Renesas RA Family

Application Design using RA8 First Stage Bootloader

Introduction

The Renesas RA8 Series MCUs listed below make available a “First Stage Bootloader” (FSBL), which is masked in ROM and can execute after reset to verify the OEM firmware programmed in the on-chip flash in the single-chip operating mode. Due to its immutable nature, the FSBL provides a silicon-based Root of Trust (RoT).

In order to use the FSBL, the OEM firmware must be verified during production programming. The MCU’s factory boot firmware provides this capability.

This application project provides a walk-through for how to establish the credentials used by the factory boot firmware and the FSBL to verify the OEM firmware. The application project then demonstrates the verification of the firmware using either technique available: Cyclic Redundancy Check 32 (CRC32) integrity verification or Elliptic Curve Cryptography (ECC) NIST P256 signature authentication combined with HMAC-SHA256 authentication.

Required Resources

Target Devices

Below are the Renesas MCU products to which the information within this document is applicable:

- RA8M1
- RA8D1
- RA8T1

Software and development tools

- e² studio IDE v2023-10
- Renesas Flexible Software Package (FSP) v5.1.0

The links to download the above software are available at https://github.com/renesas/fsp.

- SEGGER J-Link® USB driver v7.92o or later (SEGGER J-Link)
- Renesas Secure Key Management Tool v1.05 or later
- Renesas Flash Programmer v3.13 or later
- GNU Privacy Guard for Windows: Gpg4win
- Renesas Device Lifecycle Management Server

Hardware

- EK-RA8M1, Evaluation Kit for RA8M1 MCU Group (renesas.com/ra/ek-ra8m1)
- Workstation running Windows® 10 and the Tera Term console or similar application.
- One USB device cable (type-A male to micro-B male)

Prerequisites and Intended Audience

This application project assumes that you have experience using Renesas e^2 studio IDE. In addition, knowledge on application boot loading and cryptographic algorithms is desirable. Prior to exercising this application project, the user should install all the tools mentioned in the **Software and development tools** section and read the following sections in the RA8M1 Hardware User's Manual. This will provide a background for some discussions in this application project. Reading these sections will also provide convenience for the user to dig deeper and extend the learning from this application project.

- Section: Security Features
- Section: Option-Setting Memory
- Section: Flash Memory

Furthermore, the user should have the Renesas RA Secure Key Injection Application Project (R11AN0496) handy. The user will need to follow several sections in this application project to support the FSBL usage. A non-TrustZone® application is used for demonstration in this application project. Users who are interested in using the FSBL with TrustZone projects can preview section 4.1 to understand the updates needed.

Contents

1. Introduction to the First Stage Bootloader

1.1 Overview

In a secure system, an application program must be executed only after confirming that it has not been altered, either maliciously or inadvertently. This confirmation, typically performed by a boot loader, can be a simple integrity check, or it can involve signature verification to ensure authenticity. However, it is important to note that the legitimacy of the boot loader itself must also be guaranteed.

When the FSBL is enabled, there are two options to verify the product firmware.

- Integrity check using CRC32. This check will be performed at initial MCU programming and prior to execution.
- Authenticity check. During initial MCU programming, the application is authenticated using ECDSA with the secp256r1 (NIST P256) ECC curve. Prior to execution, the application is authenticated using HMAC-SHA256 with the MCU's HUK.

The secure boot sequence is often implemented in stages, starting with an immutable entity to ensure a strong Root of Trust (RoT). The RA8 Series MCUs include an immutable First Stage Bootloader (FSBL), which provides this RoT as a starting point for a secure boot sequence. Typically, the FSBL validates the OEM's bootloader, which then validates the remainder OEM firmware, forming a "Chain of Trust" (refer to Figure 1 Case 2 for the authenticate path) for booting the system. Thus, the availability of the FSBL allows for securely updating the OEM bootloader.

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

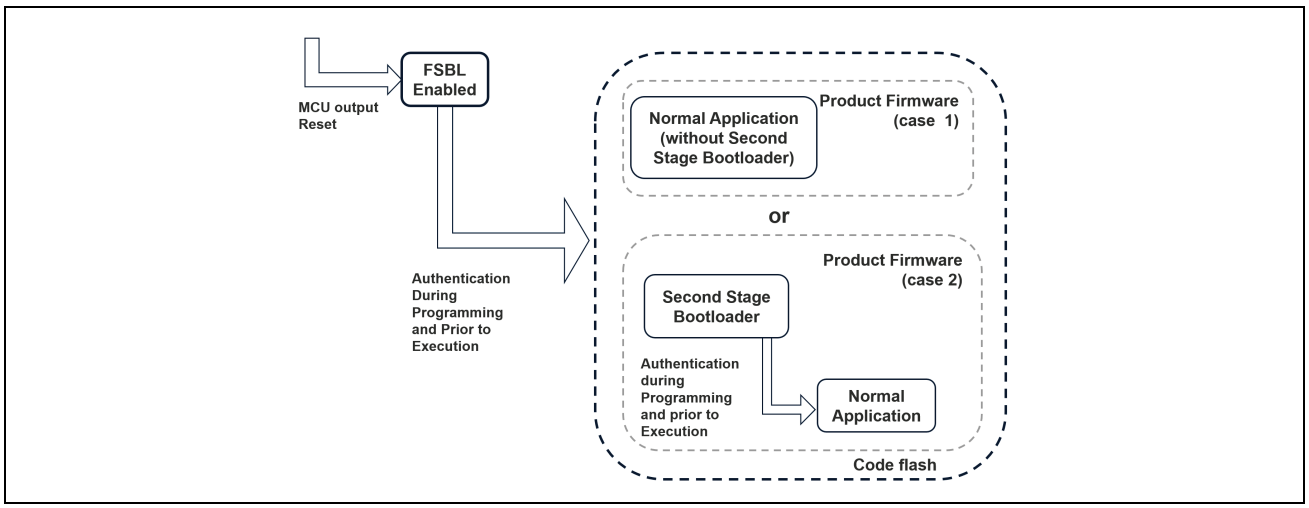


Figure 1. Usage of the FSBL in a Single Chip Mode

Note that in the RA8 MCU Hardware User's Manual, the description is centered around the second use case where a Second Stage Bootloader exists. The Hardware UM uses "OEM_BL" to refer to both usages when it comes to operations related with FSBL. In this release of the application project, only the first usage is covered. To be consistent with the Hardware User's Manual, we will use "OEM_BL" to refer to OEM product firmware.

As shown in the image below, the FSBL registers are mapped into the **Data flash option setting memory** area. The programming of these registers can be handled by the Serial programming/ JTAG or SWD programming when using the IDEs (for example, e^2 studio). They are programmed via the boot access modes when using RFP to program the applications. As a result, the settings made in these FBSL registers will determine the MCU's operation state coming out of Reset and the decision of whether to jump directly into OEM_BL.

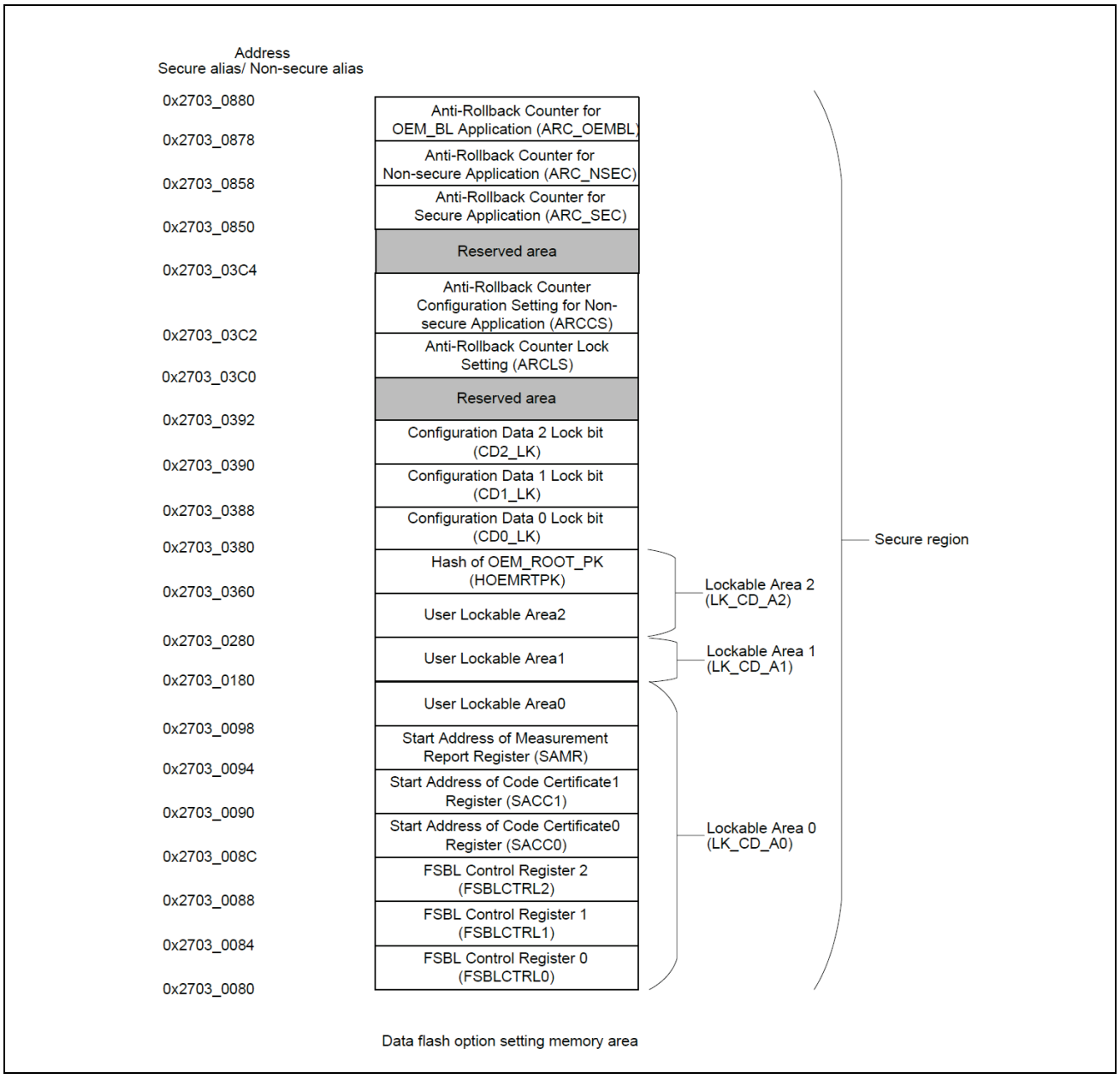


Figure 2. Data Flash Option Setting Memory Area RA8x1

The lockable data flash option setting memory area can be locked via the Renesas Flash Programming tool during production or other production programs. Be very cautious – after these areas are locked, they cannot be unlocked to reenable the configuration of these registers. Locking up these registers should be avoided during application development.

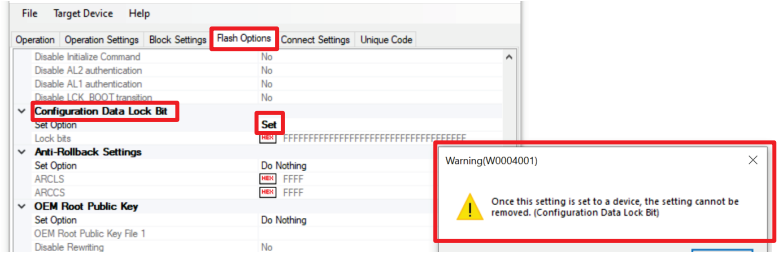


Figure 3. Using RFP to Lock the Lockable Data Flash Option Setting Registers

1.2 FSBL based Booting Options

When the FSBL is enabled, there are two schemes supported for the verification of OEM_BL.

- **CRC32 boot**: in this case, the application integrity is validated using CRC32.
- **Secure boot**: in this case, the application authenticity is validated upon programming using the secp256r1 ECC curve (NIST P256). The application authenticity is validated upon execution using HMAC-SHA256 with the MCU's Hardware Unique Key.

The FSBL configuration options can be selected via the RA Smart Configurator **BSP** tab. The configuration settings for the FSBL shown below demonstrate an example where **Secure boot with report measurement** is selected. Adjust these settings based on the security objectives of the OEM application being developed. Refer to the "Build Time Configurations" for your MCU in the RA Flexible Software Package Documentation > API Reference > BSP > MCU Board Support Package for more explanations of the various Stack properties.

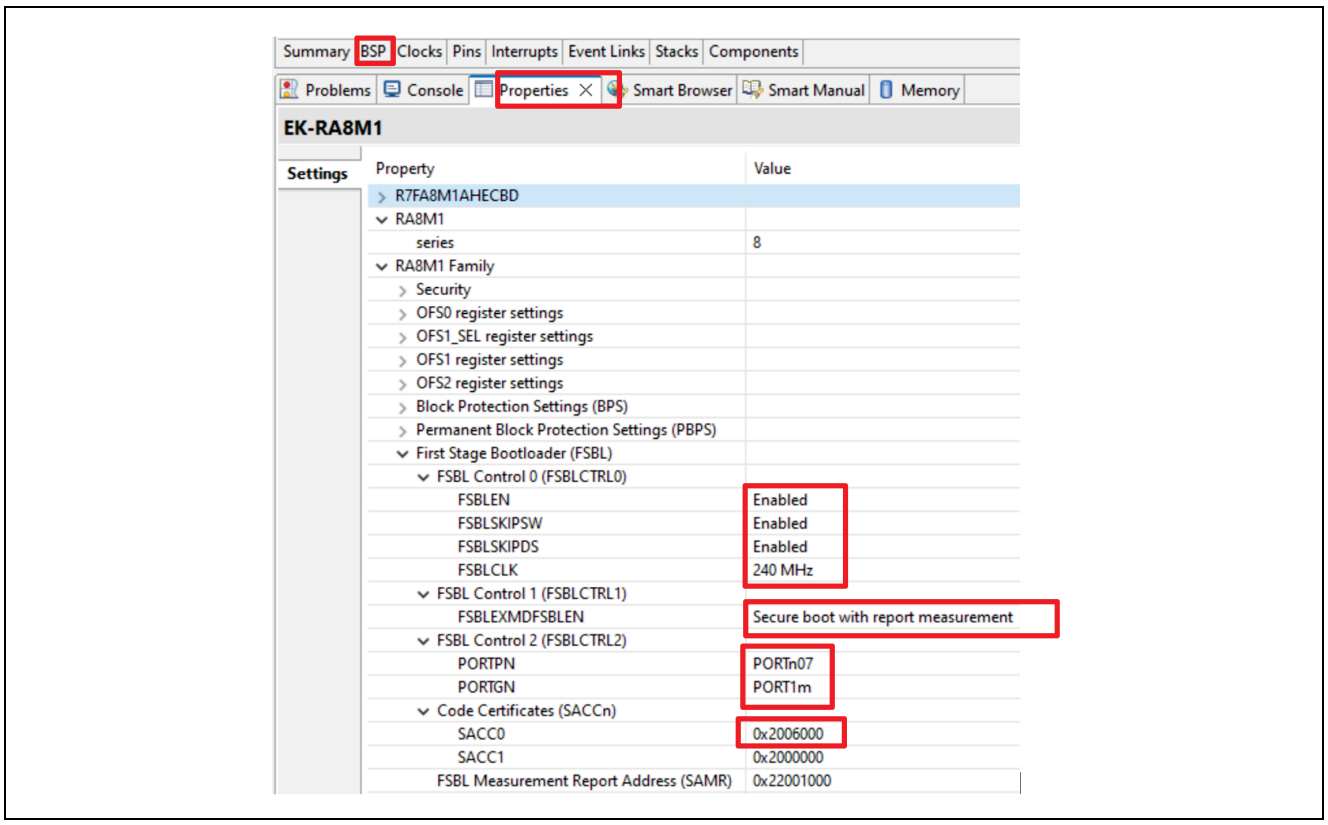


Figure 4. FSBL Configurations

Applications targeting RA8 Series MCUs are not required to use the FSBL. The FSBL can be bypassed. This is configured by setting the **FSBLEN** property to **Disabled**. In this case, there is no verification of the application code during either programming or prior to execution. The MCU can be used in the same way as other RA MCUs that do not include an FSBL.

Since the FSBL will take some time to perform the verification, it may be acceptable to skip the verification process after certain types of resets. Configure the **FSBLSKIPSW** property to **Enabled** to skip the FSBL upon a software reset. Similarly, configure the **FSBLSKIPDS** property to **Enabled** to skip the FSBL upon reset from at Deep Software Standby Reset.

In both CRC and Secure boot use cases, it is possible to choose with or without "**report measurement**" option as shown in Figure 5. Verification Options Using FSBL .

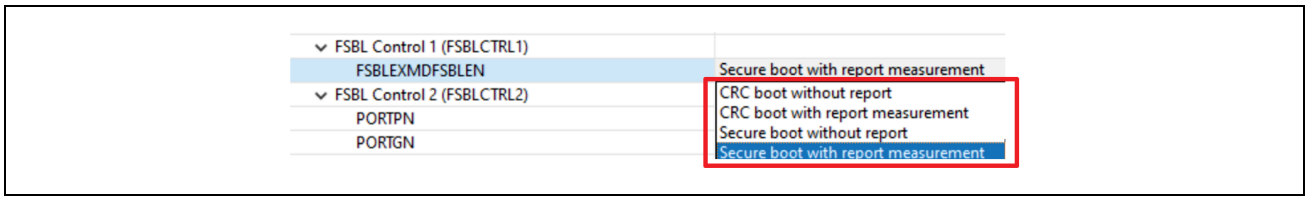


Figure 5. Verification Options Using FSBL

If report measurement is selected, the information in Figure 6 is output to the memory address stored in the SAMR register, which is configured using the **FSBL Measurement Report Address** property under the BSP tab (refer to). The contents in the Measurement Report are 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 + 0x00	SHA2-256 hash value of OEM_BL and FSBLCTRL1[7:0]
SRAM address specified by SAMR register + 0x20	Signer ID (SHA2-256 hash value of OEM_BL_PK)
SRAM address specified by SAMR register + 0x40	Version number of OEM_BL
SRAM address specified by SAMR register + 0x43	

Note: OEM_BL and FSBLCTRL[7:0] are entered into the hash function in this order.

Figure 6. Content and Location of the Measurement Report

For CRC boot, the Signer ID will output the calculated CRC 8 times. Each CRC32 is 4 bytes, and the Signer ID is 32 bytes. Using **CRC boot with report measurement** is a rare use case.

Upon a failed boot, the high level is output to the port set by the FSBLCTRL2 register (refer to Figure 4) and the MCU goes to CPU Sleep mode. For example, in Figure 4, Pin 107 is selected as the pin to output a high level when the secure boot fails. This pin controls the red LED on board, which will be turned on with a failed boot sequence.

1.2.1 OEM_BL Verification Overview During Production Programming

During production programming, the OEM_BL is validated by the MCU boot firmware using ECDSA. The RA8 MCU offers access to the MCU boot mode through JTAG, SCI and USB interfaces. For details of the boot mode, refer to section **Details of Operating Modes** in the Hardware User's Manual.

After establishing communication with the boot firmware (for example, through RFP), the boot firmware injects the hash of the OEM_ROOT_PK to unmapped flash area as the RoT of the system. Once the OEM_BL is programmed to the MCU, the boot firmware then reads the **Image Version** number in the code certificate. If the **Image Version** is a higher version than previously programmed, the boot firmware calculates the hash of the OEM_ROOT_PK (which is stored in the key certificate) and compare with the Hash of OEM_ROOT_PK that is programmed to the MCU. If the hash of the OEM_ROOT_PK matches, the OEM_BL_PK in the code certificate is verified by comparing the calculated hash of the OEM_BL_PK to the hash residing in the key certificate. Once the OEM_BL_PK is verified, it will be used by the boot firmware to verify the OEM_BL using the signature in the code certificate. If the OEM_BL is validated, the boot firmware increments the anti-rollback counter up to the Image Version written in the code certificate and calculates the OEM_BL_digest. The code certificate and the OEM_BL_digest are then programmed to the MCU.

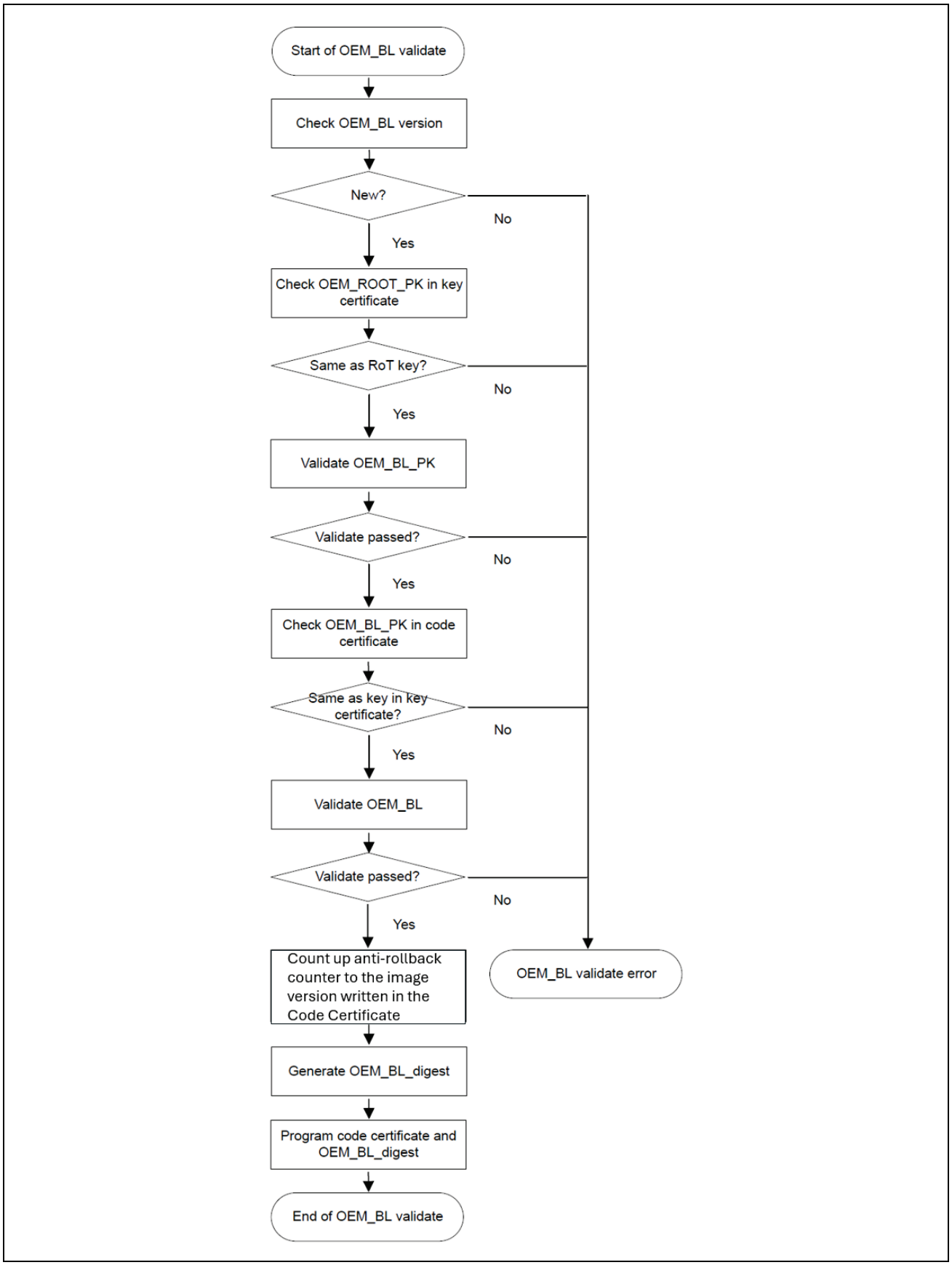


Figure 7. OEM_BL Validation Flow in Production Programming Mode

1.2.2 OEM_BL Verification During Single-Chip Operation

In single chip mode, the OEM_BL is validated by the FSBL using HMAC-SHA256. Assume the OEM_BL production programming is successful. When **Secure boot** is enabled, the immutable FSBL in ROM is executed after MCU reset. The FSBL calculates the HMAC value (the OEM_BL_digest) of the OEM_BL plus the code certificate and compares it to the expected OEM_BL_digest, which is stored in the code flash during production programming. If the OEM_BL_digest matches, the FSBL jumps to the OEM_BL. If not, then FSBL will transition the CPU to a sleep mode and optionally output a level 'high' to a port pin of user's choice.

When CRC boot is selected, the FSBL calculates the CRC of the OEM_BL and compares it to the expected CRC value, which is located in the code certificate. If the CRC values match, the FSBL jumps to the OEM_BL.

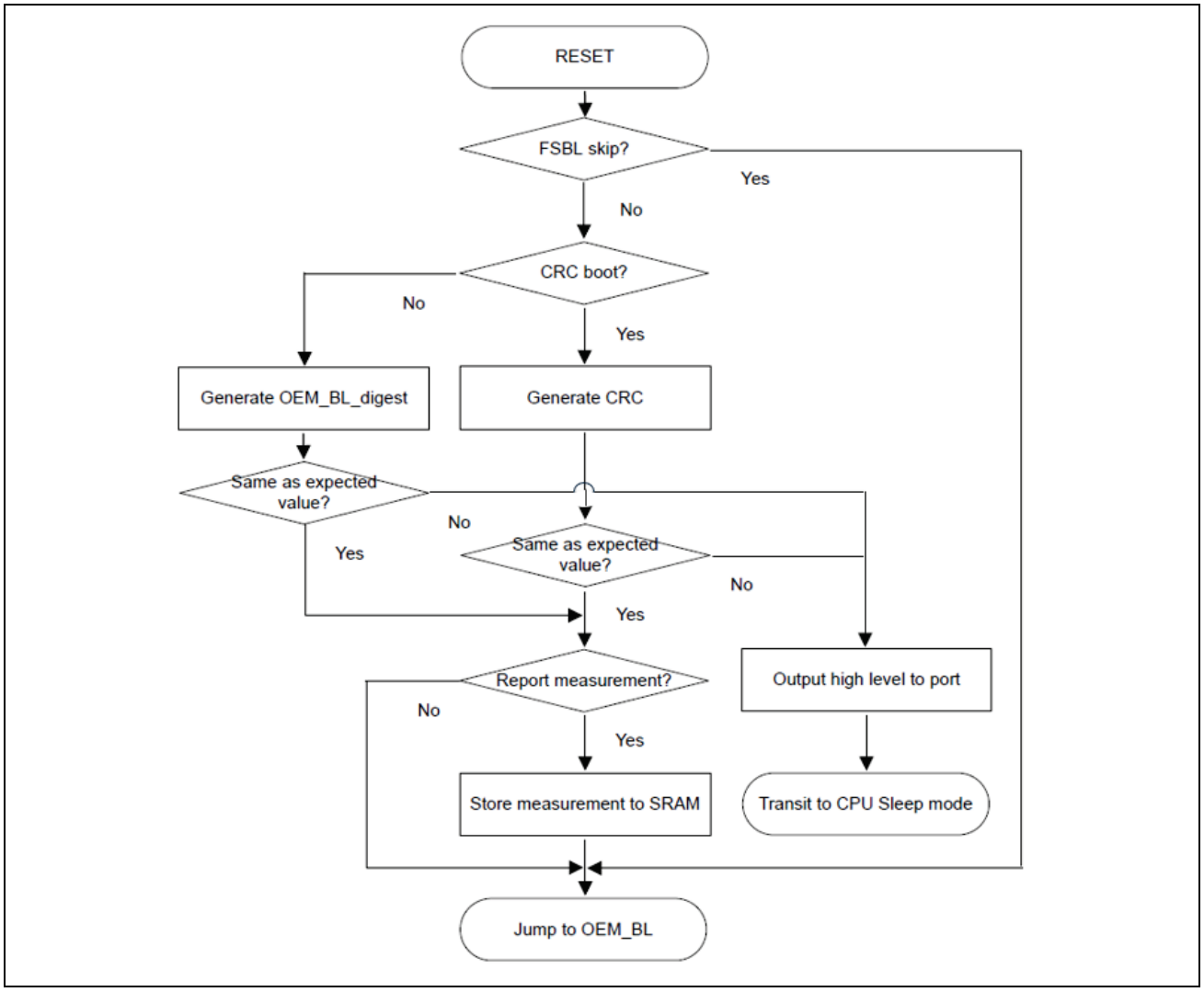


Figure 8. OEM_BL Validation in Single-Chip Mode

1.2.3 Advantages of Using HMAC-SHA256 in Single-Chip Mode

The HMAC-SHA256 provides advantages based on the RA8 MCU hardware features:

- Security is maintained because the HMAC "shared secret" (the MCU's HUK) is never exposed outside the MCU.
- The HUK is MCU unique and provides anti-cloning protection.
- The operation is significantly faster than verifying a digital signature.
- The HMAC-SHA256 operation is Quantum-resistant.

1.3 Details of the Credentials Used in Authenticity Checking

This section describes the chain of trust of the system based on ECDSA and HMAC-SHA256 operations.

The chain of trust of the system is established with the following components.

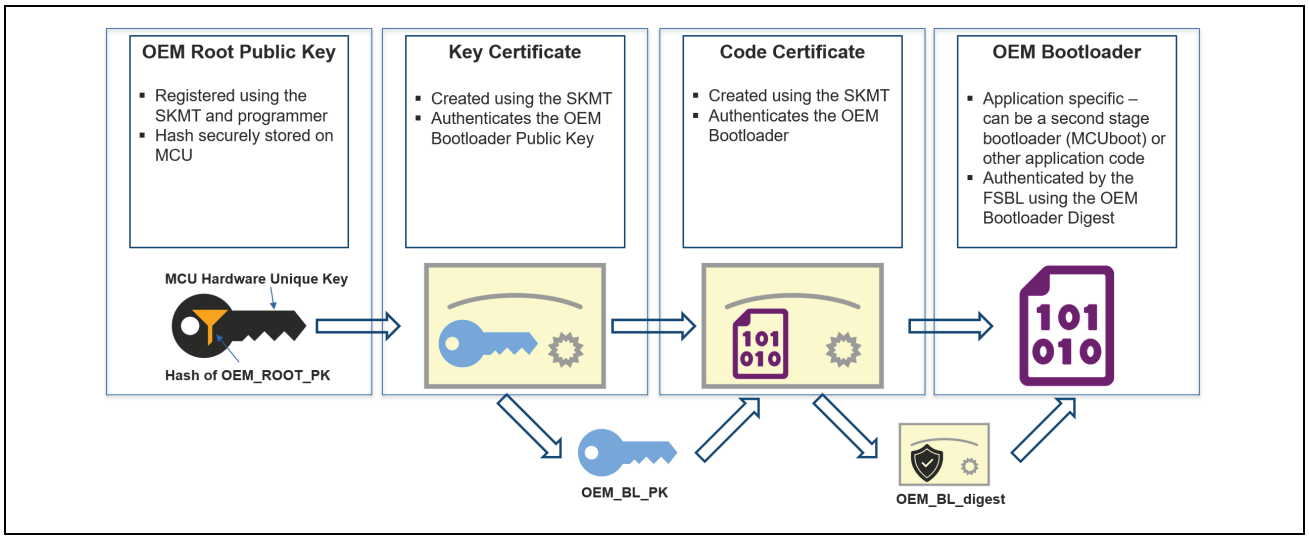


Figure 9. Chain of Trust

The following sections describe the format and content of the various credentials described in Figure 9 and their usage during production programming and prior to production firmware execution.

1.3.1 Root of Trust

Two sets of secp256r1 ECC Key Pairs must be generated when using the Secure boot:

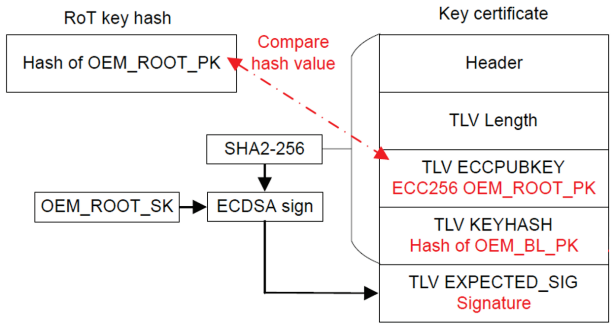
- OEM_ROOT_PK: the public part of the OEM Root Key pair
- OEM_ROOT_SK: the private (secret) part of the OEM Root Key pair
- OEM_BL_PK: the public part of the OEM_BL Key pair
- OEM_BL_SK: the private (secret) part of the OEM_BL Key pair

The SHA256 hash of the OEM_ROOT_PK is stored in the lockable data flash area (HOEMRTPK) as the Root of Trust (RoT) using the secure key injection process described in section “Secure Key Injection” in the Hardware User’s Manual for your RA8 MCUs. The injection of the Root of Trust is done during the production programming of the application.

1.3.2 Key Certificate

The key certificate plays a critical role 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 OEM_ROOT_SK.
- The OEM_ROOT_PK is included in the signed key certificate and is verified by the HOEMRTPK.
- The hash of the OEM_BL_PK is included in the key certificate. It will be used by the boot firmware to verify the OEM_BL_PK.
- The key certificate is discarded after the OEM_BL PK is validated.



Field		Size [byte]	Description
Header	Magic	4	Set to 0x6B657963
	Manifest version	4	Set to 0x00010000
	Flags	4	Reserved for future use
	Reserved	20	Reserved for future use
TLV Length		4	Length in bytes of TLV field
TLV ECCPUBKEY	Type & Length	4	Set to 0x00088010
	Value	64	ECC P-256 OEM_ROOT_PK
TLV KEYHASH	Type & Length	4	Set to 0x10144008
	Value	32	SHA2-256 hash value of OEM_BL_PK
TLV EXPECTED_SIG	Type & Length	4	Set to 0x20088410
	Value	64	Signature

Figure 10. Key Certificate to Verify the OEM_BL_PK

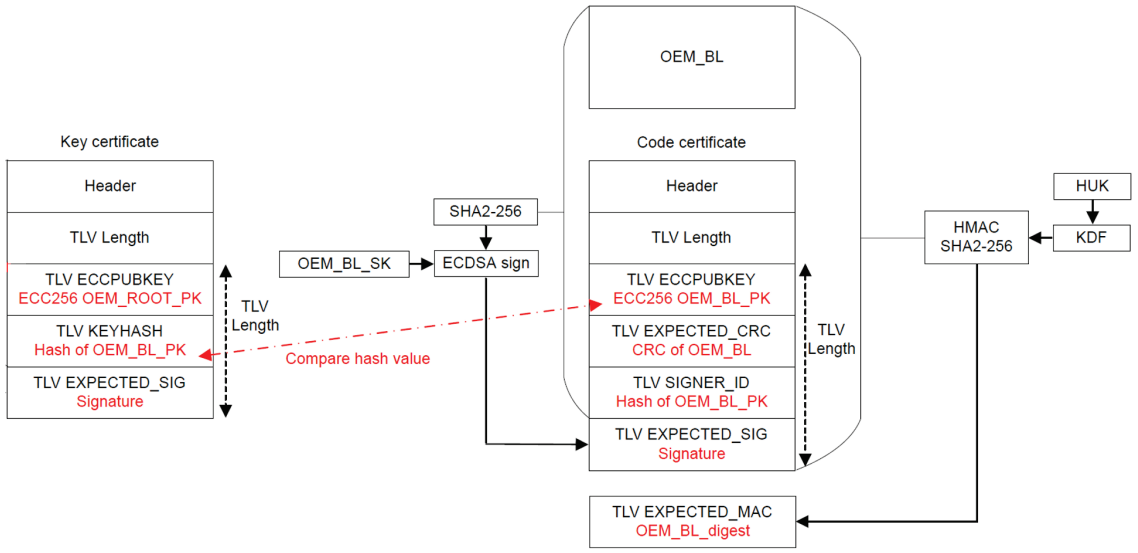
1.3.3 Code Certificate

The code certificate plays a critical role during MCU production programming as well as during MCU single chip operating mode.

- The code certificate is signed by OEM_BL_SK.
- The OEM_BL_PK is included in the code certificate. It is verified by its hash, which was stored in the key certificate. The OEM_BL_PK will then be used to verify the OEM_BL's signature, included in the code certificate.
- The code certificate contains credentials used for the FSBL verification process during single-chip operation mode. It must be programmed into flash memory (code or data flash) after the OEM_BL is programmed.
- The application **Image Version** number is part of the code certificate when using Secure boot. The valid Image Version is 1 to 64, which implies the application can be updated 64 times when FSBL with Secure boot is enabled. The FSBL implements an anti-rollback scheme. Therefore, to update to a new application image, the next new image must have a version number higher than the previous application image. For more details on the operation and explanation of the usage of the "Image Version" number, please refer to the **Anti-Rollback Counter** Section of the Hardware User's Manual.
- The code certificate location in flash must be specified in the SACC0 register or SACC1 register. 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 default block (block 0) 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 block (block 1) in linear mode. Configuration of SACCx registers is handled by the RA Smart Configurator and the FSP Board Support Package (BSP). Figure 4 provides an example configuration for the Code Certificate programming location. Be sure to adapt it as required for your specific application.
- For FSBL signature authentication, the TLV EXPECTED_CRC field is not used. The structure of the Code Certificate, however, remains unchanged.
- After successful validation of the OEM_BL during production programming, the HMAC-SHA256 digest (OEM_BL_digest) of the OEM_BL plus the code certificate is then calculated by the boot firmware using the MCU's HUK. The boot firmware will program the OEM_BL_digest in the memory area immediately following the Code Certificate.

Field		Size [byte]	Description
TLV EXPECTED_MAC	Type & Length	4	Fixed value 0x30184008
	Value	32	Unique OEM_BL_digest in each MCU

Figure 11. Details of the OEM_BL_digest



Field		Size [byte]	Description
Header	Magic	4	Set to 0x636F6463
	Manifest version	4	Set to 0x00010000
	Flags	4	Set to 0x00000000
	Load Addr	4	Set to 0x02000000
	Dest Addr	4	Set to 0x02000000
	Image size	4	Size in bytes of OEM_BL set in multiples of 16, minimum size is 64 bytes
	Image version	4	Version number of OEM_BL. Can be specified from 1 to 64
	Build number	4	Set to 0x00000000
TLV length		4	Length in bytes of TLV field
TLV ECCPUBKEY	Type & Length	4	Set to 0x01088010
	Value	64	ECC P-256 OEM_BL_PK
TLV EXPECTED_CRC	Type & Length	4	Set to 0x40000001
	Value	4	CRC32 of OEM_BL
TLV SIGNER_ID	Type & Length	4	Set to 0x10144008
	Value	32	SHA2-256 hash value of OEM_BL_PK
TLV EXPECTED_SIG	Type & Length	4	Set to 0x25088410
	Value	64	Signature of [Code Certificate | OEM_BL], signed by the OEM_BL_SK When generating the signature, enter the code certificate and OEM_BL in the hash function in this order.

Figure 12. Code Certificate to Verify the OEM_BL using HMAC Boot

1.3.4 Summary of the Usage of the Credentials in Production Programing, Secure Boot

During the firmware application programming stage, the following items will be programmed on to the MCU:

- Hash of OEM_BOOT_PK
- OEM_BL (application image)
- Code Certificate
- OEM_BL_digest

The following figure summarizes the overall workflow using SKMT and RFP during production programming using Secure boot. A step-by-step walk-through of using SKMT and RFP is provided in section 2.

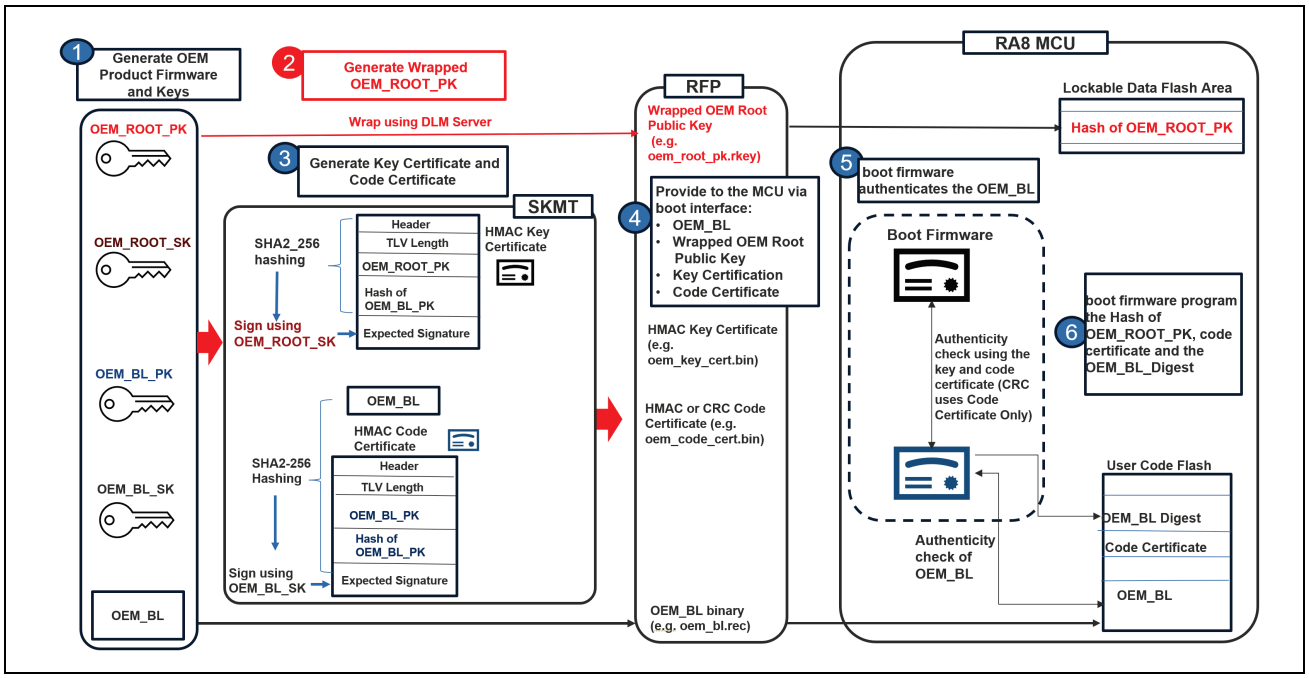


Figure 13. Summary of Credential Usage during Production Programming for Secure Boot

1.4 Details of the Credentials Used in Integrity Checking

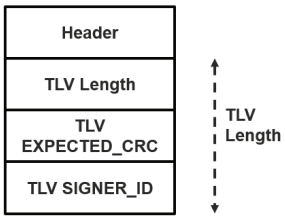
Validation using CRC is relatively simple. The differences from the **Secure boot** process described above are:

- The OEM Root Public Key (OEM_ROOT_PK) is not used. Therefore, there is no requirement for creating an OEM_ROOT key pair nor injecting the OEM_ROOT_PK to create the OEM_ROOT_PK's hash (HOEMRTPK).
- The Key Certificate is not used, and the OEM_BL is not signed. Therefore, there is no requirement to create an OEM_BL key pair.
- There is no anti-rollback policy enforced. Therefore, the anti-rollback counter (ARC_OEMBLn) and the Image Version field in the Code Certificate are not used.
- During production programming, the boot firmware calculates the CRC32 of the OEM_BL. If the calculated CRC32 value matches the value in the code certificate, the boot firmware programs the code certificate at the location specified by the SACC0/1 register.
- During single chip operation mode, the FSBL calculates the CRC32 of the OEM_BL compares it to the CRC in the code certificate. If the CRC values match, the FSBL jumps to the OEM_BL.

1.4.1 Code Certificate

The code certificate of the CRC boot includes the following components. The TLV ECCPUBKEY and TLV_EXPECTED_SIG fields do not exist in the Code Certificate. The Image Version field is not used during booting.

Code certificate



Field		Size [byte]	Description
Header	Magic	4	Set to 0x636F6463
	Manifest version	4	Set to 0x00010000
	Flags	4	Set to 0x00000000
	Load Addr	4	Set to 0x02000000
	Dest Addr	4	Set to 0x02000000
	Image size	4	Size in bytes of OEM_BL set in multiples of 16, minimum size is 64 bytes
	Image version	4	Version number of OEM_BL. Can be specified from 1 to 64
	Build number	4	Set to 0x00000000
TLV length		4	Length in bytes of TLV field
TLV EXPECTED_CRC	Type & Length	4	Set to 0x40000001
	Value	4	CRC32 of OEM_BL
TLV SIGNER_ID	Type & Length	4	Set to 0x10144008
	Value	32	SHA2-256 hash value of OEM_BL_PK

Figure 14. Code Certificate to Verify the OEM_BL using CRC Boot

1.4.2 Summary of the Usage of the Credentials in Production Programing, CRC boot

During the firmware application programming stage, the following items will be programmed on to the MCU:

- The OEM_BL (application image)
- The Code Certificate

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

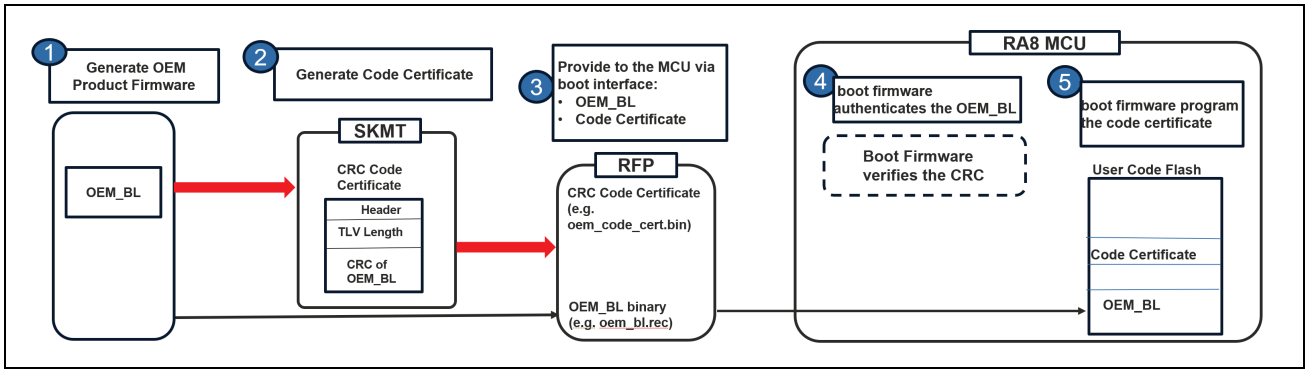


Figure 15. Summary of Credential Usage during Production Programming for CRC Boot

1.5 Using the FSBL with Renesas Secure Factory Programming

Renesas RA8 Series MCUs support programming an encrypted firmware image with Secure Factory Programming, enabling secure firmware programming in a non-secure environment. Renesas Secure Factory Programming can be implemented with or without FSBL. When FSBL is used with secure factory programming, the application must have FSBL configured. FSBL must be activated after the encrypted image is written into the MCU and before issuing a reset. To erase the device and turn off FSBL after secure factory programming, the MCU Initialize Device command should be performed (for example using the RFP as shown in Figure 23). For more details on the Secure Factory Programming, refer to the **Secure Factory Programming** section in the Hardware User's Manual.

2. Authenticated Production Programming and HMAC-SHA256 Boot Demonstration

This section provides a walk-through of generating the credentials for the authenticated production programming and HMAC boot. A simple blinky project is used to demonstrate this process. The procedure for injecting the Root of Trust has a lot in common to the secure key injection process described in R11AN0496. This application note will refer to sections in R11AN0496 whenever appropriate.

2.1 Prepare two Sets of ECC secp256r1 Key Pairs

As explained in the previous sections, two sets of ECC secp256r1 key pairs are needed to generate the chain of trust of the system. In this application project, using OpenSSL to generate the key pairs is demonstrated. Additionally, this writeup demonstrated using the NIST CAVP ECDSA test vectors to create the chain of trust.

2.1.1 Using OpenSSL to Generate the ECC Key Pairs

For Linux, get the OpenSSL from here: https://www.openssl.org/source/. For Windows, get the OpenSSL from here: https://slproweb.com/products/Win32OpenSSL.html.

After downloading OpenSSL, install it and navigate to the `\bin` folder from a command prompt.

The following command generates a pair of OEM Root Key Pair:

```
C:\Program Files\OpenSSL-Win64\bin>openssl ecparam -name prime256v1 -genkey -noout -
out c:\RA8_FSBL\openssl_keys\oem_root_private_key.pem
```

The following command generates the public key for the OEM Root Key Pair.

```
C:\Program Files\OpenSSL-Win64\bin>openssl ec -in
c:\RA8_FSBL\openssl_keys\oem_root_private_key.pem -pubout -out
c:\RA8_FSBL\openssl_keys\oem_root_public_key.pem

read EC key

writing EC key
```

The same commands can be used to generates a pair of OEM_BL Key Pair:

```
C:\Program Files\OpenSSL-Win64\bin>openssl ecparam -name prime256v1 -genkey -noout -
out c:\RA8_FSBL\openssl_keys\oem_bl_private_key.pem
```

The following command generates the public key for the OEM_BL Key Pair.

```
C:\Program Files\OpenSSL-Win64\bin>openssl ec -in
c:\RA8_FSBL\openssl_keys\oem_bl_private_key.pem -pubout -out
c:\RA8_FSBL\openssl_keys\oem_bl_public_key.pem

read EC key

writing EC key
```

2.1.2 Download NIST CAVP Test Vectors as ECC Key Pairs

The following NIST Cryptographic Algorithm Validation Program (CAVP) test vectors are used in this application project to demonstrate booting with HMAC with FSBL is used.

The CAVP NIST test vectors can be downloaded from the following link. The ECDSA vectors are what we will use.

Cryptographic Algorithm Validation Program | CSRC (nist.gov)

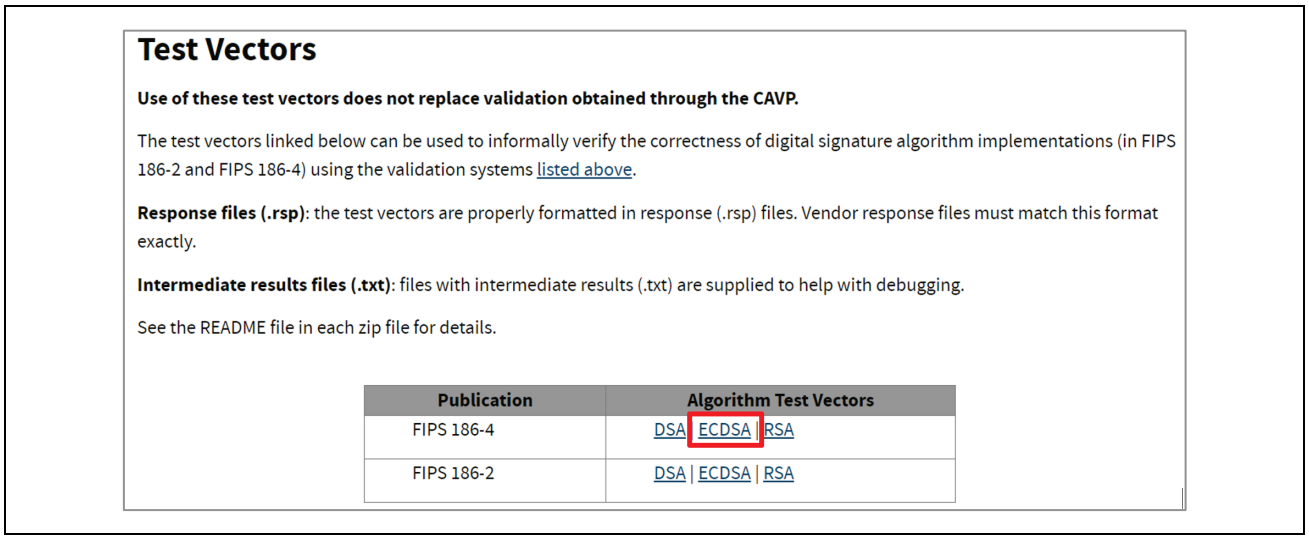


Figure 16. ECDSA Test Vectors

After downloading the zip file `186-4ecdsatestvectors.zip`, unzip it and find the following vectors in the plaintext file `KeyPair.rsp`.

ECC P256 secp256r1 key pair for the OEM Root Key Pair

The following vector is used for the OEM Root Key Pair. It is used in section 2.2 to generate the wrapped OEM Root Public Key. It is also used in section 2.4 to generate the key certificate.

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

ECC P256 secp256r1 key pair for the OEM APP (BL) Key Pair

The following vector is used for the OEM APP (BL) Key Pair. It is used in section 2.4 to generate the code certificate.

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

2.2 Generate the Root of Trust

As explained earlier, the hash of the OEM_ROOT_PK is the Root of Trust of the system and will be injected during a successful application programing when FSBL is enabled.

The OEM_ROOT_PK needs to be wrapped using the Renesas DLM server prior to injection to the MCU using RFP. Prior to wrapping the OEM Root Public Key, go through the process of Wrapping the User Factory Programming Key using the Renesas Key Wrap Service. The security considerations and the process of the UFPK wrapping is not repeated in this write up, reference the following sections in R11AN0496 to learn the usage background and establish the PGP encrypted communication with the Renesas DLM Server.

- Reference section **Create PGP Key Pair** to generate customer PGP Key Pair.
- Reference section **Registration with DLM Server** to register with the DLM Server.
- Reference section **Exchange User and Renesas PGP Public Keys** to establish encrypted communication with the DLM Server.

2.2.1 Generate the Wrapped User Factory Programming Key

After finishing the operations based on the above sections in R11AN0496, the next step is to reference section **Wrapping the UFPK** in R11AN0496 to generate W-UFPK. However, there are some critical steps to update during this process. Please read through the section **Wrapping the UFPK** first prior to reading the rest of this section to identify where to update the operations in order to generate the Wrapped UFPK for

OEM Root Public Key injection purpose. Afterwards, identify where to update the following operations in that flow and follow R11AN0496 to generate the W-UFPK.

- For a given input plaintext UPFK, the output of DLM server (W-UFPK) will vary based on the selected mode for the security engine. I.e.: Compatibility mode or Protected Mode. To be consistent with the option selected on the DLM Server when wrapping the OEM Root Key (which uses the MCU Protected Mode), choose the **RA Family, RSIP-E51A Security Functions and Protected Mode** option under the **Overview** page when using **SKMT** to generate the UFPK.
  - It is important to note that R11AN0496 uses RSIP Compatibility mode which implies selection of RA Family, RSIP-E51A Compatibility Mode in SKMT. Please pay attention to changing to the protected mode needed for FSBL usage.

Figure 17. Choose RSIP-E51A Security Functions and Protected Mode

- For the same reason, when using the DLM Sever to wrap the UFPK, it is critical to choose the following RA8 item under the **DLM and Protected Mode** rather than the Compatibility Mode entry.

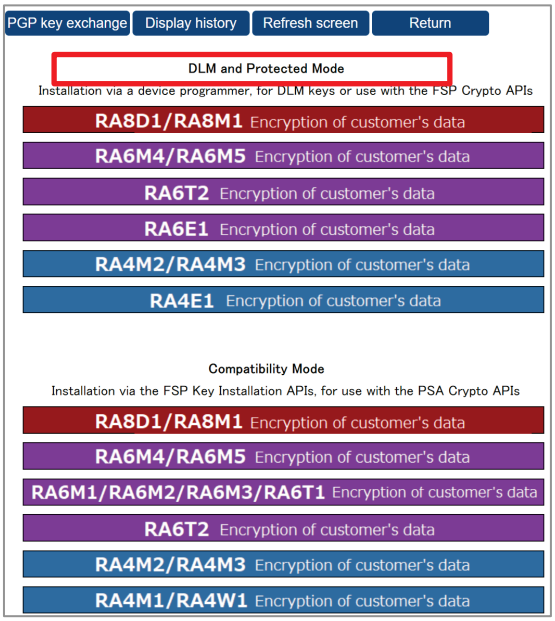


Figure 18. Select RA8D1/RA8M1 MCU Group DLM and Protected Mode

- The other steps for generating the Wrapped UFPK are identical to what is described in the R11AN0496.

The rest of this application project assumes that the following UFPK and W-UFPK have already been created:

`ra8x1_ufpk.key`: the UFPK
`ra8x1_ufpk.key_enc.key`: the DLM Wrapped W-UFPK

Once the UFPK and W-UFPK are generated, wrapping the OEM Root Public Key should be done in a similar way to the ECC Public Key wrapping process as described in the section **Wrap an Initial ECC Public Key with the UFPK** of R11AN0496.

2.2.2 Generate the Wrapped OEM Root Public Key using OpenSSL and the SKMT CLI Interface

Once the Secure Key Management Tool (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 Public Key. This key can be presented to the MCU using the secure key injection process using RFP.

```
C:\Renesas\SecurityKeyMangementTool\cli>skmt.exe /genkey /ufpk
file="C:\RA8_FSBL\ufpk\ra8x1_ufpk.key" /wufpk
file="C:\RA8_FSBL\ufpk\ra8x1_ufpk.key_enc.key" /mcu "RA-RSIP-E51A" /keytype "OEM_ROOT_PK"
/filetype "rfp" /iv "55AA55AA55AA55AA55AA55AA55AA55AA" /key
file="C:\RA8_FSBL\openssl_keys\oem_root_private_key.pem" /output "
C:\RA8_FSBL\openssl_keys\oem_root_pk_cli.rkey"

Output File: C:\RA8_FSBL\openssl_keys\oem_root_pk_cli.rkey

UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F

W-UFPK: 00000000A7BF7EB27054D78E07C504291520678AA7BF7EB27054D78E07C504291520678A

IV: 55AA55AA55AA55AA55AA55AA55AA55AA

Encrypted key:
E71776A79F2BFF879CE3A434C5D0AEFBA934214518114AA89E7CAD10BD45D25623CE6710EC929971BAD200814
B6D633A670447B51347EA24EC7908C9C66AA933F7DD64E2DBDB1B831CED6E3B1347C69B
```

Note that this example `oem_root_pk_cli.rkey` does not match the example project included in this application project. So, do not use this key in the example boot processes described in this application project.

2.2.3 Generate the Wrapped OEM Root Public Key using the SKMT GUI Interface

In this section, the OEM Root Public Key pair provided in section 2.1.2 will be used to generate the wrapped OEM Root Public Key.

Launch **SKMT**, under the **Overview** page, choose **RA Family, RSIP-E51A Security Functions and Protected Mode** as shown Figure 17. Navigate to the **Wrap Key** tab in the SKMT, select the **Key Type** as **OEM Root public** and then provide the Wrapping Key (UFPK and W-UFPK).

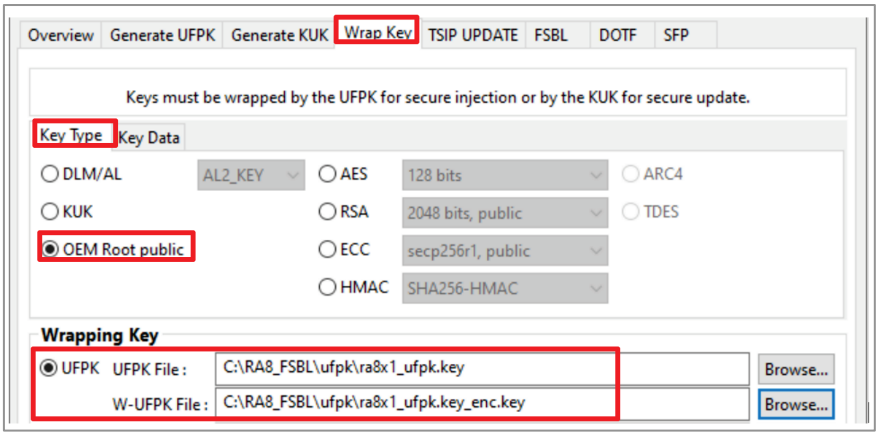


Figure 19. Wrap the OEM Root Public Key

Under the **Key Data** tab, provide the OEM Root Public Key.

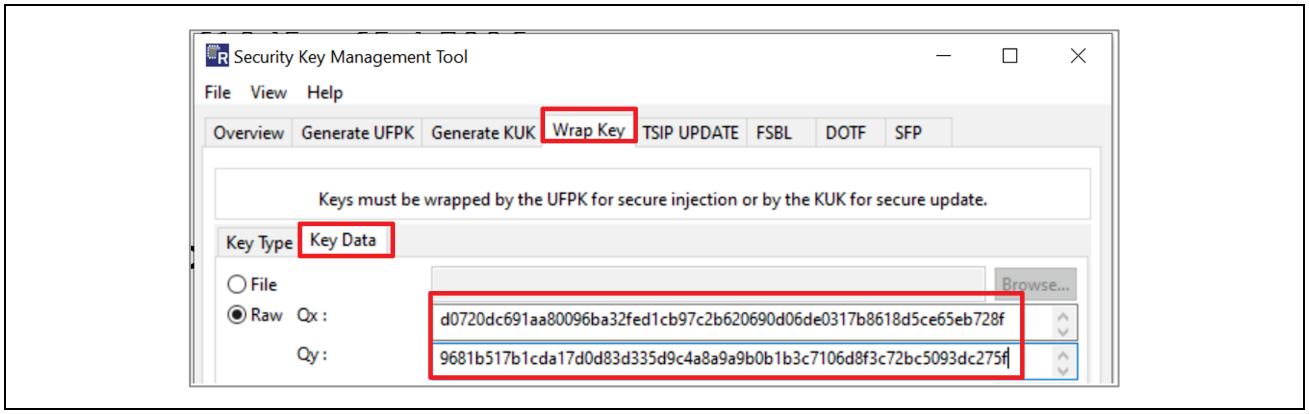


Figure 20. Provide the OEM Root Public Key

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 Public Key.

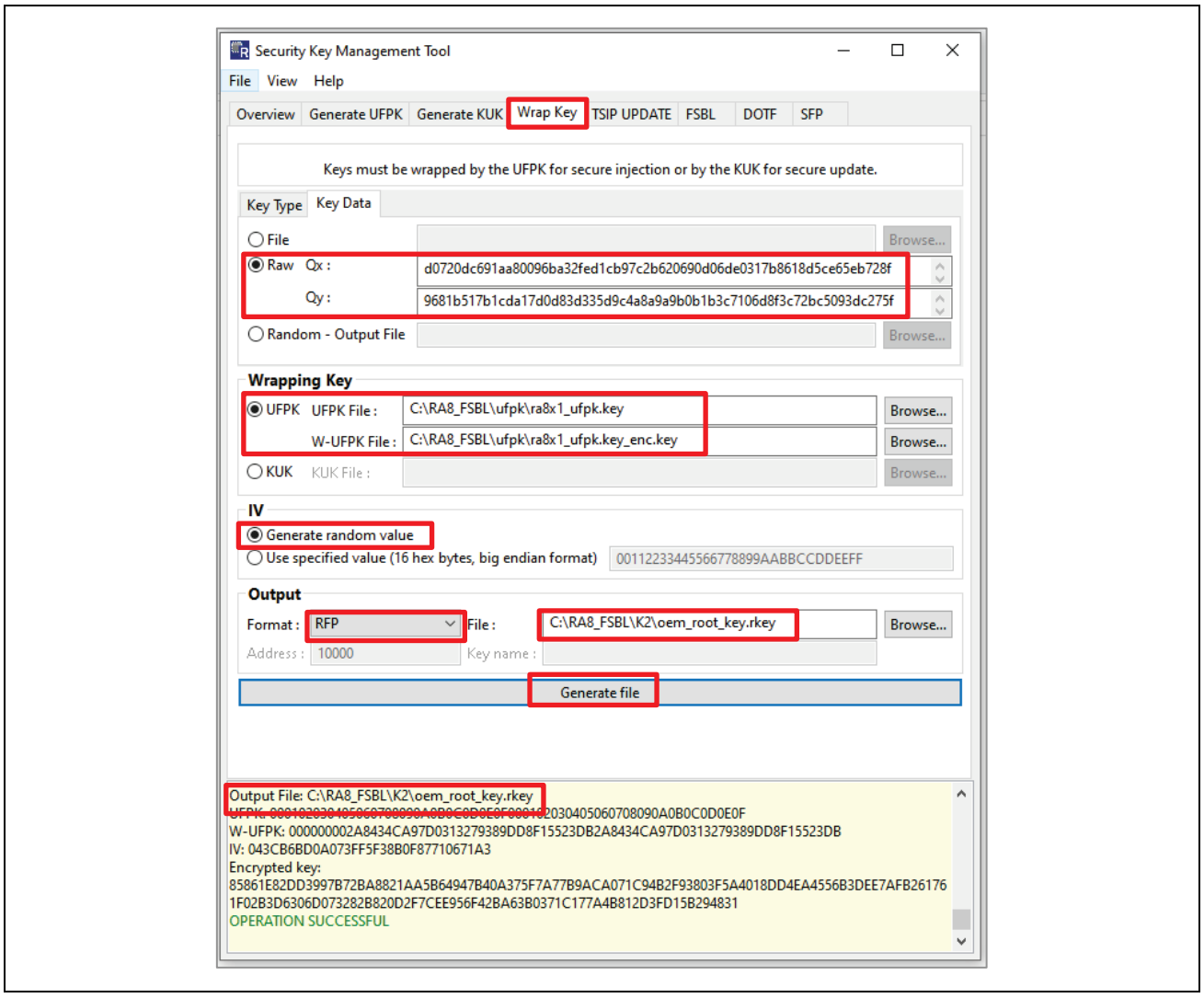


Figure 21. Generate the Wrapped OEM Root Public Key

This Wrapped OEM Root Public Key (`oem_root_key.rkey`) is provided to you to enable convenient evaluation of the system. It can be used if this same Root Public Key is used in the key and code certificate generation.

2.3 Prepare a Blinky Application with FSBL Enabled with HMAC-SHA256 Boot

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 example project `blinky_hmac_boot_ra8m1` is created by following section of “Create a New Project for Blinky” from the FSP User’s Manual with no RTOS support for RA8M1. By default, setting up the Trust Zone boundary is enabled in the Debug configuration setting. The FSBL property is configured based on the configuration in . 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 configures **P107** as **Output Initial Low** so it can be turned on to high by the FSBL upon a boot failure. Note that although the e2studio example project has the FSBL enabled, the FSBL will not be enabled after the application image is downloaded by e2studio. The FSBL will only be effective after the next power cycle.

For an MCU which has the FSBL already enabled in an early operation, run the **MCU Initialize** command to use the boot firmware to erase the existing FSBL and Trust Zone settings which resides in the option-setting memory. This can be achieved using the **Renesas Device Partition Manager** (RDPM) (which is natively installed with e2studio) or **Renesas Flash Programming** (RFP).

Once **e2studio** is launched, the RDPM can be activated by selecting **Run -> Renesas Debug Tools -> Renesas Device Partition Manager**. On EK-RA8M1, the default jumper setting has SCI boot mode enabled and the SCI signals are routed to the Debug interface, either SCI or SWD interface can be selected when working with RDPM. In the following screenshot, the SCI interface is selected.

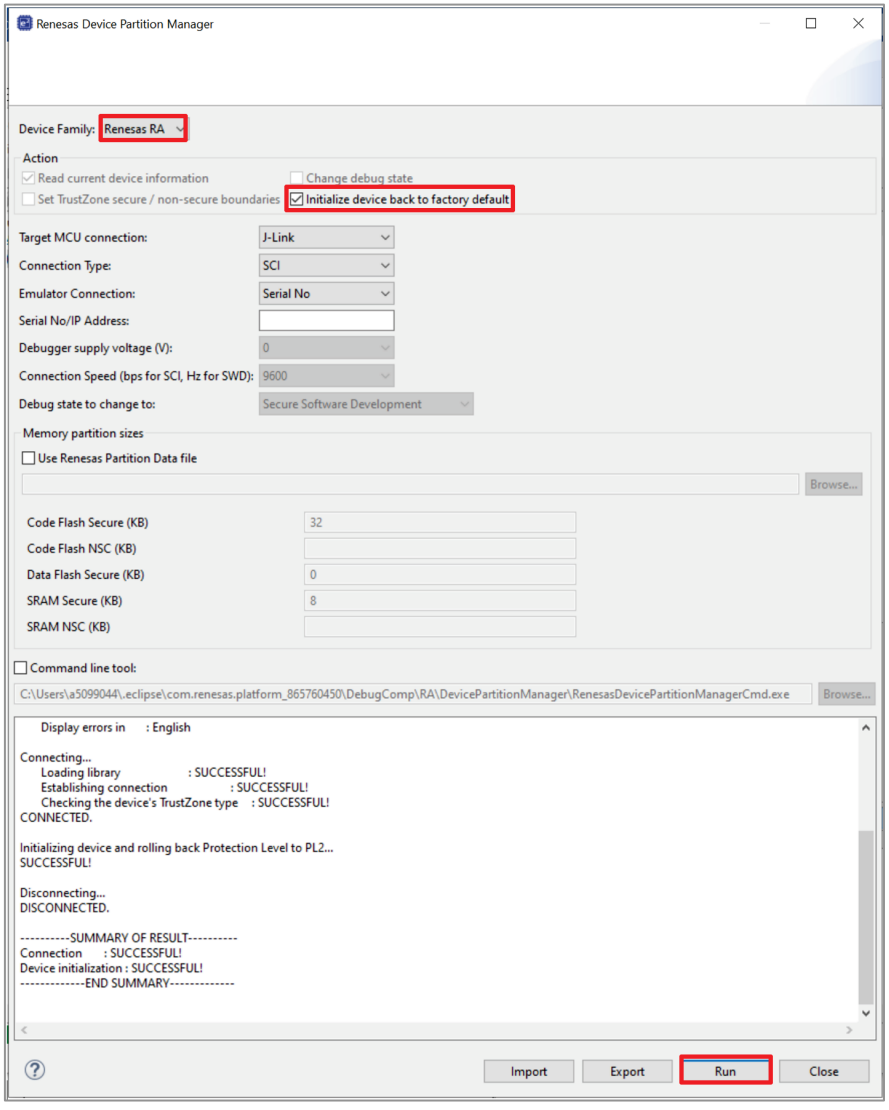


Figure 22. Using RDPM to Initialize the MCU

Figure 23 shows the command to choose to use RFP to Initialize Device.

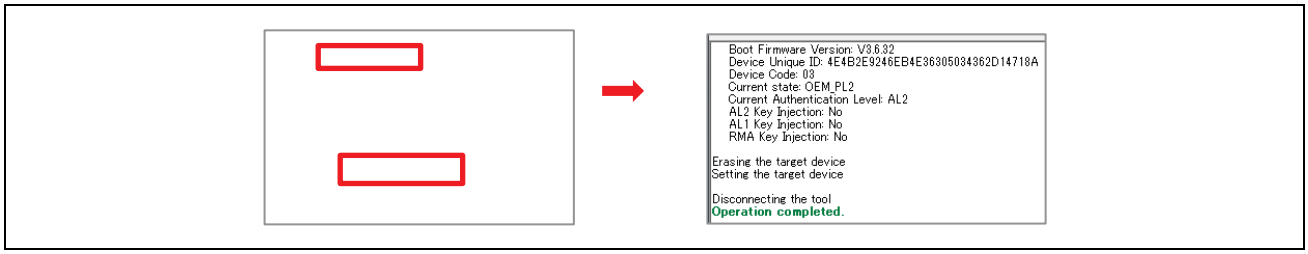


Figure 23. Using RFP to Initialize the MCU

After Initializing the Device, follow the below steps to build and run the included example project.

Import `blinky_hmac_boot_ra8m1` to a workspace, open the `configuration.xml` file and click **Generate Project Content**. Connect J-Link OB USB on J10 to the development's USB connection using a Micro USB cable, build and run the included blinky project (`blinky_hmac_boot_ra8m1`) to verify its functionality.

- The compilation result will be used in the section 2.4 for the creation of the code certificate.
- The three LEDs should be blinking.

```
arm-none-eabi-size --format=berkeley "blinky_hmac_boot_ra8m1.elf"
   text    data     bss     dec     hex filename
  11576      92    2220   13888    3640 blinky_hmac_boot_ra8m1.elf

16:53:29 Build Finished. 0 errors, 0 warnings. (took 396ms)
```

Figure 24. Blinky Project with FSBL HMAC Boot Enabled

Open Tera Term application and configure the Baud Rate to 115200.

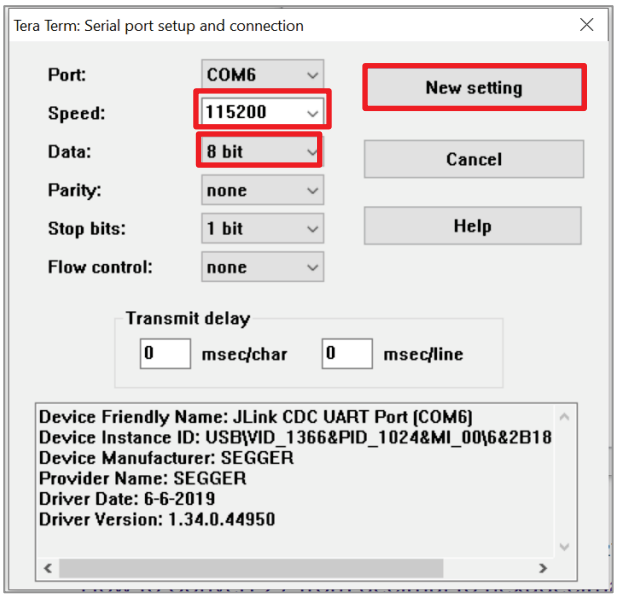


Figure 25. Configure the Tera Term

Click **New setting** and observe the following output on the terminal. Notice the output for the FSBL register settings, the boot measurement report and the HMAC digest. Refer to the register definition in the Hardware UM, the code certificate content (Figure 12) to understand the meaning of the output. Since the FSBL is not enabled yet, the `version_num_oem_bl` field will be 0s and the digest area will be 0xFFs.

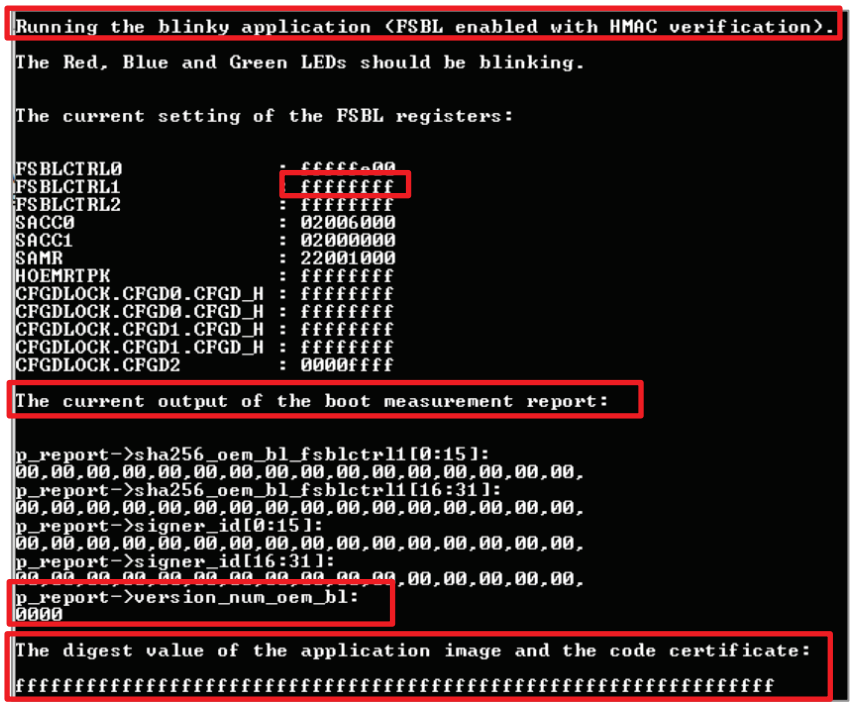
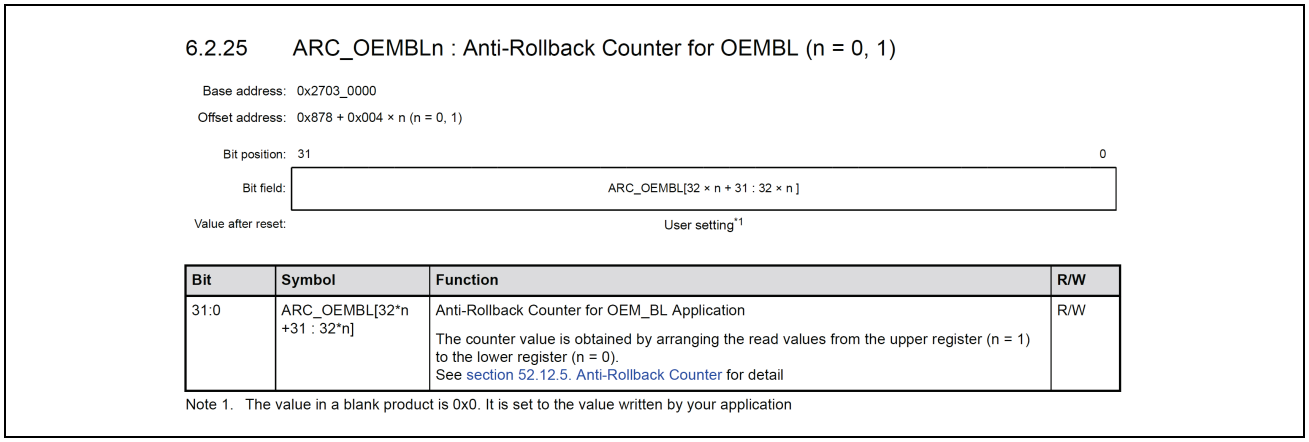


Figure 26. J-Link Console Output for HMAC Boot

When a Power-ON Reset is issued to the MCU, the FSBL will be activated and the blinky will not be booted anymore because the Root of Trust and code certificate are not yet programmed to the MCU. For a successful boot-up of the application, inject the OEM_ROOT_PK and present the key certificate and code certificate before a POR is issued. This process is explained in the subsequent sections. Section 2.6 provided details on booting an application with FSBL enabled.

2.4 Acquire the MCU OEM_BL Anti-Rollback Counter Value

As explained in section 1.3.3, the code certificate for HMAC boot should include a field for **Image Version** which ranges from 1 to 64. This Image Version number is permanently stored in a pair of option-setting memory registers called Anti-Rollback Counter for OEM_BL (refer to Figure 2) for the location of this register.



6.2.25 ARC_OEMBLn : Anti-Rollback Counter for OEMBL (n = 0, 1)

Base address: 0x2703_0000

Offset address: 0x878 + 0x004 × n (n = 0, 1)

Bit position:	31		0
Bit field:	ARC_OEMBL[32 × n + 31 : 32 × n]		
Value after reset:	User setting*1		

Bit	Symbol	Function	R/W
31:0	ARC_OEMBL[32*n +31 : 32*n]	Anti-Rollback Counter for OEM_BL Application The counter value is obtained by arranging the read values from the upper register (n = 1) to the lower register (n = 0). See section 52.12.5. Anti-Rollback Counter for detail	R/W

Note 1. The value in a blank product is 0x0. It is set to the value written by your application

Figure 27. Anti-Rollback Counter for OEMBL

Note that the Image Version number is not the value stored in these two registers, but an interpreted value. Reference section **52.12.5 Anti-Rollback Counter** for the operational details of this counter. Only boot firmware can write to this counter during production programming. Essentially, every successful application update increments the counter by setting one addition bit to 1. As an example, if the device has been updated 3 times, a number 7 (which is b 0111) will be stored in these two counters. These two 32-bit counters can be used for 64 application updates.

The application project includes a J-Link script that reads out this register value. Unzip `read_arc_oembl_jlink_script.zip` and double click on `ra8m1.bat` to read out the value in these two registers. Figure 28 shows an example output of running the J-Link script. Note that in this example, the register 0x27030878 (ARC_OEMBL0) holds a value of 0xFFFFFFFF and register 0x2703087C (ARC_OEMBL1) holds a value of 0x00000001. There are a total of 33 bits set to 1 for the ARC_OEMBL0 and ARC_OEMBL1 registers. Therefore, the next Image Version user can select should be equal or higher than 34 but equal or lower than 64.

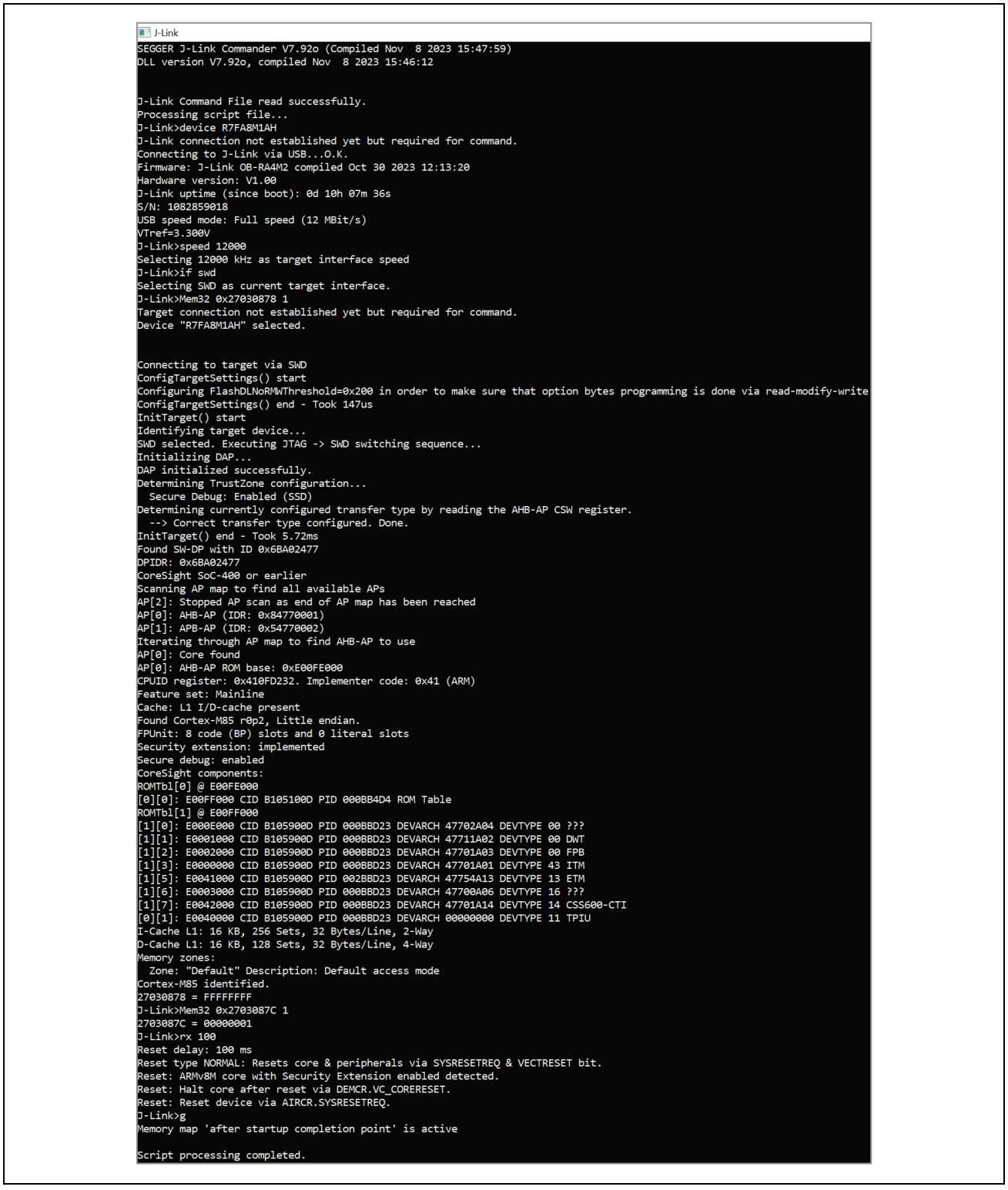


Figure 28. Read the ARC_OEMBL Register using J-Link Script

2.5 Generate Key Certificate and Code Certificate

As explained earlier, for HMAC-SHA256 based boot option, two sets of public and private ECC-P256 secp256r1 key pairs are needed to generate the key and code certificate. In this section, two demonstrations are provided. 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 interface

2.5.1 Using OpenSSL and SKMT Command Line Interface

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

- Input: the load address of the application. In the following example, this is the command option `/loadaddr`.
- Input: the size of the OEM_BL. In the following example, this is the command option `/oembl_size`.
- Input: the version of the application. In the following example, this is the command option `/ver`. Note that for an MCU that has previously booted from an old version of the application via FSBL, the next version needs to be a higher version. The version number will be included in the code certificate generated (refer to Figure 12). In this example, it is set to 3, which can be used if the ARM_OEMBLn value with 0, 1, or 2.
- Input: the code flash size of the device to be used. In the following example, this is the command option `/cfsize`.
- Input: the application binary. In the following example, this is the command option `/oembl`.
- Input: the OEM_ROOT_SK. In the following example, this is the command option `/oemroot_private`.
- Input: the OEM_BL_ROOT_SK. In the following example, this is the command option `/oembl_private`.
- Output: the key certificate and code certificate. In the following example, these are the command options for `/output_codecert` and `/output_keycert`.

Open a command window and navigate to the `\SecurityKeyMangementTool\cli` folder and provide the following command to generate the key and code certificate. The file locations should be adapted as desired.

```
C:\Renesas\SecurityKeyMangementTool\cli>skmt.exe /gencert /mode "signature" /loadaddr "02000000"
/oembl_size "8000" /cfsize "2000000" /ver "3" /oembl
"C:\RA8_FSBL\fsp_v510\blinky_hmac_boot_ra8m1\Debug\blinky_hmac_boot_ra8m1.srec"
/oembl_private file="C:\RA8_FSBL\openssl_keys\oem_bl_private_key.pem" /oemroot_private
file="C:\RA8_FSBL\openssl_keys\oem_root_private_key.pem" /output_codecert
"C:\RA8_FSBL\openssl_keys\blinky_hmac_openssl_ccert.bin" /output_keycert
"C:\RA8_FSBL\openssl_keys\blinky_hmac_openssl_kcert.bin"

Output File: C:\RA8_FSBL\openssl_keys\blinky_hmac_openssl_kcert.bin

Output File: C:\RA8_FSBL\openssl_keys\blinky_hmac_openssl_ccert.bin
```

2.5.2 Use NIST ECC Key Pair and SKMT GUI Interface

This section uses the two sets of NIST CAVP ECC secp25r1 Key Pairs provided in section 2.1.2 to demonstrate the certificate generation. Both the private and public key parts of the two key pairs should be provided to SKMT to generate the key and code certificate.

Launch SKMT GUI. On the **Overview** page, select **RA Family, RIP-E51A Security Functions and Protected Mode**.

Next select the .srec generated in section 2.1 and define the version of the application.

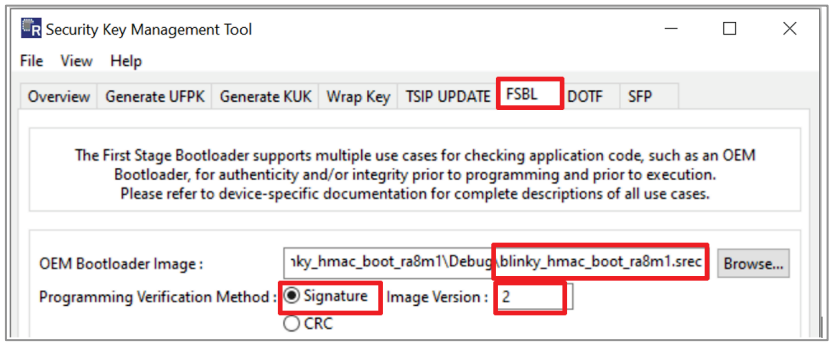


Figure 29. Select the Application and Version Number

The **Image Version** definition defined in Figure 29 will be stored in the code certificate. As explained in section 1.3.3, the allowed version number is 1 to 64 and update image needs to have a higher version than the current programmed image. So, for the first time when the FSBL HMAC based boot is exercised on a new MCU, the Image Version can be set to 1.

Stay on the **FSBL** page, configure the **OEM Root Keys**.

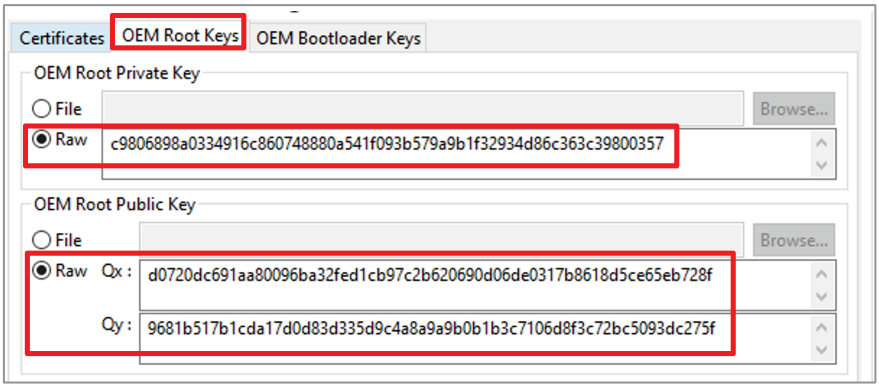


Figure 30. Provide the OEM Root Keys

Next, provide **OEM Bootloader Keys.**

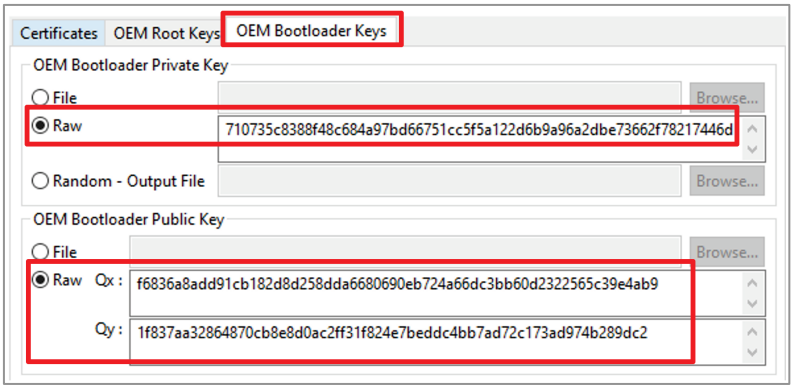


Figure 31. Provide the OEM Bootloader Keys

Next, navigate to the **Certificates** page and define the name and location of the key and code certificate and then click **Generate File(s)**. The key certificate and code certificate will then be generated.

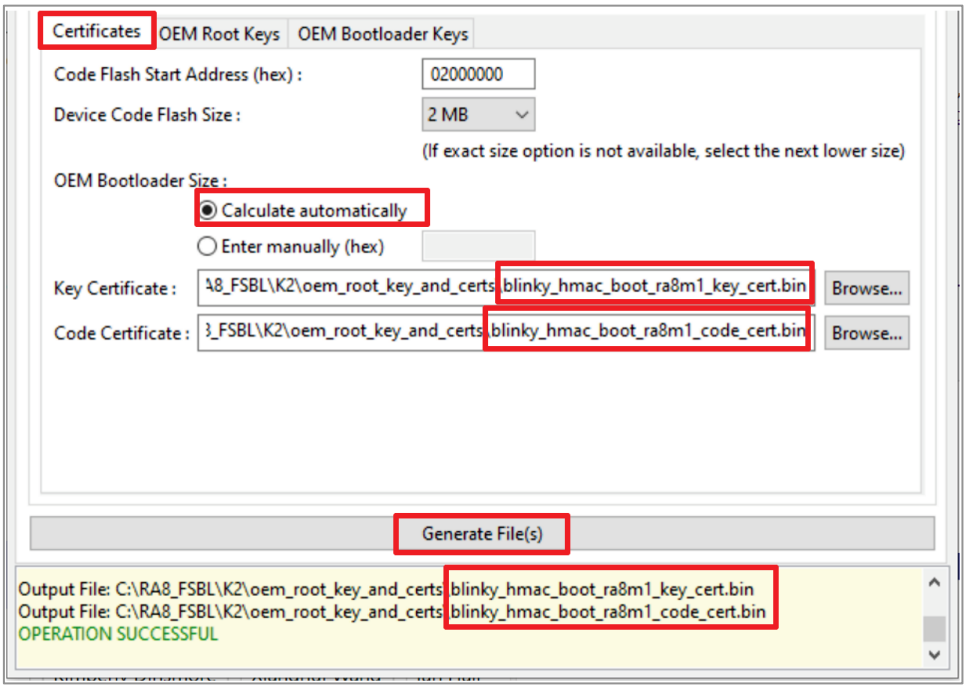


Figure 32. Generate the Key and Code Certificate

The SKMT can also internally output NIST vectors. This is achieved by selecting **Random – Output File**.

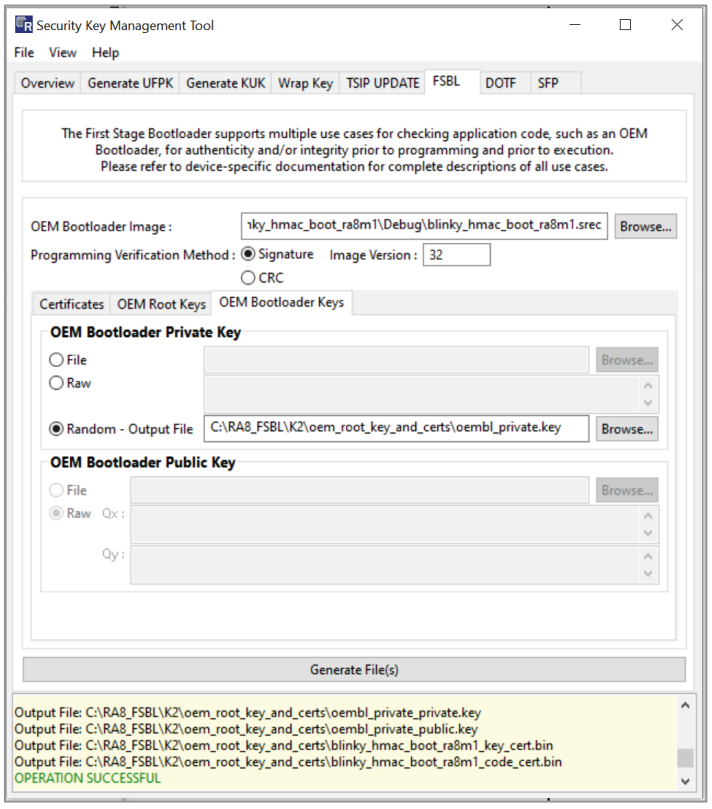


Figure 33. Using SKMT Internally Stored NIST Test Vectors

The certificates generated from either Figure 32 or Figure 33 can be used for the rest of the operations described in this application project.

Note that SKMT v1.05 has a bug which prevents it from generating code certificate with version 64. The following error will be printed when version 64 is requested.

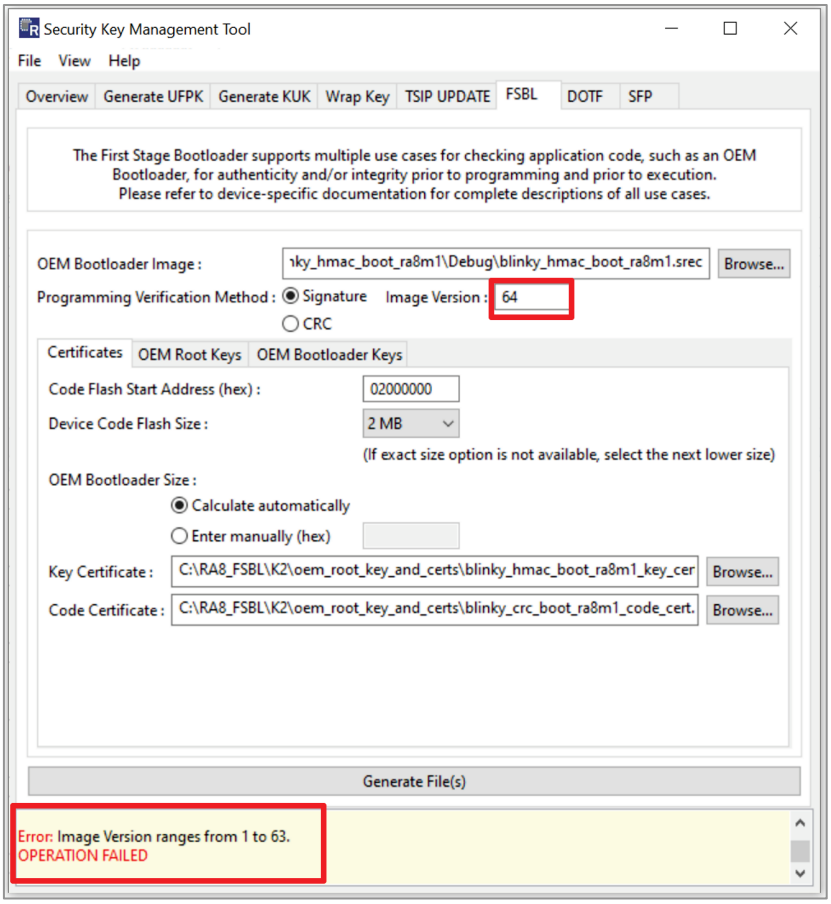


Figure 34. Bug with SKMT v1.05 for Code Certificate Generation with Version 64

2.6 Demonstrate the Authenticated Production Programming and HMAC Boot

Download `rfp_ra8m1.zip` and extract the content to a local folder. Launch **RFP** and open the included `rfp_ra8m1.rpj` project. In this example RFP project, the SWD interface is selected as the boot interface, so the USB Debug cable can be used to access the boot mode.

If FSBL has been enabled in an earlier application or TrustZone® boundary has been set up previously, the **Initialize Device** (refer to Figure 23) command should be run prior to proceed to the following sections. This is needed so the previous FSBL configurations and TrustZone configurations can be erased.

Next, the RFP to download the application to the MCU and provide the credentials to authenticate and boot the application. Through the flash option programming, some credentials will be injected to the MCU as explained in section 1.3.4. Ensure the following **Operation Settings** are configured to erase the previous application, download the application and program the Flash Options.

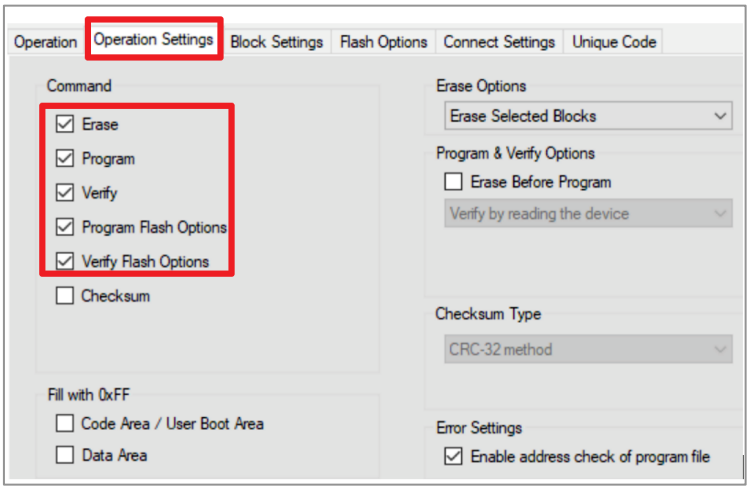


Figure 35. Configure the Operation Settings

Select the application file to program to the MCU.

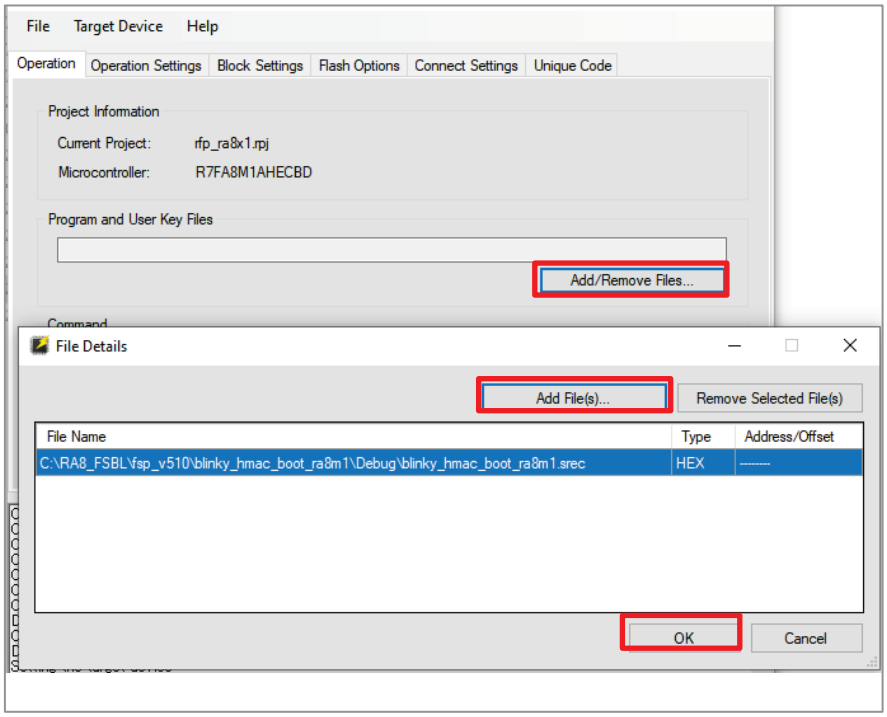


Figure 36. Select the Application Project

Set up the **Flash Options** to configure the **OEM Root Public Key**, **Code Certificate** and **Key Certificate**.

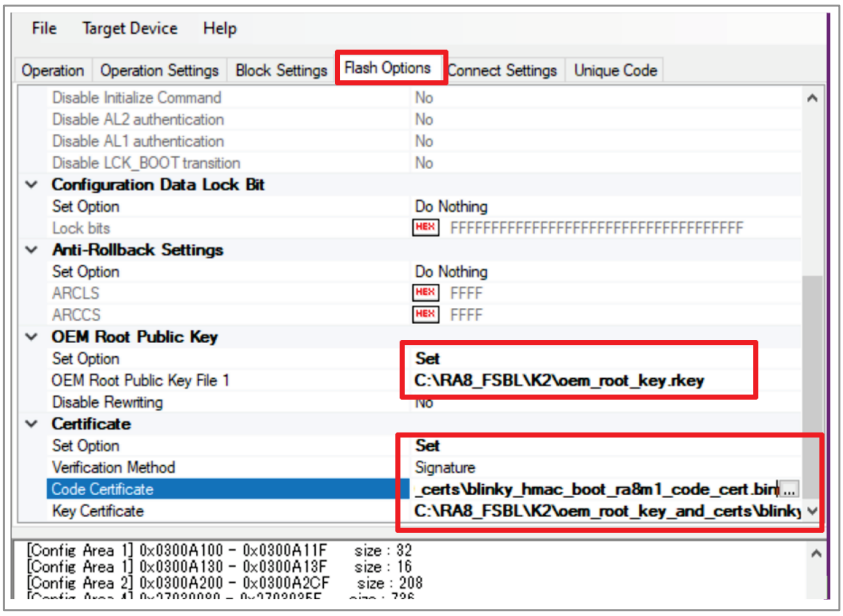


Figure 37. Configure the OEM Root Key and the Certificates

Now navigate to the **Operation** tab and then click the **Start** button. RFP will read the SACC0 register value from the SREC and program the Code Certificate at the indicated memory address. The FSBL will compare the version number of the code certificate, if the version number is higher, the authentication of the OEM_BL will be proceeded. If the authentication is successful, the OEM HMAC digest will be programmed to the area after the code certificate. On the next power cycle, the FSBL will authenticate the application using the stored digest and the FSBL calculated digest, if the comparison is successful, the new application will be booted. The three LEDs should now be blinking.

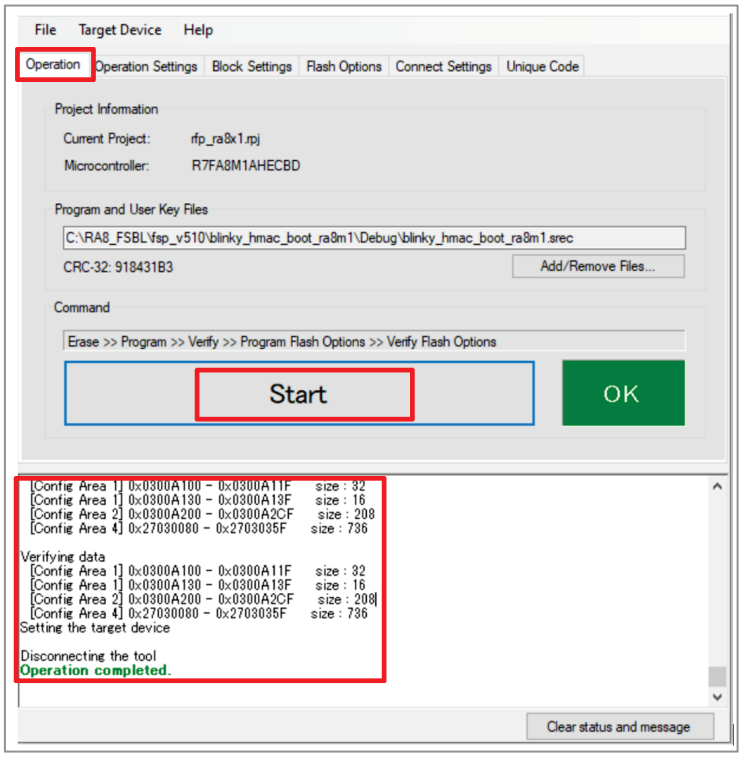


Figure 38. Successful Update of the Application based on HMAC SHA2-256 Verification

Upon successful boot, the J-Link 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.

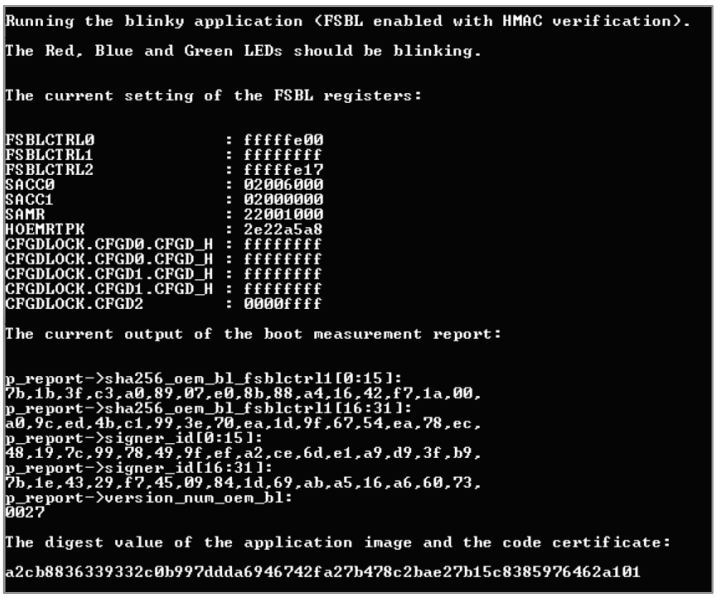


Figure 39. J-Link Console Output for HMAC Boot

To demonstrate a failure on the HMAC boot is easy. First initialize the Device as show in Figure 23 and then perform the steps in section 2.6 except provide a wrong application image, for example the binary image of the CRC boot code or an updated the HMAC boot code. In either case, the RFP programming operation in Figure 38 will fail and pin P107 will assume a high level, thus turns on the red led.

3. Production Programming with CRC Check and CRC Boot Demonstration

When using CRC Boot, CRC32 integrity check will be performed during programming and normal execution after MCU reset.

3.1 Prepare a Blinky Example Project using FSBL with CRC Boot

The included blinky example project `blinky_crc_boot_ra8m1` is created by updating the FSBL Execution Mode to **CRC boot and report measurement** as shown in Figure 40. The same error report pin is used (P107). Additionally, the J-Link Console message is updated to indicate the CRC boot mode.

Import `blinky_crc_boot_ra8m1` to a workspace, open the `configuration.xml` file and click **Generate Project Content**. Build and run the included blinky project (`blinky_crc_boot_ra8m1`) to verify its functionality.

- The compilation result will be used in the section 3.2 for the creation of the code certificate.
- The three LEDs should be blinking.

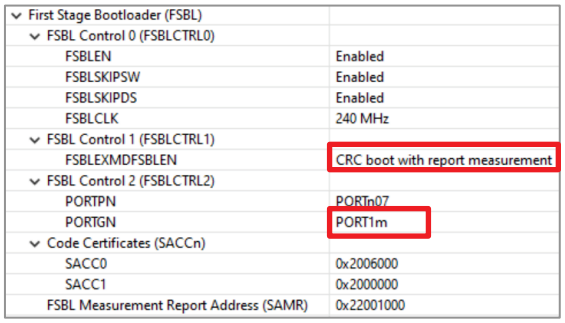


Figure 40. Configure FSBL for CRC Boot

Configure the Tera Term as in Figure 25 and observe the following Tera Term output. Note that since the code certificate is not programmed yet, the output for the CRC is all 0xFFs. In addition, since the Image Version information is not used in a CRC boot, `version_num_oem_bl` output is all 0s.

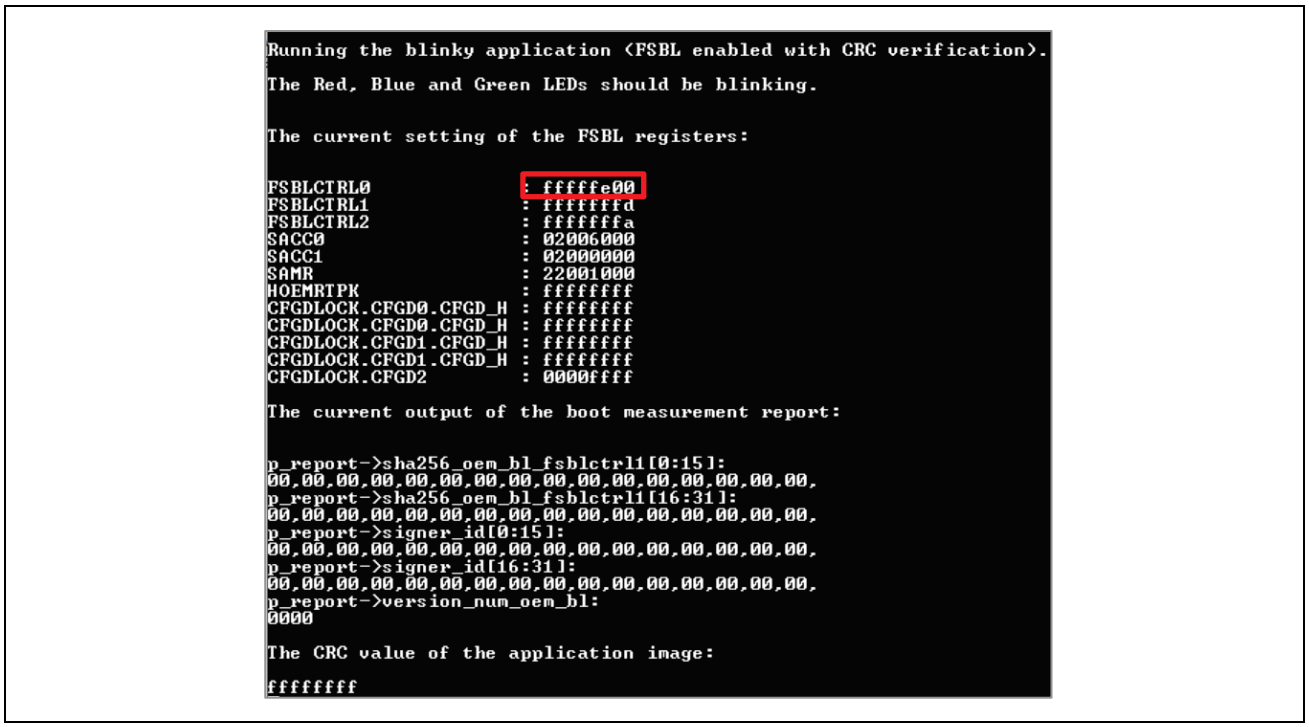


Figure 41. RTT Viewer Output for CRC Boot

3.2 Create the Code Certificate

The Code Certificate for CRC boot must contain the expected CRC of the OEM application. Upon MCU reset, if an integrity check is being performed, the FSBL will calculate the CRC of the application and compare it to the value in the stored Code Certificate.

Create the Code Certificate for CRC Boot using SKMT CLI

Open a command line and navigate to the SKMT `\cli` folder and use the following command to generate the code certificate based on the project created in section 3.1.

```
C:\Renesas\SecurityKeyMangementTool\cli>skmt.exe /gencert /mode "CRC" /loadaddr "02000000"
/oembl_size "8000" /cfsize "200000" /oembl
"C:\RA8_FSBL\fsp_v510\blinky_crc_boot_ra8m1\Debug\blinky_crc_boot_ra8m1.srec"
/output_codecert
"C:\RA8_FSBL\K2\certificates\blinky_crc_boot_ra8m1_code_cert.bin"

Output File: C:\RA8_FSBL\K2\certificates\blinky_crc_boot_ra8m1_code_cert.bin
```

Create the Code Certificate for CRC Boot using SKMT GUI

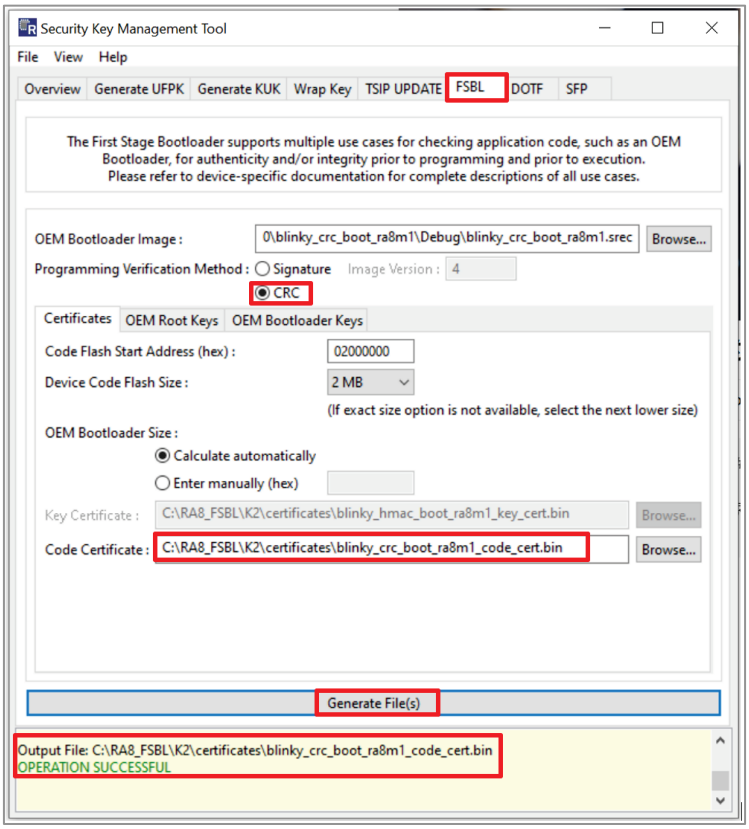


Figure 42. Create CRC Boot Code Certificate

When code certificate is generated with CRC specified, the CRC value is output as a dummy value for the Signer ID field in the code certificate. When the measurement report for CEC verification is created, the Signer ID uses 8 times of the duplicated CRC value. CRC32 is 4byte (32 bits) and Signer ID (SHA256) is 32 bytes, the CRC value is written 8 times in Signer ID field.

3.3 Demonstrate the CRC Boot of the Application using RFP

Launch **RFP** and open the included `rfp_ra8m1.rpj`. We will download the application to the MCU and provide the credentials to authenticate and boot the application through the MCU's SWD boot interface.

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

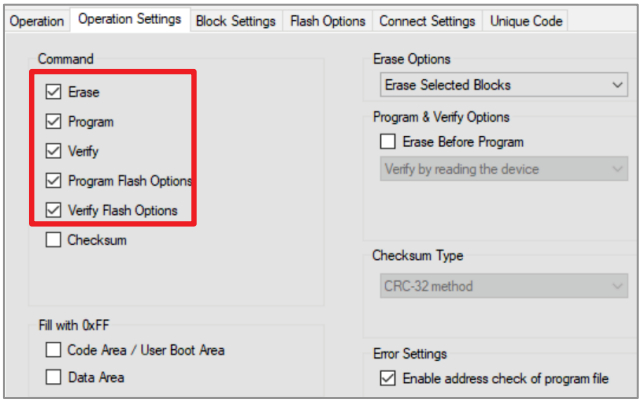


Figure 43. Configure the Operation Settings before Downloading the Application

Select the application file to program to the MCU.

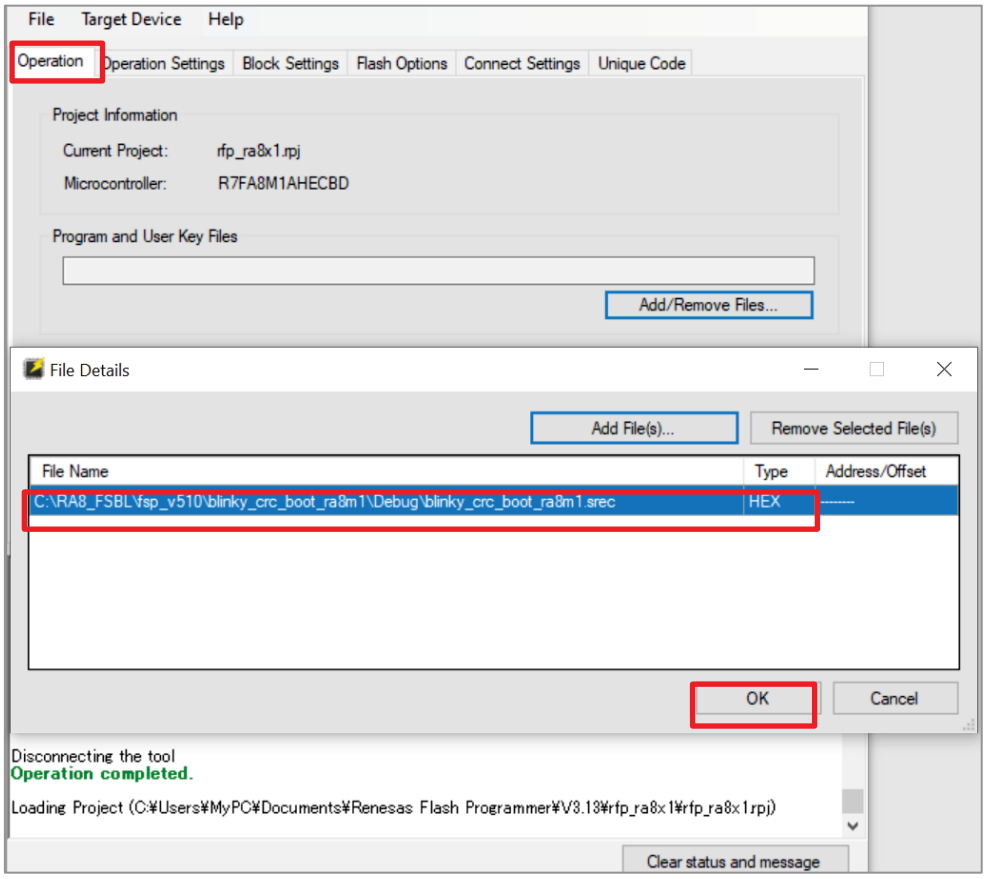


Figure 44. Select the Application Project

Click **OK** and navigate to the **Flash Options** page to configure the **Code Certificate**.

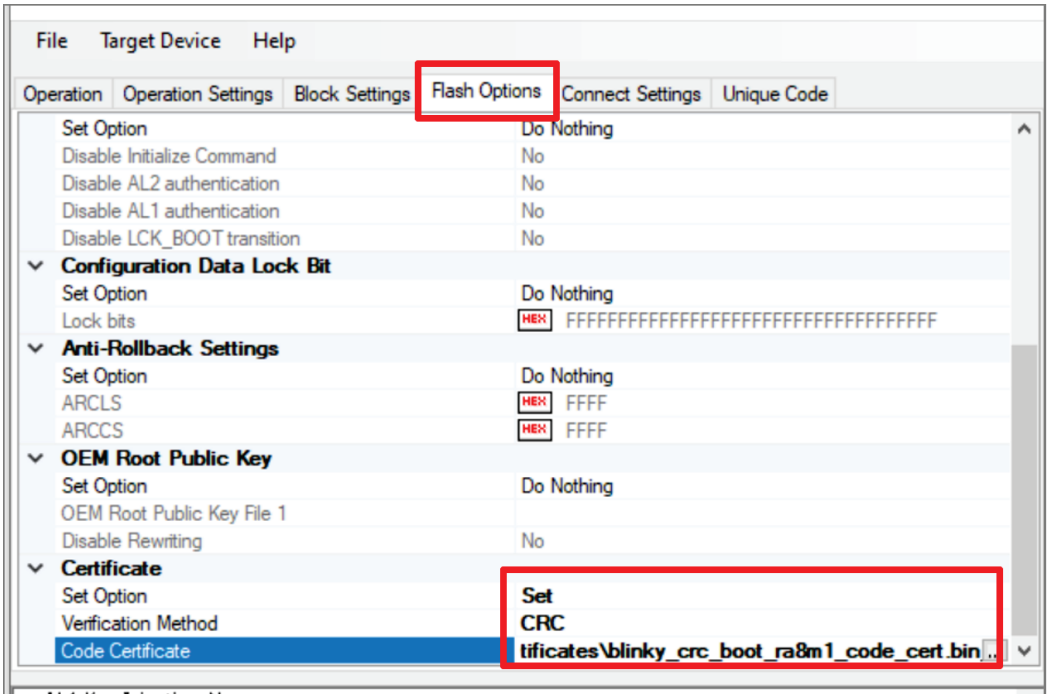


Figure 45. Configure the CRC Code Certificate

Next, navigate to the **Operation** tab and select **Start**, the application will be programming as well as the code certificate. The three LEDs should now be blinking.

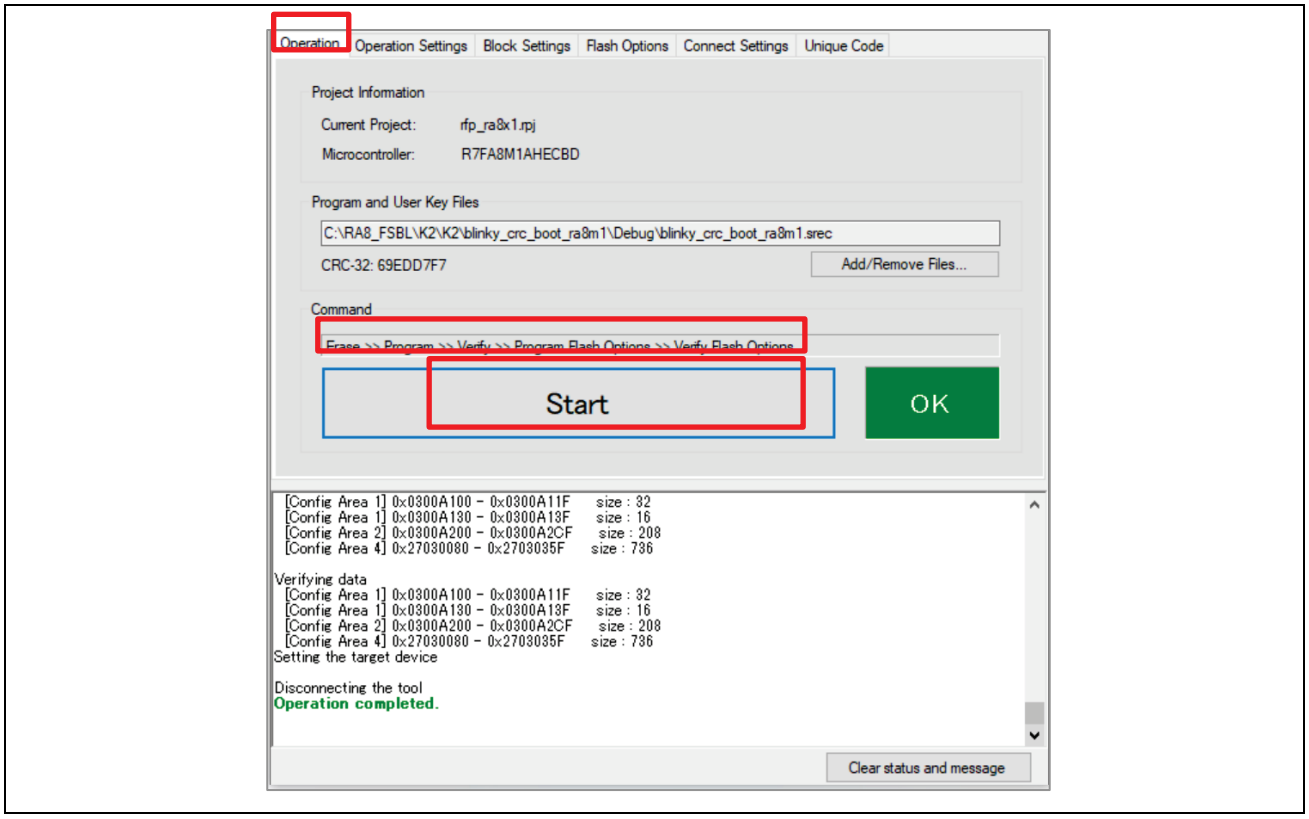


Figure 46. Downloading the CRC Boot Application

On the J-Link console, a similar message as the following will be printed. Note that the CRC value is repeated four times as the `signer_id`.

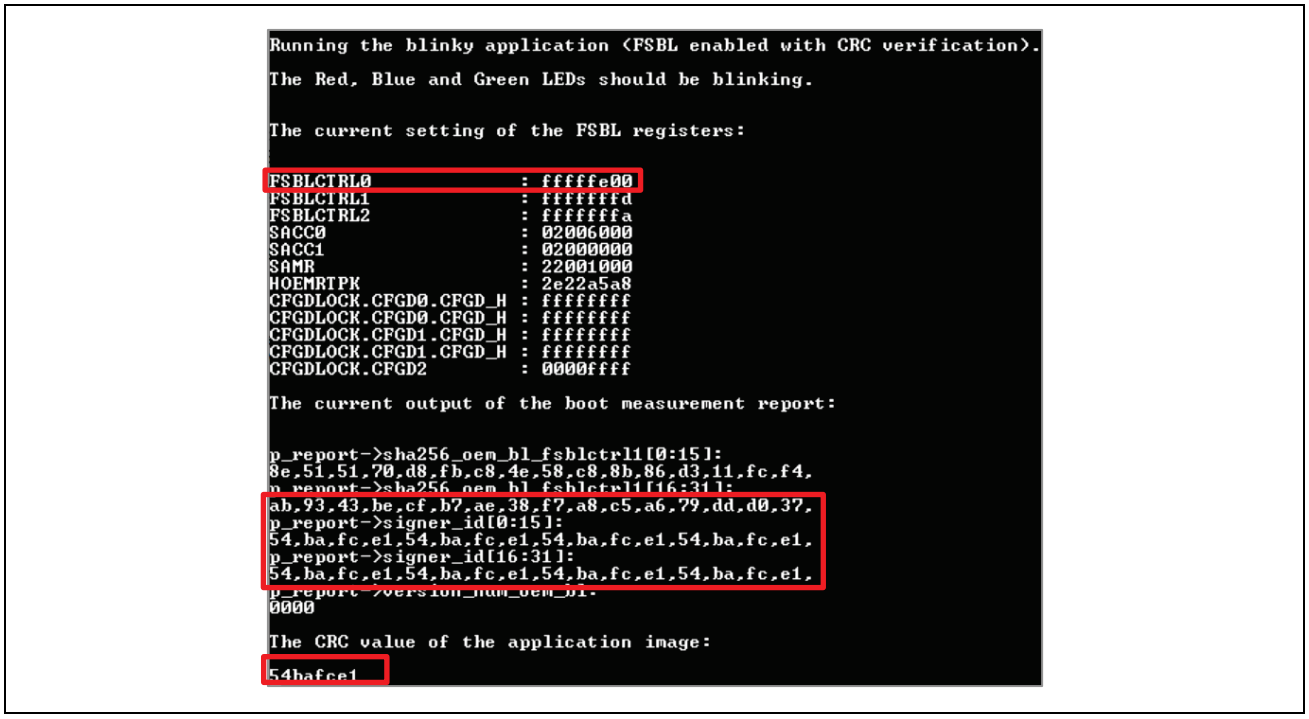


Figure 47. CRC Boot is Successful

Note that the CRC32 calculated from RFP differs from the CRC32 calculated from SKMT. The RFP calculates the CRC32 of the entire file. For example, for a .srec file, all the record fields are included in the CRC32 calculation. However, for SKMT, the CRC32 calculation only includes the binary value of the OEM_BL.

To demonstrate a failure on the CRC boot is easy, first initialize the Device as show in Figure 23 and then perform the steps in section 3.3 except provide a wrong binary image , for example provide the binary image of the HMAC boot code or perform a minor update the CRC boot code. In either case, the RFP programming operation in Figure 46 will fail and P107 assume a high level which will turn on the red LED.

4. Appendix

4.1 Usage Note with TrustZone® Project

Assume the set of Trustzone projects is the following:

- `blinky_s` for the secure application
- `blinky_ns` for the matching non-secure application

When using FSBL with a set of TrustZone projects with HMAC boot, generating the ECC Key Pairs and OEM Root Key can be achieved in the same ways as described in sections 2.1 and 2.2. The same script described in section 2.4 can also be used to retrieve the ARC_OEMBL register values.

The operations in section 2.3, 2.5, and 2.6 should be adjusted to use FSBL with a set of TrustZone projects. The .rdp file generated from the IDE includes the TrustZone boundary based on the secure project flash and SRAM usage side and will be used for the setting up the TrustZone boundary.

- In section 2.3, the FSBL should be enabled in `blinky_s` project. Ensure `SACCx` is located outside the flash area used by `blinky_s` and `blinky_ns`. When using e2studio IDE, if the following selection is enabled, the IDE will pick up the generated blinky_s.rdp file to program the TrustZone boundary.

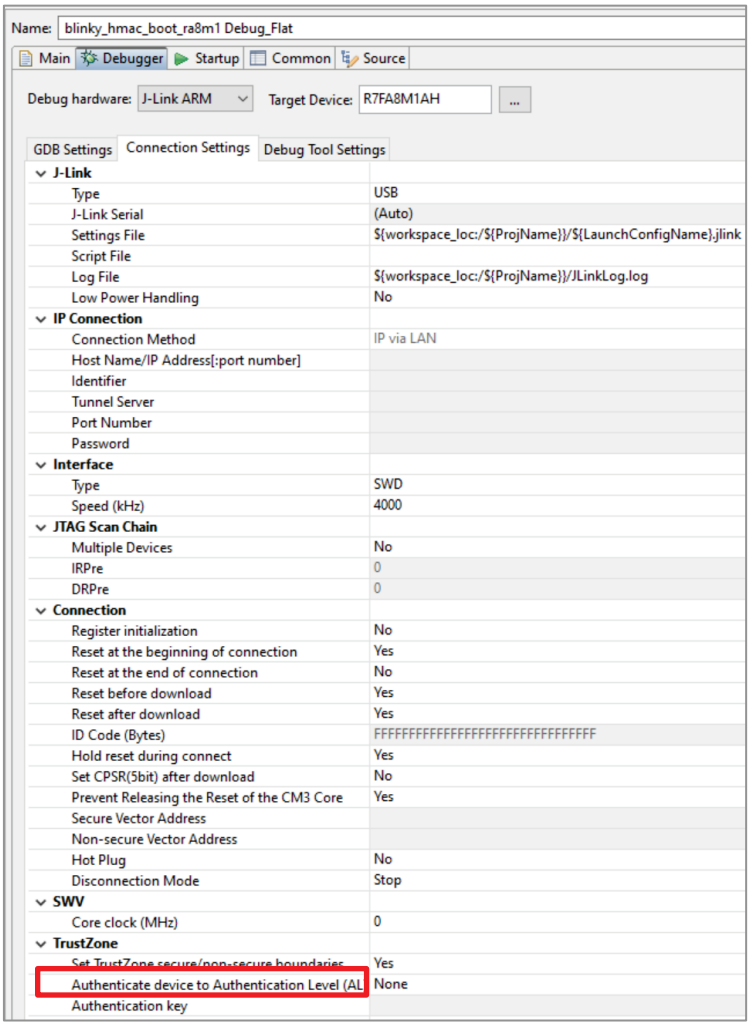


Figure 48. Enabled Programming of TrustZone® Boundary in e² studio

When using Keil and IAR for TrustZone based application development, the RDPM should be manually launched. The RDPM should be configured to use the generated .rdp file to program the TrustZone boundary as shown in Figure 49.

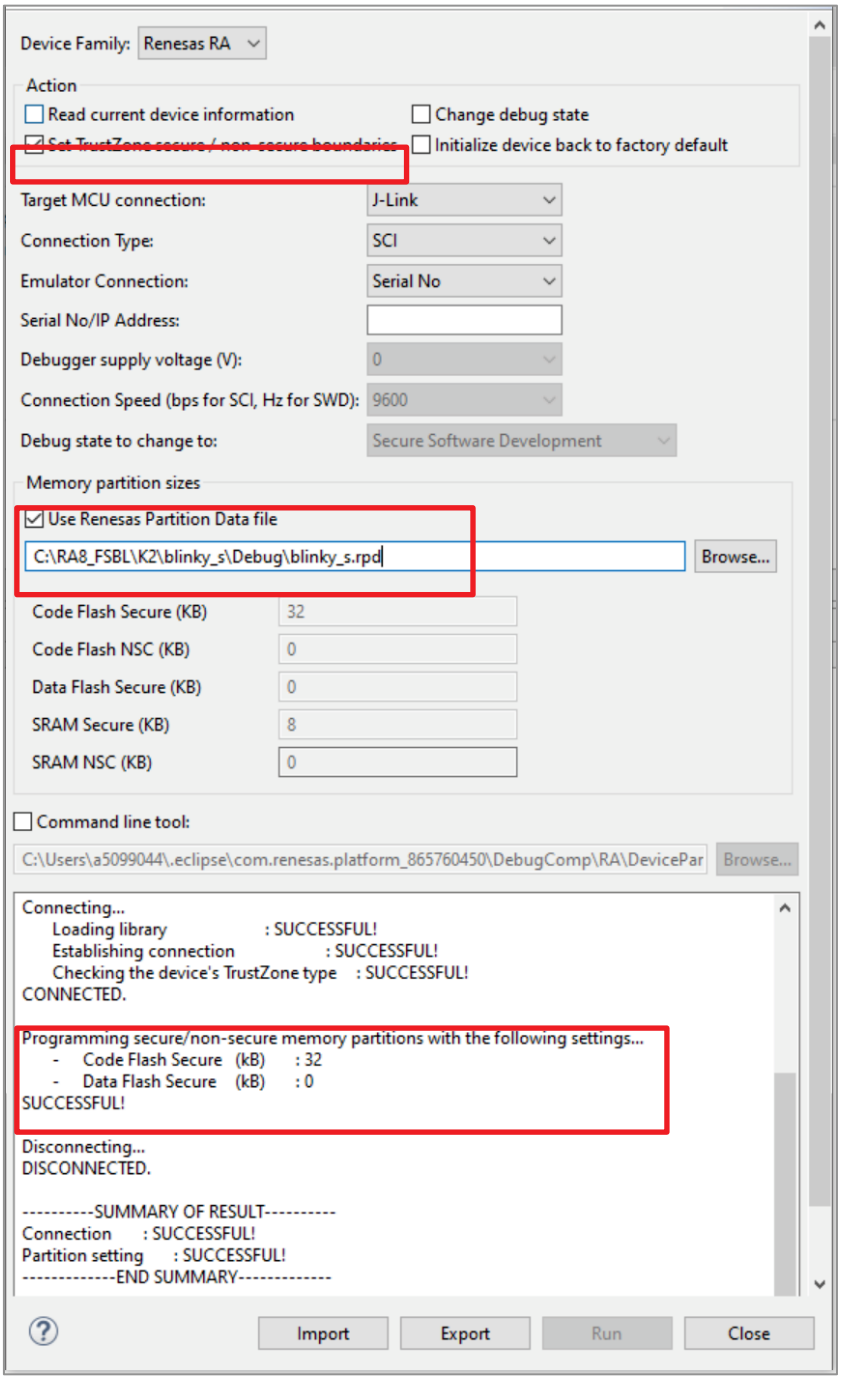


Figure 49. Using RDPM to Set Up the TrustZone® Boundary

- In section 2.5, the compiled binary from `blinky_s` should be used to generate the key and code certificates. The non-secure application is not used in the certificate generation.
- In section 2.6, the secure application and the non-secure application should be programmed. In addition, the TrustZone® boundary should be set up using the `blinky_s.rdp` file generated from the secure project. So, in addition to provide the applications similarly as in Figure 36, users should set up the TrustZone boundary by selecting the `blinky_s.rdp` file under the RFP **Flash Options** page as shown in Figure 50 in addition to all other settings needed as originally documented in section 2.6.

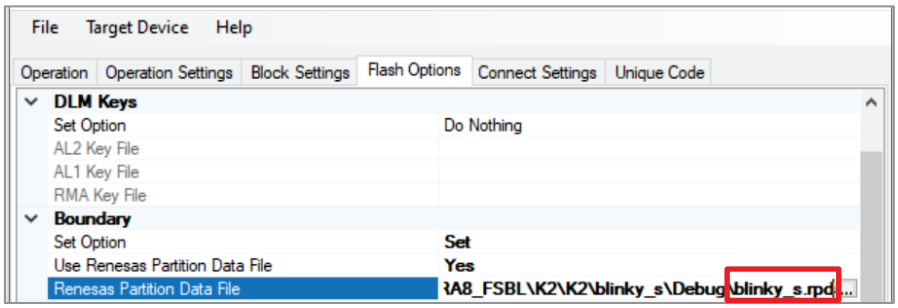


Figure 50. Set up the TrustZone® Boundary using RFP

When using FSBL with a set of TrustZone projects with CRC boot, uses need to adjust the operations in section 3.1, 3.2 and 3.3 in similar ways as the HMAC boot except only the code certificate will be generated.

For general usage of TrustZone projects, users can refer to application projects RA8 MCU Quick Design Guide R01AN7087 and Security Design using TrustZone R11AN01467.

4.2 Debug RFP Usage Errors

Users should refer to RFP User's Manual to understand the error code output from the RFP operation. Document information is provided in the References section. For your reference, the following error code indicates the Image Version number is not set up properly.

Figure 51. Image Version Number Error

5. References

1. Renesas RA Family Secure Key Injection Application Project (R11AN0496)
2. Renesas RA Family User's Manual: Hardware (R01UH0994)
3. Renesas Secure Key Management Tool User's Manual (R20UT5349)
4. Renesas Flash Programmer Flash memory programming software User's Manual (R20UT5352).
5. Renesas RA Security Design using TrustZone with IP Protection (R11AN0467)
6. Renesas RA8 MCU Quick Design Guide (R01AN7087)

6. Website and Support

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

EK-RA6M4 Resources	renesas.com/ra/ek-ra8m1
RA Product Information	renesas.com/ra
Flexible Software Package (FSP)	renesas.com/ra/fsp
RA Product Support Forum	renesas.com/ra/forum
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar.04.24	—	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/.