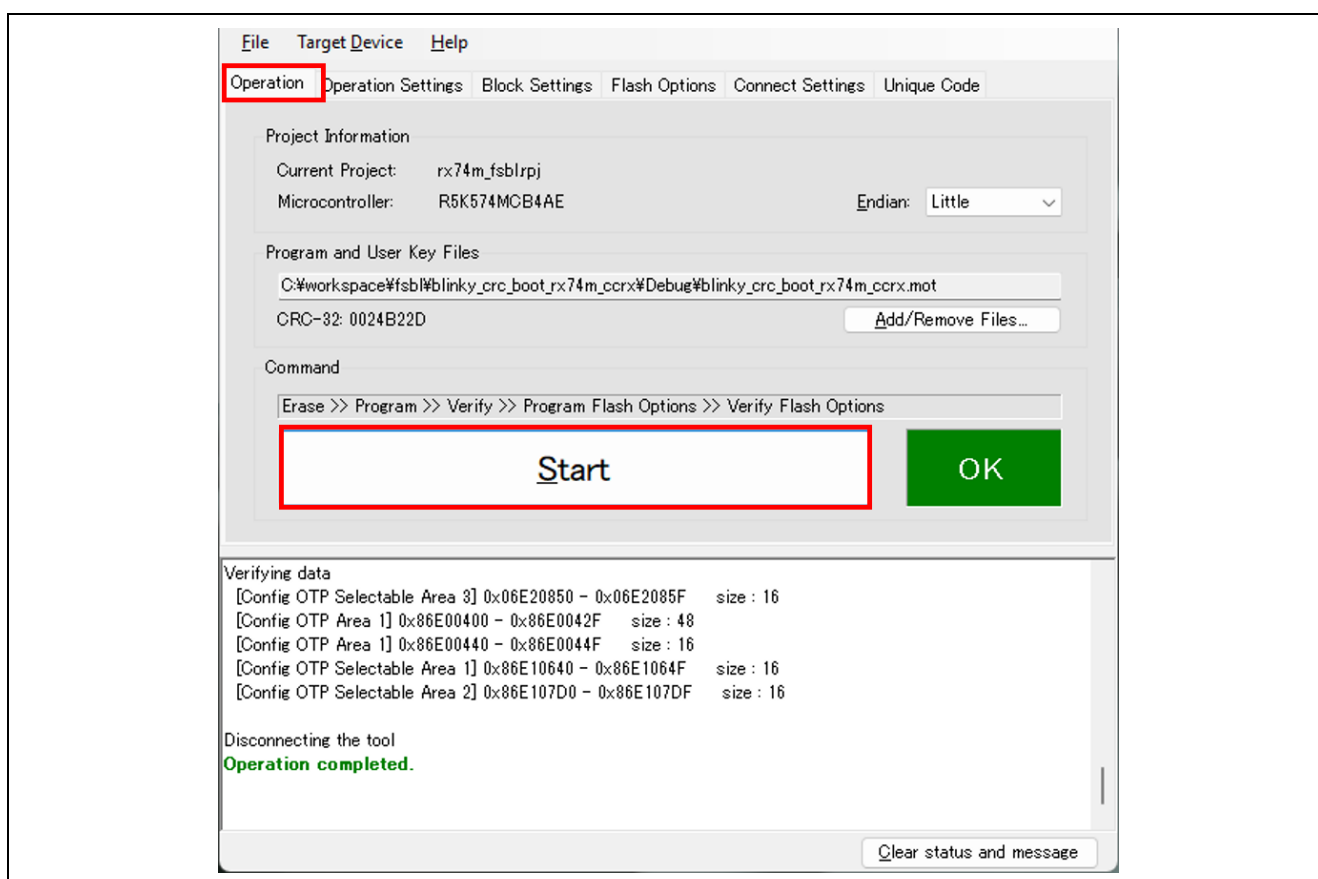
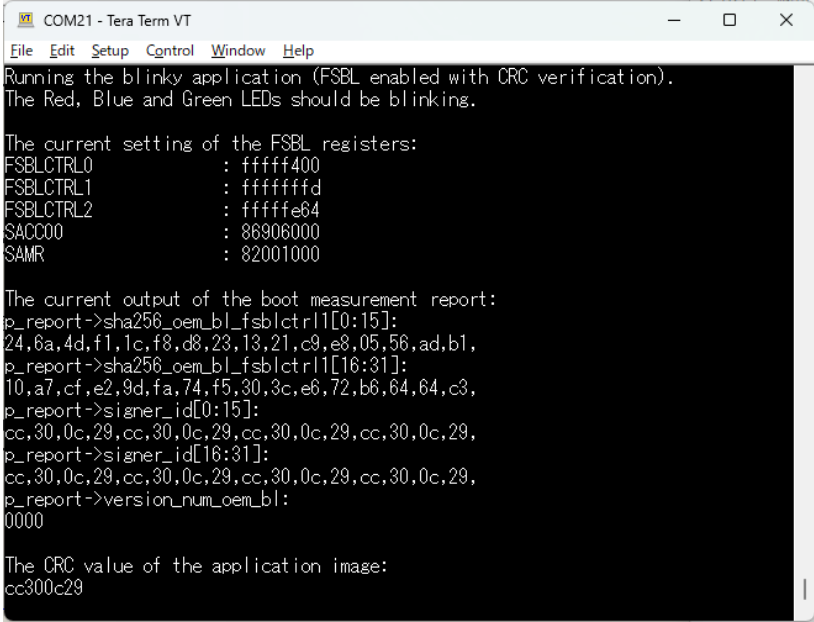


Figure 3-8 Download the CRC Boot Application

On the Tera Term console, a message similar to the following will be displayed. Note that the CRC value is repeated eight times as `signer_id`.



```

COM21 - Tera Term VT
File Edit Setup Control Window Help
Running the blinky application (FSBL enabled with CRC verification).
The Red, Blue and Green LEDs should be blinking.

The current setting of the FSBL registers:
FSBLCTRL0      : fffff400
FSBLCTRL1      : ffffffff
FSBLCTRL2      : fffffe64
SACCC00        : 86906000
SAMR           : 82001000

The current output of the boot measurement report:
p_report->sha256_oem_bl_fsblctrl1[0:15]:
24,6a,4d,f1,1c,f8,d8,23,13,21,c9,e8,05,56,ad,b1,
p_report->sha256_oem_bl_fsblctrl1[16:31]:
10,a7,cf,e2,9d,fa,74,f5,30,3c,e6,72,b6,64,64,c3,
p_report->signer_id[0:15]:
cc,30,0c,29,cc,30,0c,29,cc,30,0c,29,cc,30,0c,29,
p_report->signer_id[16:31]:
cc,30,0c,29,cc,30,0c,29,cc,30,0c,29,cc,30,0c,29,
p_report->version_num_oem_bl:
0000

The CRC value of the application image:
cc300c29
  
```

Figure 3-9 Console Output for a CRC Boot Using the Certificate

The CRC32 value calculated by the RFP does not match the CRC32 value calculated by the SKMT. The RFP calculates the CRC32 of the entire file. For example, for a .sec file, all the record fields are included in the CRC32 calculation. On the other hand, the CRC value in the code certificate calculates the CRC32 only for the binary data of the OEM_BL.

To check a failure in a CRC boot, initialize the device by using the initialization function of the RFP and perform the steps in section 3.3. The CRC boot will fail and the signal on pin P604 will go to the high level, turning on the red LED.

4. Sample Project for an HMAC Boot

This section explains the usage of the sample project for an HMAC boot.

The procedure for implementing the root of trust (SHA256 hash of the OEM_ROOT_PK) has a lot in common with the secure key injection process described in the application note R01AN8067. Refer to the relevant sections in R11AN0496 as required.

The sample project of this application note uses the NIST CAVP test vector values as an example. To check the original test vector values, download from the link below.

[Cryptographic Algorithm Validation Program | CSRC \(nist.gov\)](https://csrc.nist.gov/Cryptographic-Algorithm-Validation-Program)

Though this section presents the case where the compiler in use is CC-RX, the details of the operations are the same for the GCC compiler.

4.1 Preparing Keys

As explained in section 2.2.1, two sets of ECC secp256r1 key pairs are required to build a chain of trust for the system. This application note explains how to generate the key pairs by using OpenSSL and how to use the key pairs of the NIST CAVP test vectors.

4.1.1 Generate ECC Key Pairs by Using OpenSSL

When the operating system of your evaluation PC is Linux, get OpenSSL from the link below:

<https://www.openssl.org/source/>

For Windows, get OpenSSL from the link below:

<https://slproweb.com/products/Win32OpenSSL.html>

After downloading OpenSSL, install it and navigate to a desired folder from a command prompt to use the OpenSSL commands.

The following command generates an OEM_BL verification root key pair.

```
openssl ecparam -name prime256v1 -genkey -noout -out oem_root_private_key.pem
```

The following command generates a public key for the OEM_BL verification root key pair (OEM_ROOT_PK).

```
openssl ec -in c:\RX74M_FSBL\openssl_keys\oem_root_private_key.pem -pubout -out oem_root_public_key.pem
```

The same command can be used to generate an OEM_BL verification key pair.

```
openssl ecparam -name prime256v1 -genkey -noout -out oem_bl_private_key.pem
```

The following command generates a public key for the OEM_BL verification key pair (OEM_BL_PK).

```
openssl ec -in c:\RX74M_FSBL\openssl_keys\oem_bl_private_key.pem -pubout -out oem_bl_public_key.pem
```

4.1.2 NIST CAVP Test Vectors as ECC Key Pairs

The sample project of this application note uses the NIST CAVP test vector values as an example. To check the original test vector values, download from the link below.

[Cryptographic Algorithm Validation Program | CSRC \(nist.gov\)](https://csrc.nist.gov/Cryptographic-Algorithm-Validation-Program)

The following key pairs are used here.

ECC P-256 (secp256r1) Key Pair for the OEM_BL Verification Root Key Pair

The following key pair is used for the OEM_BL verification root key pair. It is used to generate the wrapped OEM_ROOT_PK in section 4.2. It is also used for the generation of the key certificate in section 4.5.

```
d = c9806898a0334916c860748880a541f093b579a9b1f32934d86c363c39800357
Qx = d0720dc691aa80096ba32fed1cb97c2b620690d06de0317b8618d5ce65eb728f
Qy = 9681b517b1cda17d0d83d335d9c4a8a9a9b0b1b3c7106d8f3c72bc5093dc275f
```

ECC P-256 (secp256r1) Key Pair for the OEM_BL Verification Key Pair

The following key pair is used for the OEM_BL verification key pair. It is used to generate the code certificate in section 4.5.

```
d = 710735c8388f48c684a97bd66751cc5f5a122d6b9a96a2dbe73662f78217446d
Qx = f6836a8add91cb182d8d258dda6680690eb724a66dc3bb60d2322565c39e4ab9
Qy = 1f837aa32864870cb8e8d0ac2ff31f824e7beddc4bb7ad72c173ad974b289dc2
```

4.2 Implementing the Root of Trust

As explained earlier, the hash of the OEM_ROOT_PK is the root of trust of the system. When the FSBL is enabled, this hash will be injected on successful programming of the application.

Before proceeding with the operation in this section, go through the process of wrapping the User Factory Programming Key (UFPK) using the Renesas Key Wrap Service. For the security considerations and the process of the UFPK wrapping, refer to the following sections in the application note R01AN8067 to learn the usage background and establish PGP encrypted communication with the Renesas Key Wrap Server.

- Section “Create PGP Key Pair” for the generation of customer PGP key pair
- Section “Registration with Key Wrap Server” for the registration with the Key Wrap Server
- Section “Exchange User and Renesas PGP Public Keys” for the establishment of encrypted communication with the Renesas Key Wrap Server

4.2.1 Generate the Wrapped User Factory Programming Key (W-UFPK)

Safely injecting the OEM_ROOT_PK into the RX74M requires the use of a UFPK to wrap the OEM_ROOT_PK. For how to generate a UFPK and W-UFPK, refer to “Generate a UFPK and Obtaining a W-UFPK” in the application note R01AN8067. To generate a UFPK and W-UFPK for key injection of the OEM_ROOT_PK into the RX74M, apply the following settings in the procedure described in R01AN8067.

1. On the **Overview** screen of SKMT, select **RX Family, RSIP-E50D Security Functions and Protected Mode**.

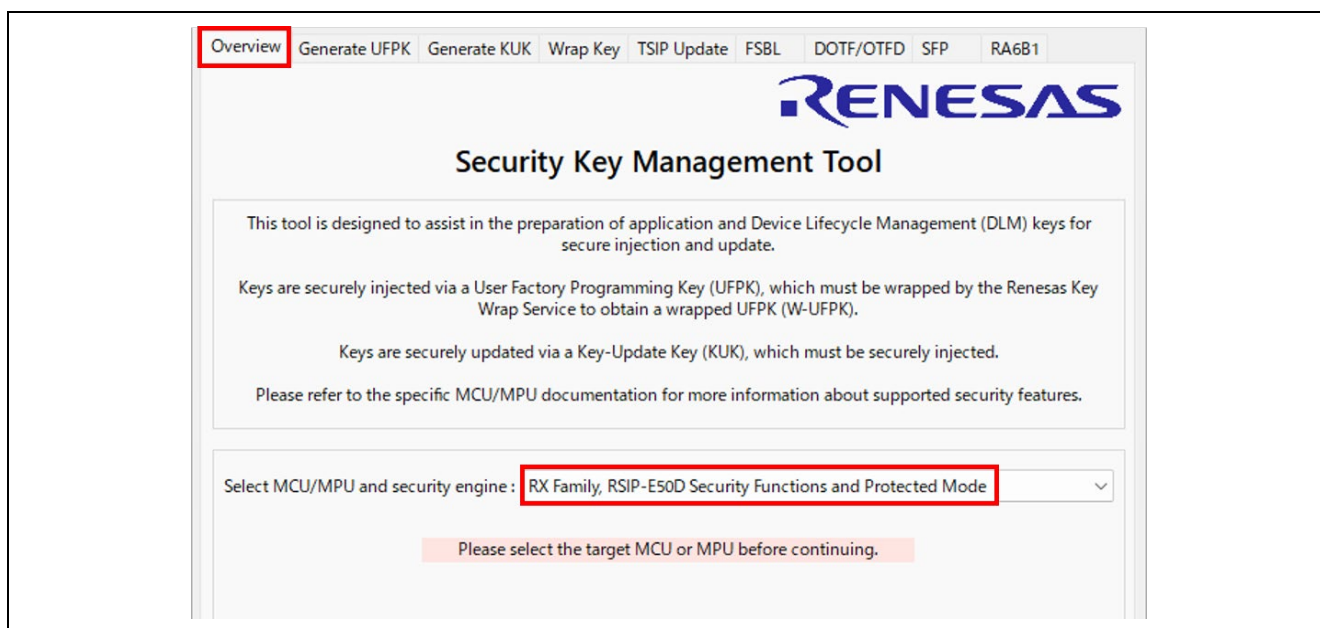


Figure 4-1 Select RSIP-E51A Security Functions and Protected Mode

2. When using the Renesas Key Wrap Service, select the item for RX74M under **RSIP Protected Mode** rather than **RSIP Compatibility Mode**.

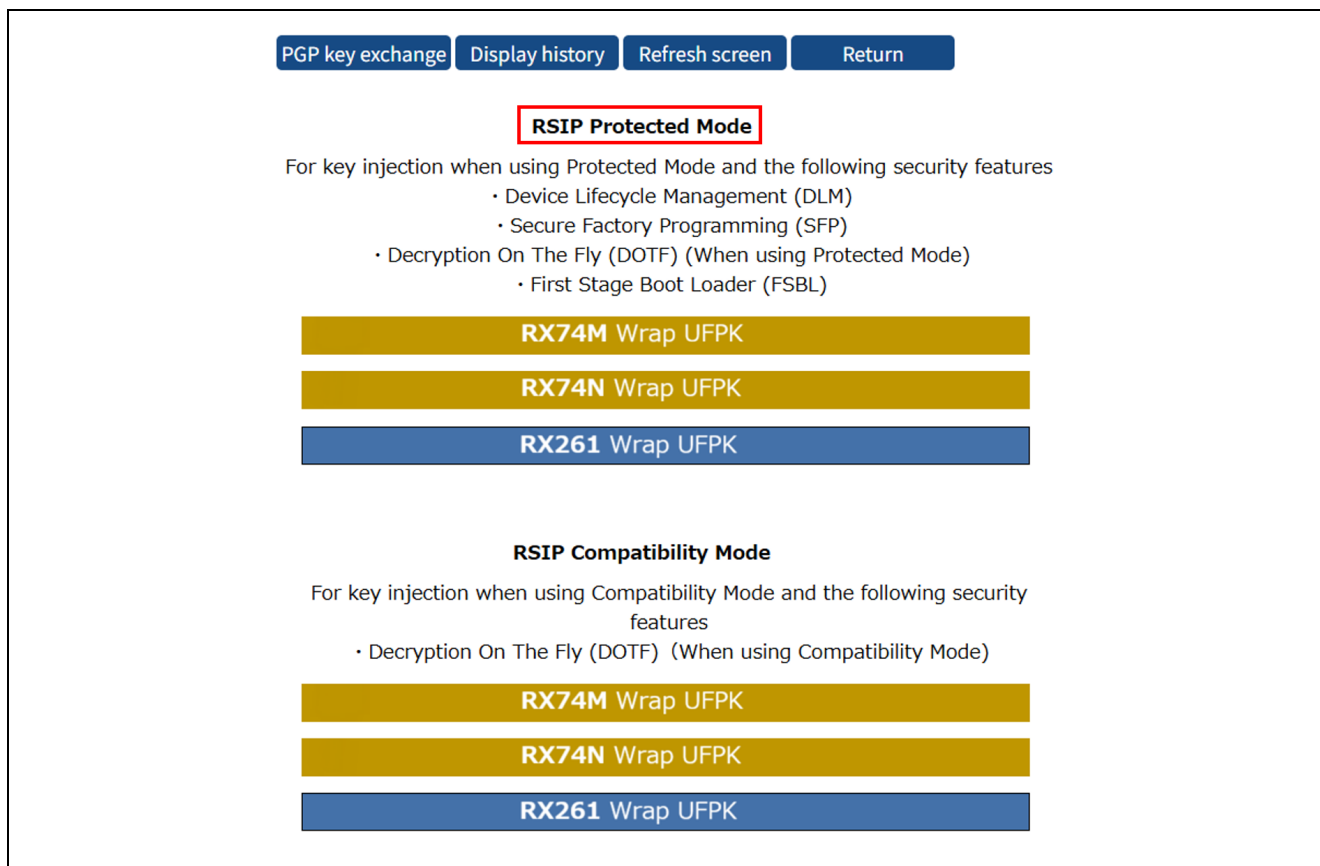


Figure 4-2 Select the RX74M Group for RSIP Protected Mode

When using the SKMT and Key Wrap Service, the same security mode must be selected. A UFPK and W-UFPK that have been generated in a different mode cannot be used.

For the rest of the operations, follow the procedure described in R01AN8067.

The following explanation assumes that the next file is stored in the C:\RX74M_FSBL\ufpk folder.

- UFPK: rx74m_ufpk.key
- W-UFPK: rx74m_ufpk.key_enc.key

In sections 4.2.2 and 4.2.3, these files are used to wrap the OEM_ROOT_PK.

4.2.2 Generate the Wrapped OEM_ROOT_PK by Using the SKMT CLI

Once the SKMT is installed, open a command line window and navigate to the \cli folder, and use the following command to generate the wrapped OEM_ROOT_PK. This key is injected into the MCU by using the secure key injection process with the use of the RFP.

```
C:\Renesas\SecurityKeyManagementTool\cli>skmt.exe /genkey /ufpk
file="C:\RX74M_FSBL\ufpk\rx74m_ufpk.key" /wufpk
file="C:\RX74M_FSBL\ufpk\rx74m_ufpk.key_enc.key" /mcu "RX-RSIP-E50D" /keytype
"OEM_ROOT_PK" /filetype "rfp" /iv "55AA55AA55AA55AA55AA55AA55AA55AA55AA" /key
file="C:\RX74M_FSBL\openssl_keys\oem_root_private_key.pem" /output
"C:\RX74M_FSBL\openssl_keys\oem_root_pk_cli.rkey"
```

Output File: C:\RX74M_FSBL\openssl_keys\oem_root_pk_cli.rkey

UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F

W-UFPK: 00000000F77A67AB713D174EB9D7C6639DAADB1F77A67AB713D174EB9D7C6639DAADB1

IV: 55AA55AA55AA55AA55AA55AA55AA55AA55AA

Encrypted key:

F714A8B68CB80637F9EE1F35A310EBC262DE6D4311E1E56B762ECD4BA709BD94827A9B7906628B71
1E3CC8CCBA68BA8CE8B3392DEF2591574D647219F8F0DC11599EA9637BC7A94A53B260B5C248AB2C

Note that this example oem_root_pk_cli.rkey does not match the sample project in this application note. Therefore, do not use this key in the sample project.

4.2.3 Generate the Wrapped OEM_ROOT_PK by Using the SKMT GUI

In this section, the OEM_BL verification root key pair stated in section 4.1.2 is used to generate the wrapped OEM_ROOT_PK.

Launch the **SKMT**, and on the **Overview** screen, select **RX Family**, **RSIP-E50D Security Functions and Protected Mode** (see Figure 4-1). Then, navigate to the **Wrap Key** tab in the SKMT, select **OEM Root public** for **Key Type**, and input the Wrapping Key (UFPK and W-UFPK) (see Figure 4-3).

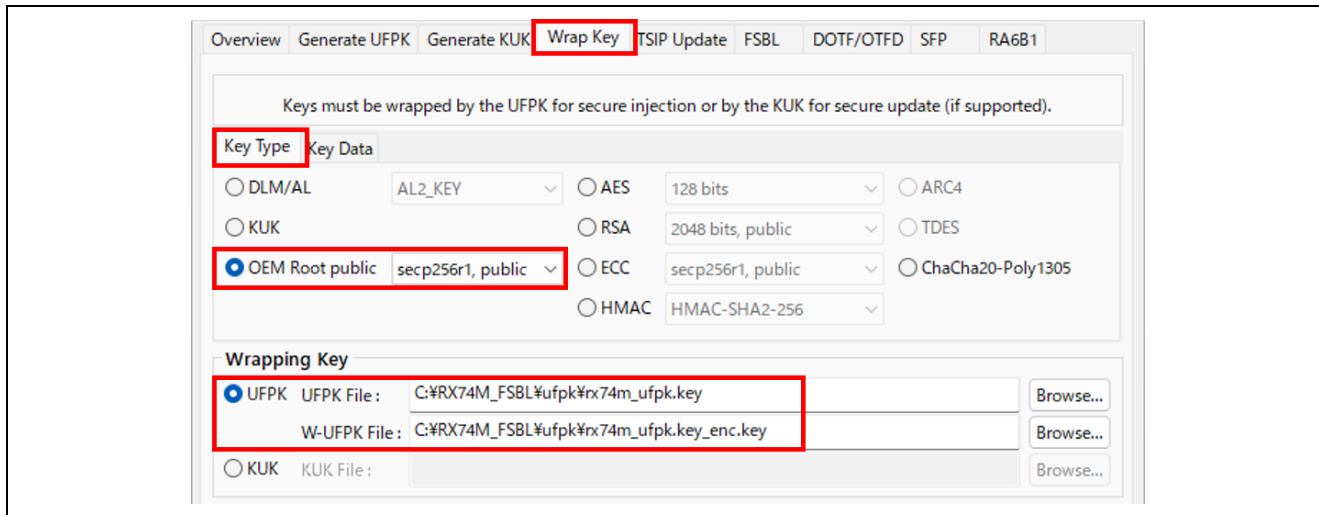


Figure 4-3 Wrap the OEM_ROOT_PK

On the **Key Data** tab, input the OEM Root Public Key.

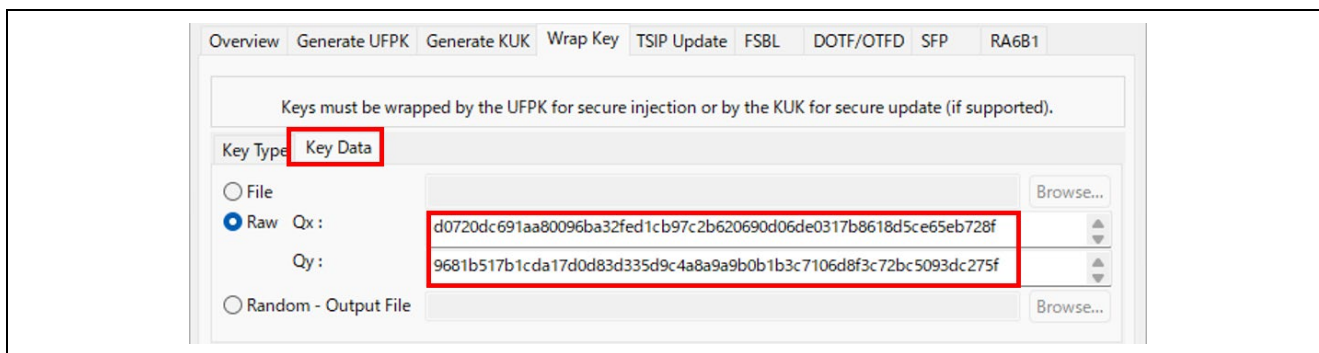


Figure 4-4 Specify the OEM_ROOT_PK

Now select **RFP** as the output type, select the output location, name the file, and then click **Generate File** to generate the wrapped OEM_ROOT_PK.

Security Key Management Tool

File View Help

Overview Generate UFPK Generate KUK **Wrap Key** TSIP Update FSBL DOTF/OTFD SFP RA6B1

Keys must be wrapped by the UFPK for secure injection or by the KUK for secure update (if supported).

Key Type Key Data

☐ File

☒ Raw Qx : d0720dc691aa80096ba32fed1cb97c2b620690d06de0317b8618d5ce65eb728f
Qy : 9681b517b1cda17d0d83d335d9c4a8a9a9b0b1b3c7106d8f3c72bc5093dc275f

☐ Random - Output File

Wrapping Key

☒ UFPK UFPK File : C:\RX74M_FSBL\ufpk\rx74m_ufpk.key
W-UFPK File : C:\RX74M_FSBL\ufpk\rx74m_ufpk.key_enc.key

☐ KUK KUK File :

IV

☒ Generate random value
☐ Use specified value (16 hex bytes, big endian format) 00112233445566778899AABBCCDDEEFF

Output

Format: **RFP** File: C:\RX74M_FSBL\key\oem_root_key.rkey
Endian: Little ☐ Output additional data
Address: 10000 Key name:

Command Line: skmt.exe /genkey /ufpk "C:\RX74M_FSBL\ufpk\rx74m_ufpk.key" /wufpk "C:\RX74M_FSBL\ufpk\rx74m_ufpk.key_enc.key" /mcu "RX-RSIP-E50D" /keytype "OEM_ROOT_PK" /key "d0720dc691aa80096ba32fed1cb97c2b620690d06de0317b8618d5ce65eb728f9681b517b1cda17d0d83d335d9c4a8a9a9b0b1b3c7106d8f3c72bc5093dc275f" /filetype "rfo" /output "C:\RX74M_FSBL\key\oem_root_key.rkey"

Output File: C:\RX74M_FSBL\key\oem_root_key.rkey

UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F
W-UFPK: 00000000F77A67AB713D174EB9D7C6639DAADB8D1F77A67AB713D174EB9D7C6639DAADB8D1
IV: 1B61147FC862585F1A548D8B11612945

Encrypted key:
2C9917193E9263EEAD7608F6EA68E7878B95EA7224D4465974D4A681151B48611F5206909F0F2905922BA14DE817D005488F2
43E316E7C63BD3FF4189FF905B0505F22585D6B555DD9CC87F51056F916

OPERATION SUCCESSFUL

Figure 4-5 Generate the Wrapped OEM_ROOT_PK

This wrapped OEM_ROOT_PK (oem_root_key.rkey) is provided to you to enable convenient evaluation of the system. It can be used if the same public key is used in the key and code certificate generation.

4.3 HMAC Boot Settings for the RX74M FSBL

Typically, to use FSBL enabled boot, the OEM_BL should be developed first with FSBL disabled. After the OEM_BL is fully tested, the FSBL can be enabled.

The included blinky sample project `blinky_hmac_boot_rx74m_ccrx` is created with no RTOS support for RX74M as described in the “Create a New Project for Blinky” section in RX Flexible Software Package Documentation. By default, setting up the Protect Zone boundary is enabled in the Debug configuration setting.

The FSBL property is configured based on the configuration in Figure 4-6. For the convenience of debugging the application, FSBL skip is enabled upon software reset or exit from deep software standby mode. In addition, the project uses pin P604 as an error report pin, which is set to the low level in the initial state and can be set to the high level by the FSBL upon a boot failure. Note that although the e² studio sample project has the FSBL enabled, the FSBL will not be enabled after the application image is downloaded by e² studio. The FSBL will only be effective after the next power cycle.

To use the E2 Emulator on Board of the EK-RX74M, set pins 1 and 2 of SW6 to OFF, and connect J8 and the development PC with a USB Type-C® cable. When using the RDPM, any among the SCI, FINE, and SWD interfaces is selectable, but use FINE for the connection in this case.

Follow the steps below to build and run the included sample project.

Import the blinky project (blinky_hmac_boot_rx74m_ccrx) to the workspace, open the configuration.xml file, and click Generate Project Content.

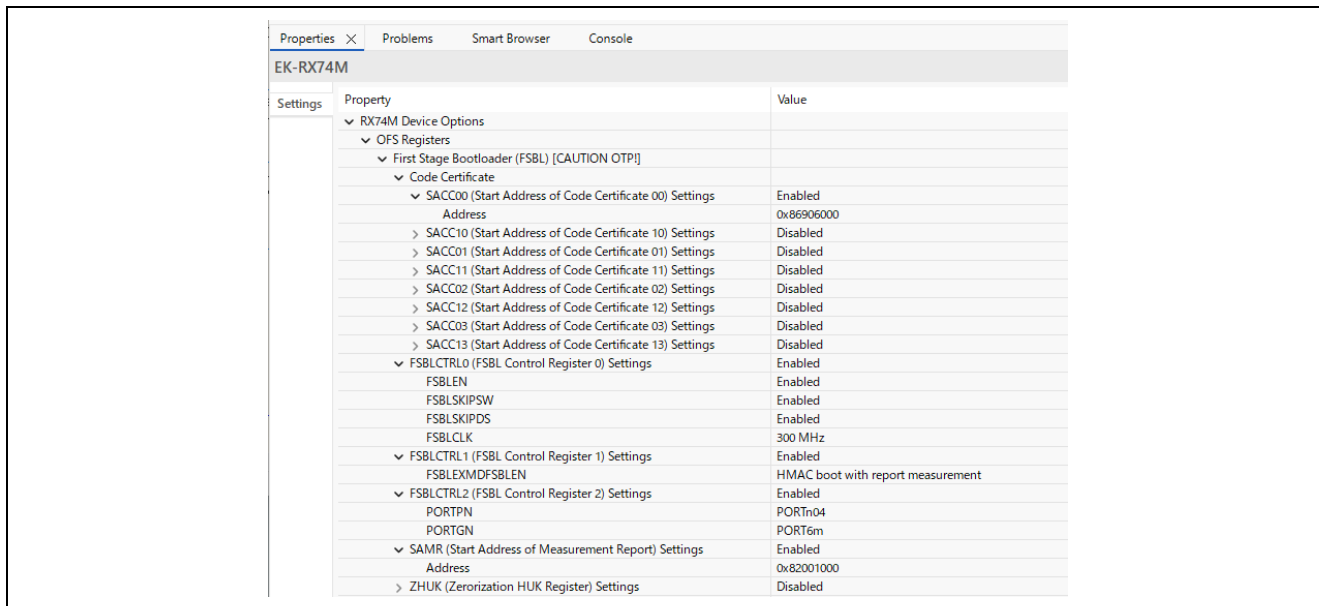


Figure 4-6 FSBL Settings for an HMAC Boot

Then, connect the E2 emulator On Board on J8 to the USB of the evaluation board with a USB Type-C® cable.

Build and run the blinky project (blinky_hmac_boot_rx74m_ccrx), and verify its functionality.

- The compilation result will be used for the creation of the code certificate in the section 4.5.
- The three LEDs will blink.

Open Tera Term and set **Speed** to **115200** and **Data** to **8 bit**.

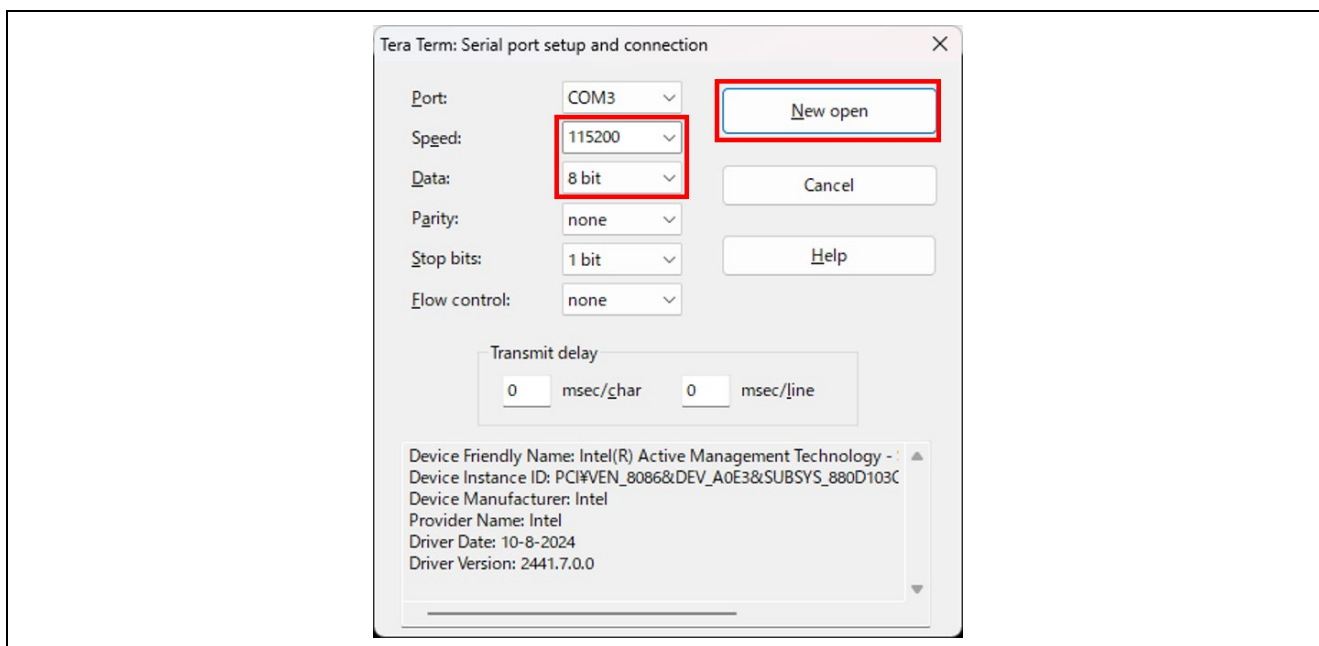
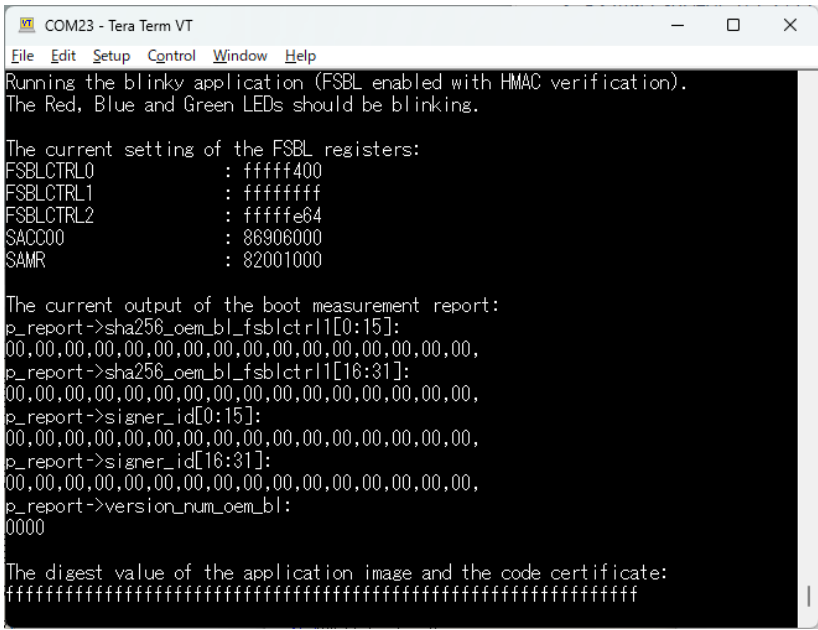


Figure 4-7 Configure Tera Term

Click **OK** and observe the following output on the console. Notice the output for the FSBL register settings, the boot measurement report, and the HMAC digest. For the content of the output, refer to the content of the “Secure Boot” section in the *RX74M Group User’s Manual: Hardware* (Figure 2-4). Since the FSBL is not enabled yet, the version_num_oem_bl field is 0s and the digest area is undefined.



```

COM23 - Tera Term VT
File Edit Setup Control Window Help
Running the blinky application (FSBL enabled with HMAC verification).
The Red, Blue and Green LEDs should be blinking.

The current setting of the FSBL registers:
FSBLCTRL0      : fffff400
FSBLCTRL1      : ffffffff
FSBLCTRL2      : fffffe64
SACCD0         : 86906000
SAMR           : 82001000

The current output of the boot measurement report:
p_report->sha256_oem_bl_fsblctrl1[0:15]:
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
p_report->sha256_oem_bl_fsblctrl1[16:31]:
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
p_report->signer_id[0:15]:
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
p_report->signer_id[16:31]:
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
p_report->version_num_oem_bl:
0000

The digest value of the application image and the code certificate:
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff

```

Figure 4-8 Console Output for an HMAC Boot

At this time, the root of trust and certificates have not been programmed in the MCU. Therefore, if the FSBL is executed after a power-on reset, it will fail to authenticate the OEM_BL, and the application will not be started up. Starting up the application normally requires injection of the root of trust and authenticated production programming. The details are explained in section 4.6.

4.4 Anti-Rollback Counters

The RX74M has anti-rollback counters to prevent write-back to an old version of the OEM_BL. When an HMAC boot is used, the boot firmware compares the image version number in the code certificate and the anti-rollback counter when writing the image, to check that rollback does not apply to the image to be written.

The anti-rollback counter to be used is selected by the ARC usage mode select bits (ARCMD) of the anti-rollback counter setting register (ARCCS). The target of verification for an HMAC boot can be set to either OEM_BL or secure application by the setting of ARCMD.

Table 4-1 ARCMD Setting and Counter to be Used

ARCMD Setting	Mode	Verification Target	Counter to be Used	Version Setting Range
01b	Mode 1	SSBL	ARCOEMBL	1 to 64
00b	Mode 2	Application	ARCS	1 to 1024

7.2.33 Anti-Rollback Counter Setting Register (ARCCS)

Address(es): OFSM.ARCCS 86E3 0E30h

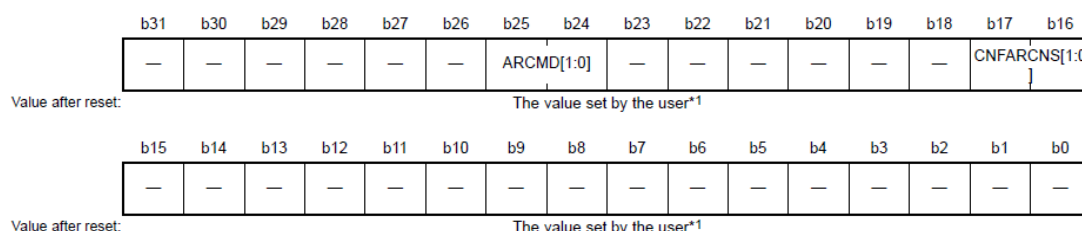


Figure 4-9 Anti-Rollback Counter Setting Register

When ARCMD[1:0] are set to 01b (mode 1), the target of verification for an HMAC boot is an SSBL. In this mode, the image version number in the code certificate and the current value of the anti-rollback counter for OEM boot loader (ARCOEMBL) are compared.

The SSBL can be programmed as long as the image version number in the code certificate is greater than or equal to the current value of ARCOEMBL. If the verification of the SSBL was successful and the image version number in the code certificate is greater than the current value of ARCOEMBL, the boot firmware updates ARCOEMBL to the image version number from the certificate.

7.2.37 Anti-Rollback Counter for OEM Boot Loader [i] (ARCOEMBL[i]) (i = 0, 1)

Address(es): OFSM.ARCOEMBL[0] 86F4 1F00h, OFSM.ARCOEMBL[1] 86F4 1F04h

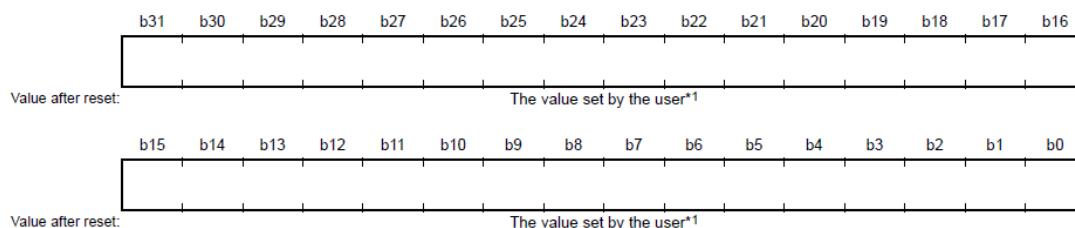


Figure 4-10 Anti-Rollback Counter for OEM Boot Loader

When ARCMD[1:0] are set to 00b (mode 2), the target of verification for an HMAC boot is an application. In this mode, the image version number in the code certificate and the current value of the anti-rollback counter for secure application (ARCS) are compared.

The secure application can be programmed as long as the image version number in the code certificate is greater than or equal to the current value of ARCS. If the verification of the secure application was successful and the image version number in the code certificate is greater than the current value of ARCS, the boot firmware updates ARCS to the image version number from the certificate.

7.2.34 Anti-Rollback Counter for Secure Application [i] (ARCS[i]) (i = 0 to 31)

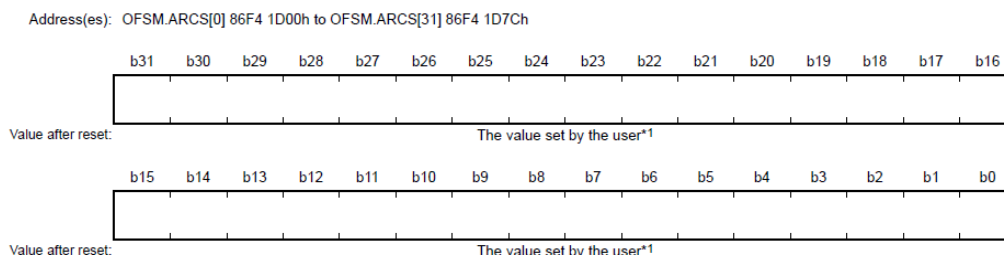


Figure 4-11 Anti-Rollback Counter for Secure Application

The anti-rollback counter records the image version number by setting the unused bits corresponding to the version number to 1 when updating the counter value. For example, when the first three bits are set, the register value becomes 0x00000007, which corresponds to image version 3.

When updating the image, set a value greater than the current version number recorded in the anti-rollback counter as the image version number of the code certificate.

Using an HMAC boot requires setting ARCMD. Select **Mode1** or **Mode2** for **Anti-Rollback Counter Mode** in the RFP, depending on the use case.

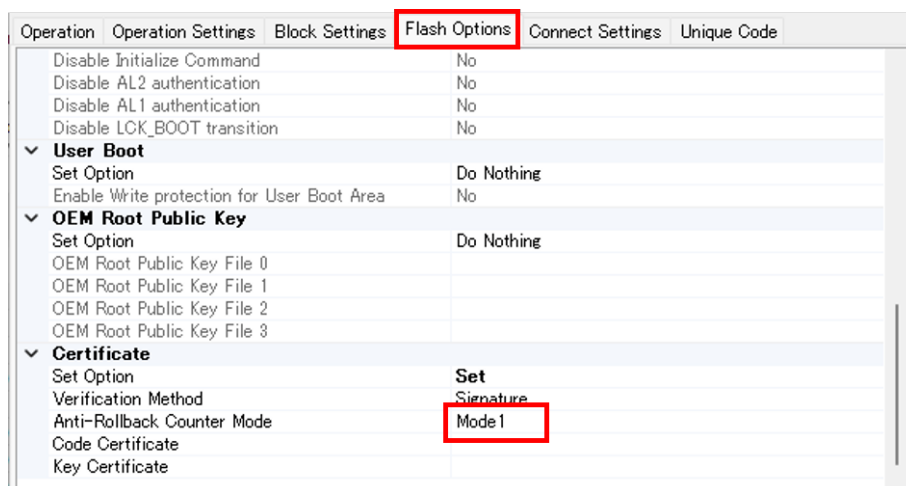


Figure 4-12 Set the Anti-Rollback Counter Usage Mode

For the configuration of the anti-rollback counter, how to read its value, and the condition for updating it, refer to the "Anti-Rollback Counter" section of the *RX74M Group User's Manual: Hardware*.

4.5 Generating Certificates

As explained earlier, for an HMAC boot, two sets of ECC P-256 (secp256r1) key pairs are required to generate the key and code certificates. In this section, two methods are demonstrated. The example keys should not be used for production support for security considerations.

- Using OpenSSL and SKMT command line interface
- Using NIST vectors and SKMT GUI

4.5.1 Using OpenSSL and SKMT Command Line Interface

To generate the key certificate and code certificate, there are several properties to configure.

- Input: /loadaddr: the load address of the application
- Input: /oembl_size: the size of the OEM_BL
- Input: /ver: the version of the application. Note that for an MCU that has previously booted from an old version of the application via the FSBL, the next version number must be a higher version number. The version number will be included in the code certificate generated (see Figure 2-4). In this example, it is set to 3, which can be used if the ARC_OEMBLn value is 0, 1, or 2.
- Input: /cfsize: the size of the code MRAM of the device to be used
- Input: /oembl: the binary file of the OEM_BL
- Input: /oemroot_private: OEM_ROOT_SK
- Input: /oembl_private: OEM_BL_SK
- Output: /output_codecert: the code certificate
- Output: /output_keycert: the key certificate

Open a command window and navigate to the \SecurityKeyManagementTool\cli folder, and run the following command to generate the key certificate and code certificate. Change the file locations as required.

```
C:\Renesas\SecurityKeyManagementTool\cli>skmt.exe /gencert /mode "signature" /loadaddr  
"02000000" /oembl_size "8000" /cfsize "100000" /ver "3" /oembl  
"C:\RX74M_FSBL\fsp_v510\blinky_hmac_boot_rx74m\Debug\blinky_hmac_boot_rx74m.srec  
" /oembl_private file="C:\RX74M_FSBL\openssl_keys\oem_bl_private_key.pem"  
/oemroot_private file="C:\RX74M_FSBL\openssl_keys\oem_root_private_key.pem"  
/output_codecert "C:\RX74M_FSBL\openssl_keys\blinky_hmac_openssl_ccert.bin"  
/output_keycert "C:\RX74M_FSBL\openssl_keys\blinky_hmac_openssl_kcert.bin"
```

Output File: C:\RX74M_FSBL\openssl_keys\blinky_hmac_openssl_kcert.bin

Output File: C:\RX74M_FSBL\openssl_keys\blinky_hmac_openssl_ccert.bin

4.5.2 Using NIST ECC Key Pairs and SKMT GUI

This section uses the two sets of NIST CAVP ECC P-256 (secp25r1) key pairs provided in section 4.1.2 to demonstrate the certificate generation. Both the private and public key parts of the two key pairs should be provided to the SKMT to generate the key certificate and code certificate.

Launch the SKMT GUI. On the **Overview** screen, select **RX Family, RSIP-E50D Security Functions and Protected Mode**.

Next, navigate to the **FSBL** tab and click the **Browse** button to the right of **OEM Bootloader Image**. Select the .srec file generated in section 4.3. Select **Signature** for **Programming Verification Method** and define the version number of the application for **Image Version**.

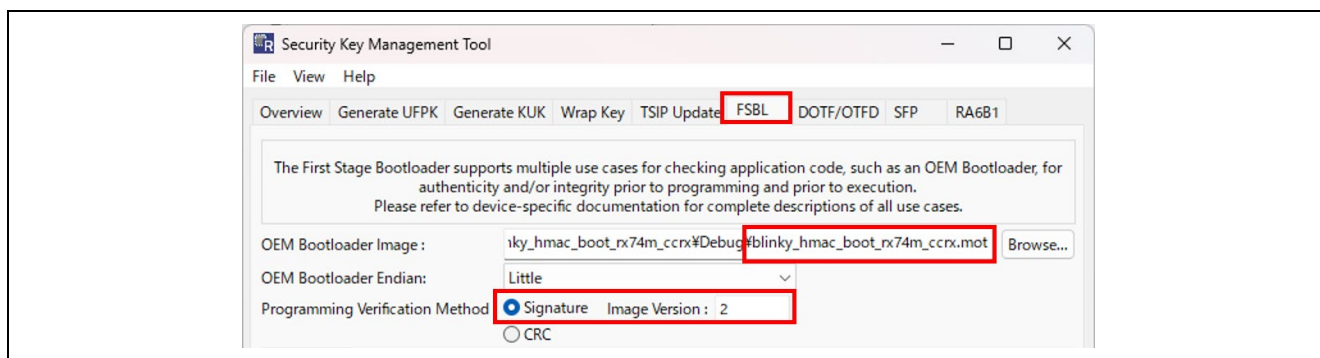


Figure 4-13 Select the Application and Version Number

The image version number defined in Figure 4-13 will be stored in the code certificate. As explained in section 2.2.3, the allowed version number depends on the target of FSBL verification. It is 1 to 64 when the target of FSBL verification is an SSBL and 1 to 1024 for an application. The image for use in updating must have a higher version number than that of the current programmed image. When an FSBL HMAC-based boot is executed on a new MCU for the first time, the image version number can be set to 1.

On the **FSBL** tab, set the **OEM Root Keys**. In the following example, select **Raw** to set the key values.

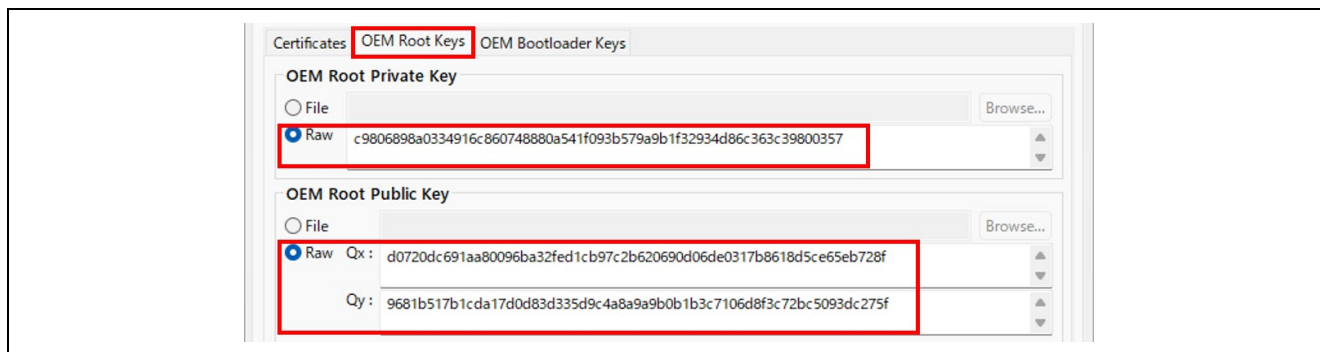


Figure 4-14 Specify the OEM_BL Verification Root Keys

Next, set the **OEM Bootloader Keys**. In the following example, select **Raw** to set the key values.

Figure 4-15 Specify the OEM_BL Verification Keys

Next, navigate to the **Certificates** tab and define the names and locations for **Key Certificate** and **Code Certificate**, and then click **Generate File(s)**. The key certificate and code certificate will then be generated. In the following example, select **Calculate automatically** for **OEM Bootloader Size**.

Figure 4-16 Generate the Key Certificate and Code Certificate

The SKMT can also internally generate ECC key pairs. This is achieved by selecting **Random – Output File**.

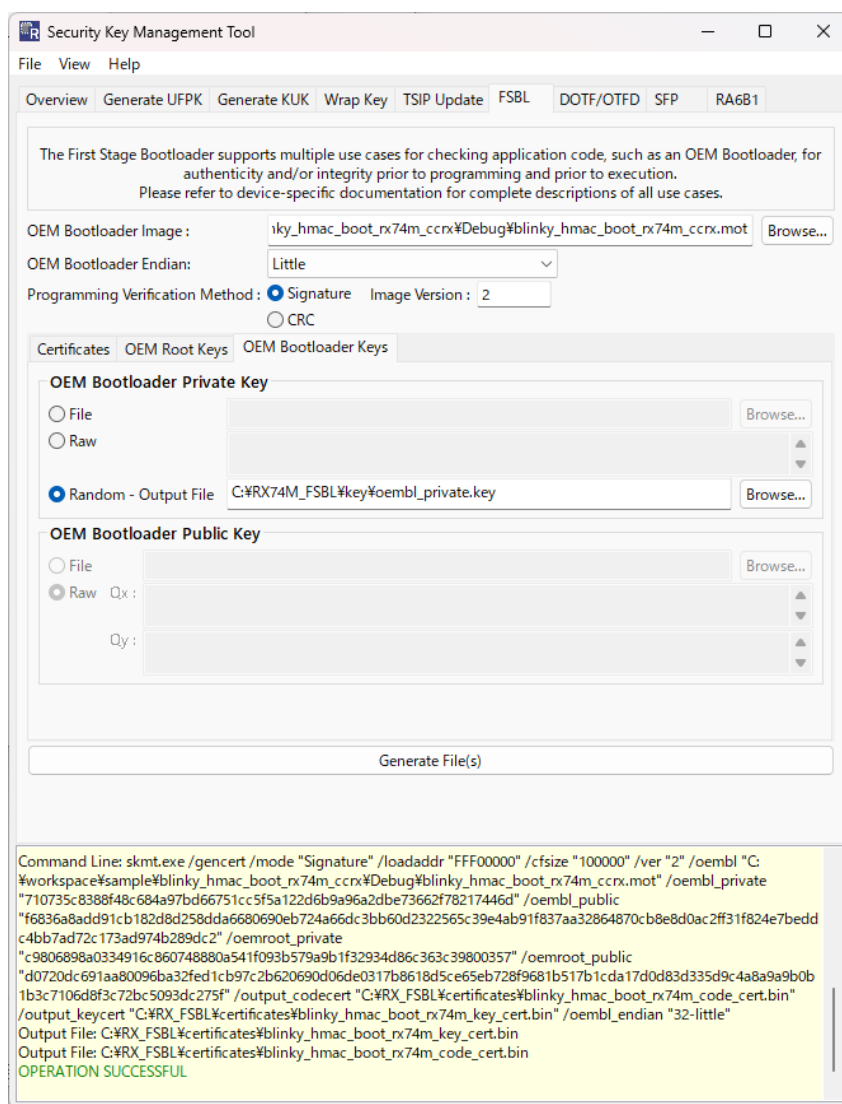


Figure 4-17 Output of SKMT Generated ECC Vectors

The certificates generated from Figure 4-16 or Figure 4-17 can be used for the rest of the operations described in this application note.

4.6 Authenticated Production Programming

Download rfp_rx74m.zip and extract the content to a local folder. Launch the RFP and open the rfp_rx74m.rpj project. In this sample RFP project, the FINE interface is selected as the boot interface, and the FINE interface boot mode is accessed by using a USB Type-C® cable.

If the FSBL has been enabled in an earlier application or the Protect Zone boundary has been set up previously, the initialization command must be run prior to proceeding to the following section. This will delete the previous FSBL and Protect Zone settings.

Next, use the RFP to download the application to the MCU and provide the items of authentication information to authenticate the application. Through the flash option programming, some items of authentication information will be injected into the MCU. Ensure the following **Operation Settings** are configured to erase the previous application, download the application and program the Flash Options.

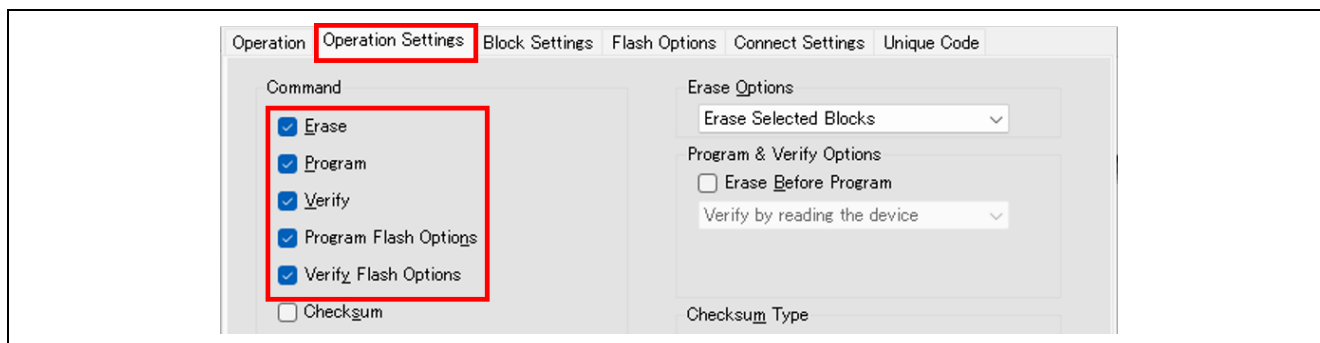


Figure 4-18 Configure the Operation Settings

Select the application file to be programmed into the MCU. On the **Operation** tab, click **Add/Remove Files**. Click **Add File(s)** and select blinky_hmac_boot_rx74m.mot. Additionally, select ARCCS_ARCMD_mode1.mot or ARCCS_ARCMD_mode2.mot in accord with the setting of ARCMD. Click **OK**.

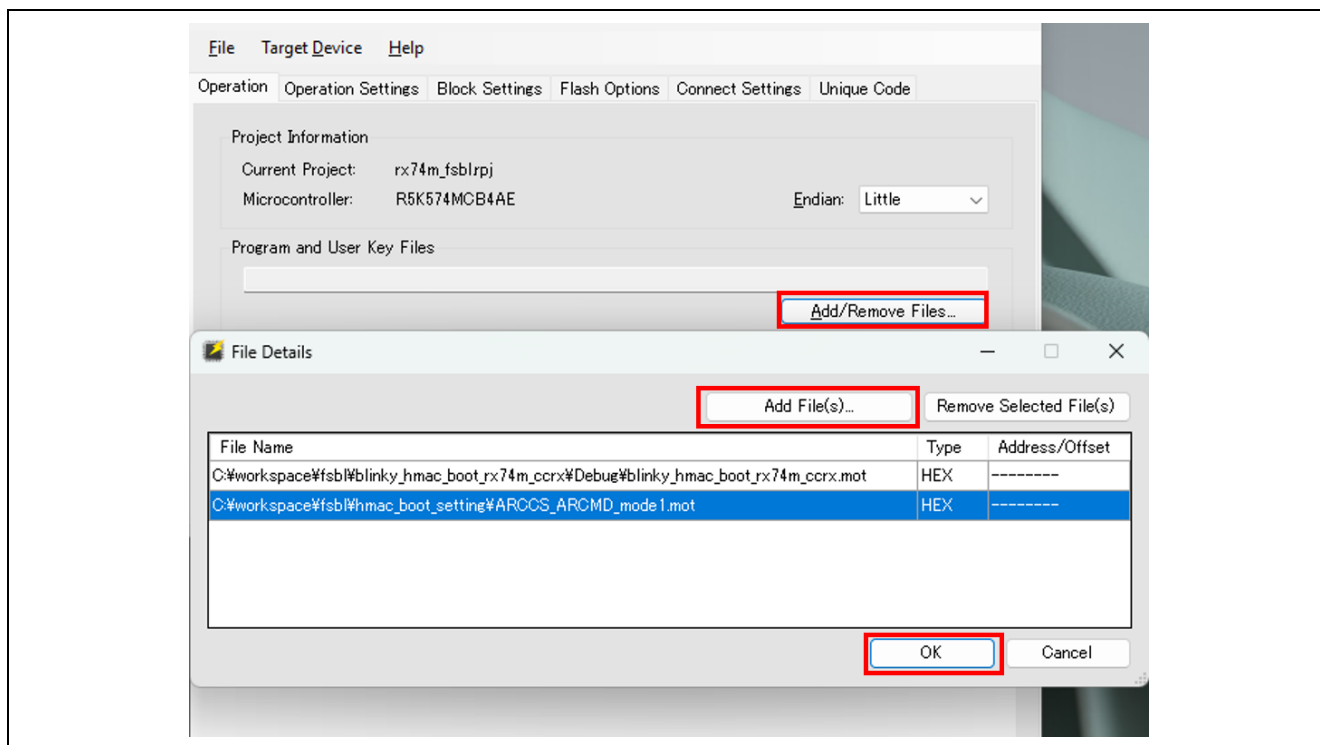


Figure 4-19 Select the Application Project

On the **Flash Options** tab, set the **OEM Root Public Key**, **Code Certificate**, and **Key Certificate**.

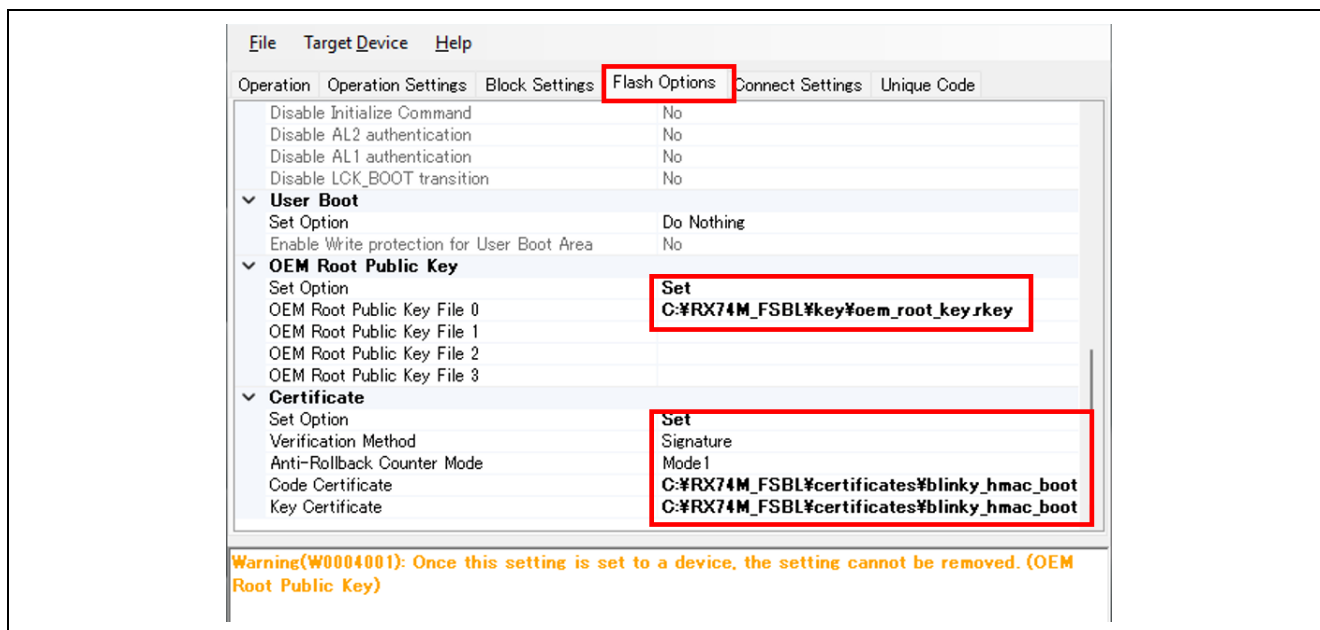


Figure 4-20 Set the OEM_ROOT_PK and Certificates

Now navigate to the **Operation** tab and click the **Start** button. The RFP will read the SACC0 register value from the .mot file and program the code certificate at the indicated memory address. The FSBL will compare the version number in the code certificate, and if the version number is higher, the authentication of the OEM_BL will proceed. If the authentication is successful, the OEM_BL digest will be programmed into the area after the code certificate. On the next power cycle, the FSBL will authenticate the application by using the stored digest and the FSBL calculated digest. If the comparison is successful, the new application will be started up, after which normal operation can be confirmed by seeing the three LEDs blink.

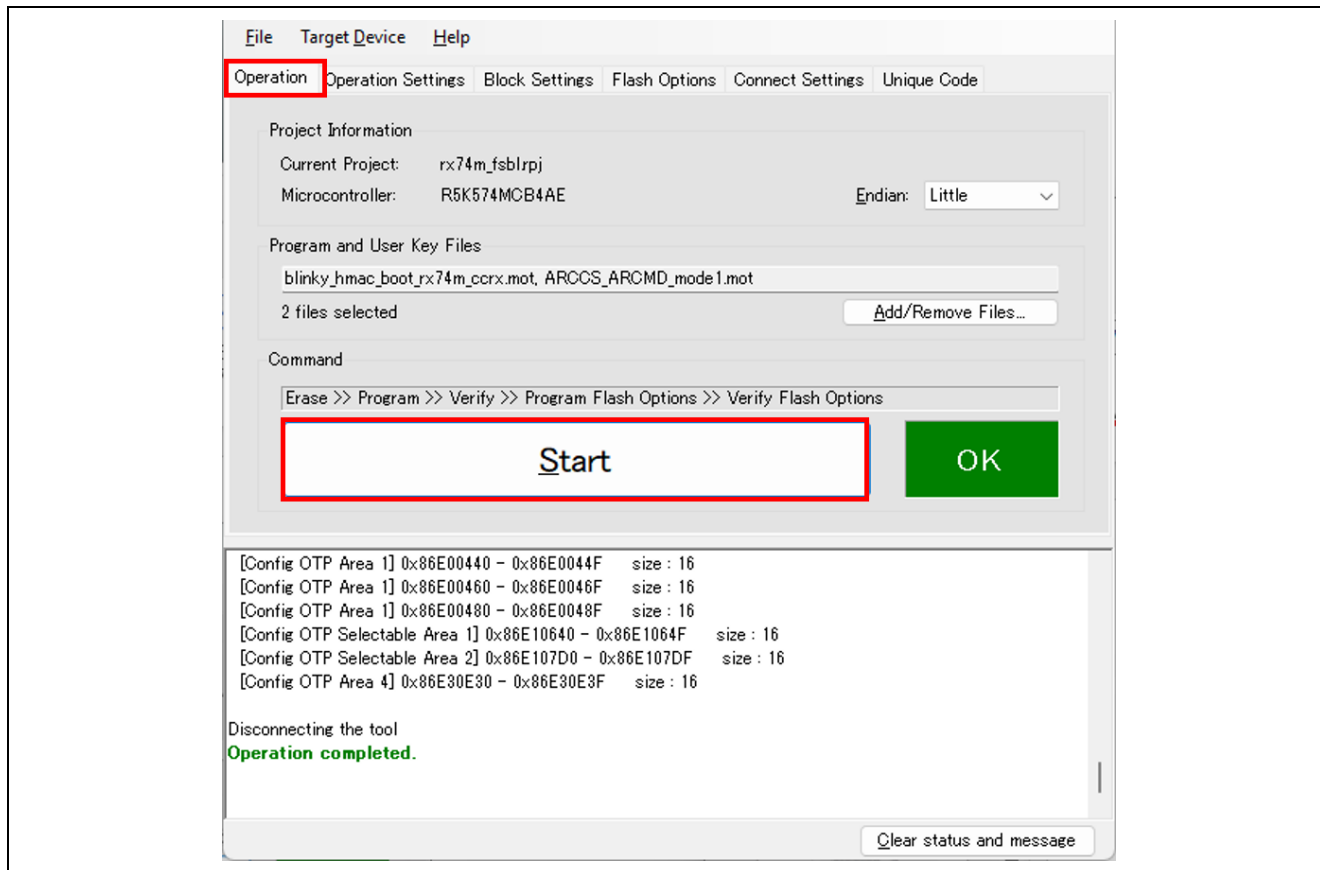
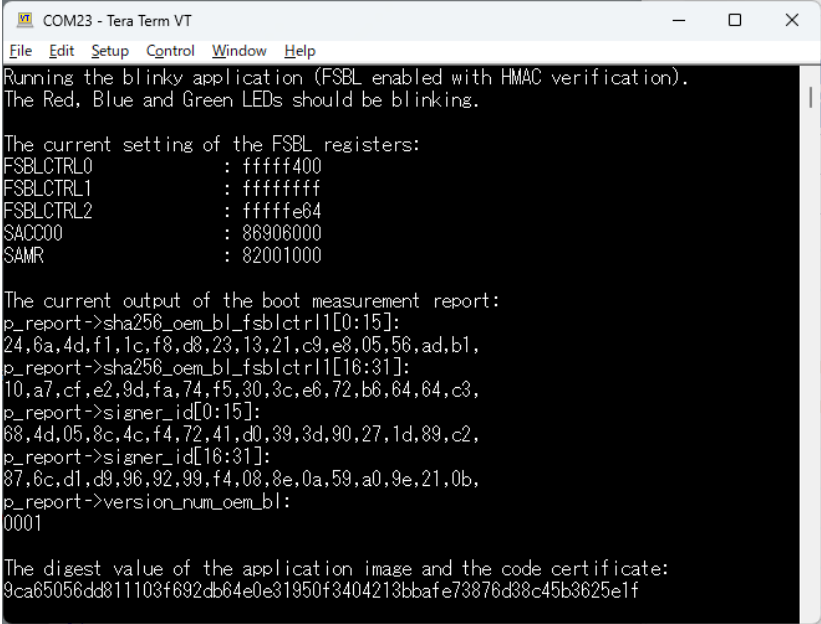


Figure 4-21 Successful Update of the Application Based on HMAC SHA2-256 Verification

Upon a successful boot, the Tera Term console will output similar information as the following. The version_num_oem_bl and the image digest will vary based on the kit status and the application programmed.



```

COM23 - Tera Term VT
File Edit Setup Control Window Help
Running the blinky application (FSBL enabled with HMAC verification).
The Red, Blue and Green LEDs should be blinking.

The current setting of the FSBL registers:
FSBLCTRL0      : fffff400
FSBLCTRL1      : ffffffff
FSBLCTRL2      : fffffe64
SACCD00        : 86906000
SAMR           : 82001000

The current output of the boot measurement report:
p_report->sha256_oem_bl_fsblctrl1[0:15]:
24,6a,4d,f1,1c,f8,d8,23,13,21,c9,e8,05,56,ad,b1,
p_report->sha256_oem_bl_fsblctrl1[16:31]:
10,a7,cf,e2,9d,fa,74,f5,30,3c,e6,72,b6,64,64,c3,
p_report->signer_id[0:15]:
68,4d,05,8c,4c,f4,72,41,d0,39,3d,90,27,1d,89,c2,
p_report->signer_id[16:31]:
87,6c,d1,d9,96,92,99,f4,08,8e,0a,59,a0,9e,21,0b,
p_report->version_num_oem_bl:
0001

The digest value of the application image and the code certificate:
9ca65056dd811103f692db64e0e31950f3404213bbafe73876d38c45b3625e1f

```

Figure 4-22 Console Output for an HMAC Boot Using the Certificate

Demonstrating a failure in an HMAC boot is easy. First, initialize the device, and then perform the steps in section 4.6. However, if a wrong application image, for example, part of the OEM_BL is rewritten, the RFP programming operation in Figure 4-21 will fail and the signal on pin P604 will go to the high level, turning on the red LED.

5. Appendix

5.1 Using the FSBL with Renesas Secure Factory Programming

The RX74M supports programming an encrypted application program image with Secure Factory Programming, enabling secure programming of the application program in a non-secure environment.

Renesas Secure Factory Programming can be implemented with or without the FSBL. When the FSBL is to be used with Secure Factory Programming, the application must set the FSBL. The FSBL is booted after the encrypted image is programmed to the MCU and before executing a reset. For details, refer to the “Secure Factory Programming” section of the *RX74M Group User's Manual: Hardware*.

5.2 Register Settings for the FSBL

The FSBL setting options can be selected via the RA Smart Configurator **BSP** tab. The settings for the FSBL shown below demonstrate an example where **HMAC boot with report measurement** is selected for **FSBLEXMDFSBLEN** on the **Properties** tab. Adjust these settings based on the security objectives of the application being developed. Refer to the “Build Time Configurations” in the RX Flexible Software Package Documentation > API Reference > BSP > MCU Board Support Package on the Renesas GitHub page (<https://github.com/renesas>) for more explanations of the various Stack properties.

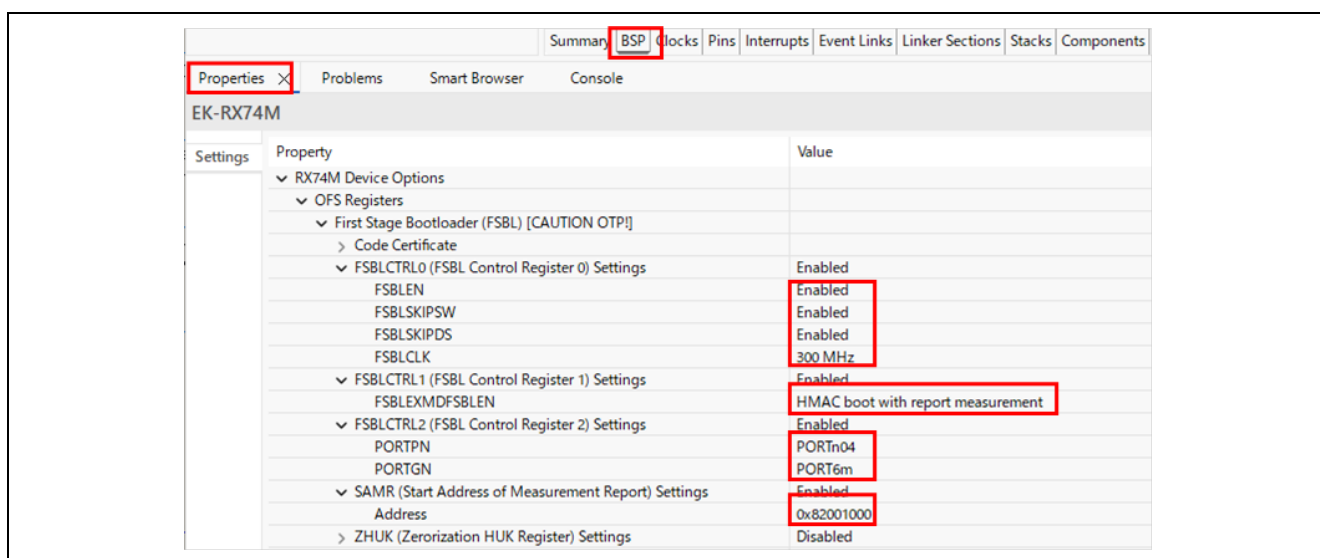


Figure 5-1 FSBL Settings

Applications targeting the RX74M are not always required to use the FSBL. The FSBL can be disabled by setting the **FSBLEN** property to **Disabled**. In this case, there is no verification of the application code during programming or prior to execution. The MCU with the FSBL set to disabled can be used in the same way as other RX MCUs that do not have the FSBL installed.

Since the FSBL will take some time to perform the verification, it may be acceptable to skip the verification process after certain types of reset. Set the **FSBLSKIPSW** property to **Enabled** to skip the FSBL upon a software reset. Similarly, set the **FSBLSKIPDS** property to **Enabled** to skip the FSBL upon a reset from the deep software standby reset.

In both CRC boot and HMAC boot use cases, it is possible to select with or without “**report measurement**” option as shown in Figure 5-2.

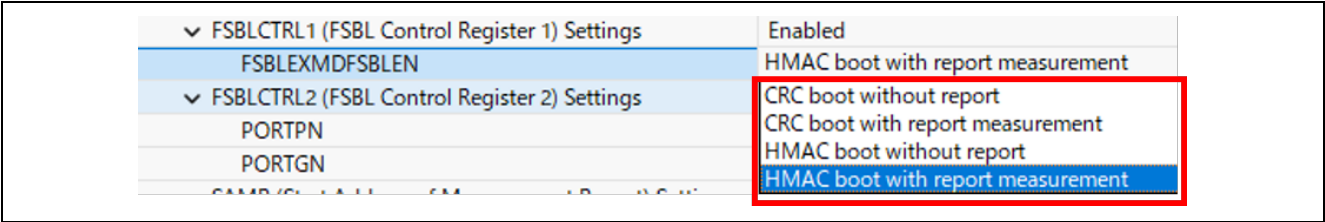


Figure 5-2 Verification Options Using the FSBL

If “**report measurement**” is selected, the information in Figure 5-3 is output to the memory address stored in the SAMR register, which is configured by using the **FSBL Measurement Report Address** property on the BSP tab (see Figure 5-1). The content of the measurement report is generated to meet the requirements of the [Arm Platform Security Architecture Attestation API](#). The latest version of the Arm PSA Certified Attestation API manual from this link has explanations on the usage of these entities.

SRAM address specified by SAMR register + 00h	SHA256 hash value of OEM_BL and FSBLCTRL1[7:0]
SRAM address specified by SAMR register + 20h	Signer ID (SHA256 hash value of OEM_BL_PK)
SRAM address specified by SAMR register + 40h	Version number of OEM_BL
SRAM address specified by SAMR register + 43h	
Note: OEM_BL and FSBLCTRL[7:0] are entered into the hash function in this order.	

Figure 5-3 Content and Location of the Measurement Report

For a CRC boot, the calculated CRC is output eight times as the Signer ID. Each CRC32 is 4 bytes, and the Signer ID is 32 bytes.

Upon a failed boot, the high level is output to the pin set by the **FSBLCTRL2** register.

The FSBL registers are mapped into the **Extra MRAM option setting memory area**. Programming of these registers is possible by using serial programming (JTAG/SWD or FINE) with an IDE (e² studio, etc.). Programming of the application using the RFP is performed via the programming interface. Thus, the settings made in the FBSL registers determine whether the FSBL will be booted up or a direct jump to the OEM_BL will be made after a reset.

6. References

1. RX Family Secure User Key Injection and Update Application Note (R01AN8067)
2. RX74M Group User's Manual: Hardware (R01UH1055)
3. Renesas Security Key Management Tool User's Manual (R20UT5349)
4. Renesas Flash Programmer Flash Memory Programming Software User's Manual (R20UT5820)

7. Website and Support

Visit the following URLs to learn about the RA family of microcontrollers, download tools and documentation, and get support.

EK-RX74M Resources renesas.com/rx/ek-rx74m

RX Product Information renesas.com/rx

Flexible Software Package (FSP) renesas.com/rx/fsp

RX Product Support Forum renesas.com/rx/forum

Renesas Support renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct.22.26	—	Initial release

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/