RA8M1 MCU Group

Security Manual

Introduction

The RA8M1 MCU Group incorporates many features that can be used to protect the content and operation of the MCU when it is integrated into an end product. This document describes these features and provides general recommendations for their use. Associated Application Notes and other applicable documents that provide more details are listed for reference.

Since the RA8M1 is a general purpose MCU, this Security Manual cannot offer specific guidance for all possible use cases, nor can it provide system-level security guidance. It is highly recommended that the application developer conduct a security analysis of the system into which the RA8M1 is integrated and create a threat model and security policy for the system and for the RA8M1, for example, using the ISO/SAE 21434 Threat Analysis and Risk Assessment (TARA) approach. It is the application developer's responsibility to decide what RA8M1 security features are necessary for the end product and to implement them appropriately.

Although this Security Manual describes the best secure practices for configuring the RA8M1 at the time of its release, the microcontroller threat landscape is constantly changing and evolving. Please refer to the Notice at the end of this document.

Contents

1. Introduction

1.1 Overview

This application note describes the security-related features of the RA8M1 MCU Group and offers guidance on how to use these features to create a product that has the appropriate level of cybersecurity based on the threat model for the intended application.

NOTE: Ultimately, it is up to the discretion of the application developer to determine how to design the security architecture of the end-product and what security features to employ. Critical items relating to the proper functioning of the MCU and possible security threats are noted in red sections.

1.2 Security Features

The RA8M1 MCU provides the following security-related features:

- MCU product and chip-unique identification
- Hardware-enforced isolation using Arm® TrustZone® technology
- The RSIP-E51A integrated security engine, providing cryptographic functions and secure key handling
- Immutable memory
- First Stage Bootloader
- Secure external storage with Decryption On-The-Fly
- Various tamper detection features
- Device Lifecycle Management and authenticated debug
- Secure Factory Programming

The RA Flexible Software Package (FSP) is the Renesas software platform available for use with the RA8M1 MCU Group. The FSP provides support for security solutions that are often required to secure an end-product, including:

- Creation of a cryptographic identity
- Secure boot
- Secure firmware updates
- Secure internet connectivity

1.3 References

The following documents are referenced in this application note. These documents are available from the Renesas website, www.renesas.com.

Reference	Document Name	Document Number or URL
[RA8M1 User Manual]	RA8M1 Group User's Manual: Hardware	R01UH0994
[FSP User Manual]	RA Flexible Software Package Documentation	RA Flexible Software Package Documentation: Introduction (renesas.github.io)
[Boot Firmware]	Renesas Boot Firmware for the RA8M1 MCU Group	R01AN7140
[SP800-90B Test Report]	NIST SP800-90B Entropy Assessment Report for RA8M1	R01AN7269

1.4 Additional Guidance Documentation

The following additional guidance documents are referenced in this application note. Please note that due to documentation update cycles, the RA8M1 MCU Group may not be explicitly referenced in the latest version of these documents. These documents are available from the Renesas website, www.renesas.com.

Reference	Document Name	Document Number
[Bootloader Basic]	RA8 Basic Secure Bootloader Using MCUboot and Internal Code Flash	R11AN0909
[Cloud Connectivity]	RA AWS MQTT/TLS Cloud Connectivity Solution - Ethernet	R11AN0605
[Cloud OTA]	RA AWS Cloud Connectivity and Firmware Update OTA on CK-RA6M5 v2 with Ethernet	R11AN0915
[Design with TrustZone]	Security Design using Arm TrustZone – Cortex M85	R11AN0897
[Device Identity]	Device Identity with RSIP and TrustZone	R11AN0979
[DLM]	Device Lifecycle Management for RA8 MCUs	R11AN0785
[DOTF]	Application Design using RA8 Series MCU Decryption On The Fly	R11AN0773
[FSBL]	Application Design using RA8 First Stage Bootloader	R11AN0774
[Plaintext Key Injection]	Injecting Plaintext User Keys	R11AN0473
[RSIP Modes]	Renesas Security Engine Operational Modes	R11AN0498
[Secure Factory Programming]	RA8 Secure Factory Programming	R01AN7516
[Secure Key Injection]	Injection and Updating Secure User Keys for RA Family	R11AN0496

1.5 Development Tools

The following tools are provided by Renesas to support the development of security solutions.

Reference	Name	URL
[FSP]	Flexible Software Package (FSP) for Renesas RA MCU Family with e^2 studio IDE	www.renesas.com/fsp
[RASC]	RA Smart Configurator	www.renesas.com/fsp
[RFP]	Renesas Flash Programmer	https://www.renesas.com/software-tool/renesas-flash-programmer-programming-gui
[SKMT]	Security Key Management Tool	https://www.renesas.com/software-tool/security-key-management-tool

1.6 Abbreviations and Legend

The following abbreviations are used in this document.

Abbreviation	Meaning
AL	Authentication Level
AL1_KEY	Key used to return to the AL1 Authentication Level
AL2_KEY	Key used to return to the AL2 Authentication Level
BSP	Board Support Package
CAC	Clock Frequency Accuracy Measurement Circuit (MCU peripheral)
CEU	Capture Engine Unit (MCU peripheral)
DLM	Device Lifecycle Management

DMA	Direct Memory Access
DMAC	DMA Controller (MCU peripheral)
DTC	Data Transfer Controller (MCU peripheral)
ECC	Elliptic Curve Cryptography
ECDH/ECDHE	Elliptic Curve Diffie-Hellman (Ephemeral)
EDMAC	Ethernet DMA Controller (MCU peripheral)
FSBL	First Stage Bootloader
FSP	Flexible Software Package
HRK	Hardware Root Key
HUK	Hardware Unique Key
IDAU	Implementation Defined Attribution Unit
IDE	Interactive Development Environment
ISR	Interrupt Service Routine
KAS	Key Agreement Scheme
KUK	Key-Update Key
LCK_BOOT	Lock Boot Interface (DLM state)
MAC	Message Authentication Code
MPU	Memory Protection Unit
OEM	Original Equipment Manufacturer (DLM state)
OEM_ROOT_PK	OEM Root Public Key
PL	Protection Level
PVD	Programmable Voltage Detection (MCU module)
RASC	RA Smart Configurator
RMA_ACK	Return Material Authorisation Acknowledged (DLM state)
RMA_KEY	Return Material Authorisation Key
RMA_REQ	Return Material Authorisation Request (DLM state)
RMA_RET	Return Material Authorisation Return (DLM state)
RFP	Renesas Flash Programmer
RSIP	Renesas Secure IP
RTC	Realtime Clock (MCU peripheral)
SAU	Secure Attribution Unit
SKMT	Security Key Management Tool
TRNG	True Random Number Generator
TSN	Temperature Sensor (MCU peripheral)
UFPK	User Factory Programming Key
W-UFPK	Wrapped User Factory Programming Key

The following figure shows the concepts that are illustrated in graphics in this document.

Symbol	Meaning
	A cryptographic key (plaintext)
	A cryptographic key encrypted by another key (inner key encrypted by the outer key)
	A cryptographic key wrapped (encrypted plus MAC) by another key (inner key wrapped by the outer key)

Figure 1. Graphical Representations of Cryptographic Keys

1.7 Software Platform

The RA Flexible Software Package (FSP) is a software package provided by Renesas for developing applications on RA Family MCUs. The FSP provides Board Support Packages, HAL drivers, and middleware for RA Family MCUs.

The FSP Platform Installer includes the e^2 studio IDE, toolchain, and FSP packs. Alternatively, the FSP Standalone Installer plus the RA Smart Configurator can be used with the IAR Embedded Workbench IDE or Arm Keil MDK.

These installers, plus additional information, can be found at www.renesas.com/fsp.

1.8 Vulnerability Reporting

Renesas is committed to proactively addressing any potential security vulnerabilities within our products. To report a security vulnerability, please go to www.renesas.com/psirt and follow the instructions on the web page.

2. Identity

2.1 MCU Group Identity

The RA8M1 MCU Group chips contain a 16-byte read-only register that contains an immutable ASCII representation of the device part number. The PNRn Part Numbering Registers are documented in the [RA8M1 User Manual], including a part number listing.

This value can be obtained by using the FSP function `R_BSP_PartNumberGet()`.

For more details, refer to the following document:

- [RA8M1 User Manual]
- [FSP User Manual]

2.2 Unique Identity

The RA8M1 MCU Group chips contain a 16-byte read-only register that contains an immutable 16-byte ID code (unique ID) for identifying the individual MCU. The UIDRn Unique ID Registers are documented in the [RA8M1 User Manual].

This value is guaranteed to be unique. It is serialised and therefore predictable, so it should not be used for identifying the end-product if a random or pseudorandom identifier is required.

This value can be obtained by using the FSP function `R_BSP_UniqueIdGet()`.

For more details, refer to the following document:

- [RA8M1 User Manual]
- [FSP User Manual]

2.3 Cryptographic Identity

The RA8M1 MCU is not delivered with a provisioned cryptographic identity. The RSIP-E51A security engine can be used to create an ECC or RSA cryptographic identity by using the key (pair) generation capability of the security engine, which is statistically unique based on the SP800-90B conformance of the TRNG.

For more details, refer to the following documents and sections of this document:

- [RA8M1 User Manual]
- [FSP User Manual]
- [Device Identity]
- Section 5 Cryptographic Functions
- Section 6 Random Number Generation

3. Isolation

3.1 Arm® TrustZone®

The RA8M1 includes Arm TrustZone technology. TrustZone divides the system and the application into Secure and Non-secure domains. This separation applies to:

- On-chip flash memory (i.e., code flash)
- SRAM
- Peripherals
- Pins

For best security design practices, external memory should always be considered Non-secure.

The Secure, Non-secure Callable, and Non-secure regions of on-chip code flash and SRAM are enforced using an IDAU (Implementation Defined Attribution Unit) combined with an SAU (Secure Attribution Unit).

The address range boundaries for secure flash and SRAM are programmed in non-volatile flash and applied before the reset vector is fetched. This ensures that malicious code cannot override the boundaries. These values are calculated automatically by the e^2 studio IDE when the application code is built, and by default, they are programmed at the start of a debug session. If necessary, these calculated values can be adjusted prior to being programmed and programmed independently of the debug session.

The security configuration of pins and peripherals is done using Peripheral Security Attribution Registers. The FSP's Board Support Package start-up code will initialize these attribution registers for the pins and peripherals as per the configuration set in the application project using either the e^2 studio IDE or the RA Smart Configurator (RASC).

The system starts up in the Secure state. The application performs any secure initialization, such as firmware authentication, and then jumps to the Non-secure region for main application processing. Execution of Non-secure region code is then governed by TrustZone usage restrictions, which require all calls to Secure region code to utilise Non-secure Callable veneers. These veneers and accompanying guard functions can be configured with the e^2 studio IDE.

TrustZone access violations will generate a TrustZone access error. The application can optionally alert and/or record these events for unauthorised access attempt logging.

TrustZone usage is not mandatory. It is the responsibility of the application developer to assess the security risks of a monolithic architecture, whether to utilise TrustZone as an isolation mechanism, and what services and peripherals to place in the Secure region. The referenced documents below give some examples of how TrustZone can be used in an application.

For production programming, ensure that the TrustZone boundaries are accurately programmed. This must be done for all applications, even if TrustZone is not utilised by the application.

For more details, refer to the following documents:

- [RA8M1 User Manual]
- [Design with TrustZone]

3.2 Memory Protection Unit (MPU)

The RA8M1 MCU Group contains an Arm Cortex-M33 core with an Arm MPU and a Bus Master MPU.

The Arm MPU can be used to protect memory regions by defining access permissions for different privilege states. See the Arm Developer documentation for more information about how to use the Arm MPU: Armv8-M Memory Model and Memory Protection User Guide.

The Arm Bus Master MPU provides memory protection for bus masters other than the CPU, namely DMAC/DTC. If access to a protected region is detected, the Bus Master MPU generates an internal reset or a non-maskable interrupt.

For more details, refer to the following documents:

- [RA8M1 User Manual]

3.3 RSIP-E51A

The RSIP-E51A security engine is a cryptographic subsystem that is integrated into the MCU silicon, managed and protected by dedicated control logic. The cryptographic operations are physically isolated from the rest of the chip, with dedicated RAM for holding sensitive material (e.g., plaintext keys) during cryptographic operations. RSIP-E51A is further protected with TrustZone isolation technology.

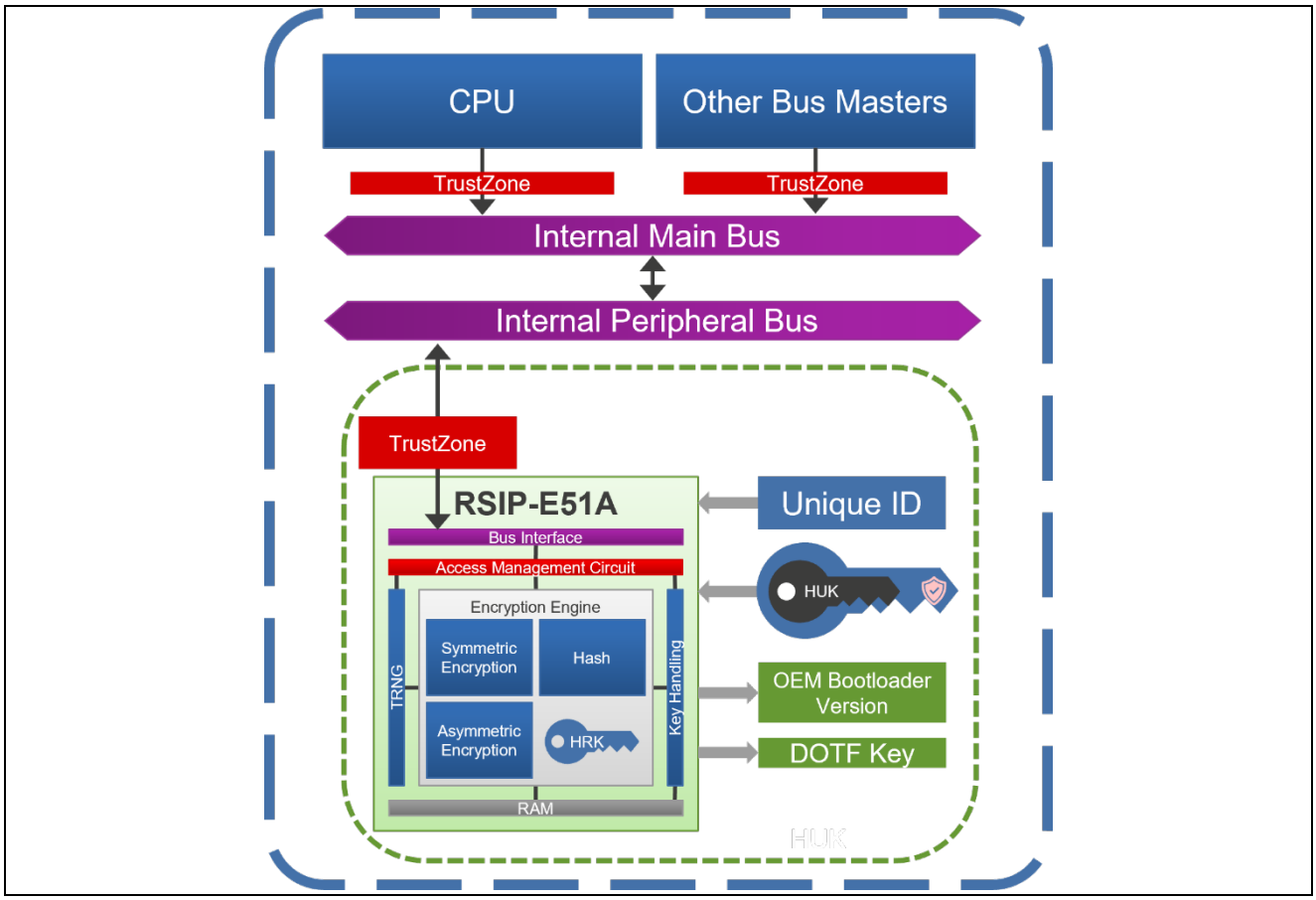

Figure 2. RSIP-E51A On-chip Isolation

For more details, refer to the following documents and sections of this document:

- [RA8M1 User Manual]
- [RSIP Modes]
- Section 4 Secure Key Injection, Storage, and Usage
- Section 5 Cryptographic Functions
- Section 9.2 RSIP-E51A Protection Features

4. Secure Key Injection, Storage, and Usage

4.1 RSIP-E51A Operational Modes

The RSIP-E51A can function in two operational modes, Protected Mode and Compatibility Mode. These modes are supported by different APIs, and they offer different levels of physical key protection.

The secure key storage mechanism is different between the two modes, resulting in incompatible securely stored keys. Therefore, the Flexible Software Package supports using **only one operational mode within a single project**. Note that applications that utilise a bootloader are implemented with a separate project for the bootloader, enabling Protected Mode use by the bootloader and Compatibility Mode use by the rest of the application.

For more details, refer to the following documents and sections of this document:

- [RSIP Modes]
- [FSP User Manual]
- Section 9.2 RSIP-E51A Protection Features

4.1.1 RSIP-E51A Protected Mode Operation

Protected Mode is designed to enforce best practice security design and implementation by supporting only the use of securely stored (i.e., wrapped) private keys. As such, the application can guarantee that there is no plaintext private key exposure on any CPU or externally accessible bus.

Both private and public keys must be securely injected. During the injection process, a MAC is appended to the key data. For most cryptography operations, this MAC is required; the only exception is for public keys used in Key Agreement Schemes (KAS) in supporting ECDH/ECDHE.

Protected Mode is used via the FSP Crypto APIs, designed to be compatible across Renesas RA, RX, and RZ Families for simplified code reuse.

Refer to section 9.2 RSIP-E51A Protection Features for more information about the protection features included in Protected Mode operation.

4.1.2 RSIP-E51A Compatibility Mode Operation

Compatibility Mode is designed to facilitate integration with legacy systems and software, while still supporting secure key injection, storage, and usage. In Compatibility Mode, plaintext private keys can be used for cryptographic functions, simplifying integration with third-party stacks and libraries.

Compatibility Mode is used via the PSA Certified Crypto APIs, providing Arm ecosystem alignment.

Refer to section 9.2 RSIP-E51A Protection Features for more information about the protection features included in Compatibility Mode operation.

Since plaintext key material can be present and used in the application, it is required that the developer evaluate and accept any associated risks.

For security best practices, private keys that are stored in non-volatile memory should be stored securely as per section. 4.3 Secure and Plaintext Key Injection and Update. Depending on the threat model, it may be acceptable to store session keys in plaintext in RAM.

Plaintext private keys should be protected via other key protection mechanisms, such as isolation in the Secure region.

4.2 Secure Key Storage

The secure key storage mechanism leverages the chip's Hardware Unique Key, a 256-bit key that is unique for each chip and is accessible only by the RSIP-E51A security engine. Application keys that have been wrapped with this key via a key injection or key update process can be stored securely at any location in any memory. This mechanism provides unlimited secure key storage and cloning protection.

For additional protection, it is recommended to store wrapped keys in the Secure region if the application utilises TrustZone.

Keys that have been generated, injected, or updated in Protected Mode cannot be used in Compatibility Mode, and vice versa.

For more details, refer to the following documents:

- [RA8M1 User Manual]
- [FSP User Manual]
- [Secure Key Injection]
- [Plaintext Key Injection]

4.3 Secure and Plaintext Key Injection and Update

The secure key injection process leverages the MCU's hardware root key, a key that is contained within the RSIP-E51A security engine and is common across all chips. This key is used only for the secure key injection process. It is not used for secure storage of application keys, and it is not accessible outside the security engine.

There are some differences between Protected Mode and Compatibility Mode, but the basic concept for secure key injection is the same for both modes.

Keys that have been generated, injected, or updated in Protected Mode cannot be used in Compatibility Mode, and vice versa.

To protect both the Renesas MCU Hardware Root Key and the developer's application keys, no sensitive key material is transferred between Renesas and the application developer. Instead, as shown in Figure 3. Secure Key Injection, the developer creates an intermediary key called a User Factory Programming Key (UFPK). This key is PGP-encrypted and sent to the Renesas Key Wrap Service, an automated secure server that wraps the UFPK (creating a W-UFPK), PGP-encrypts it, and returns it to the developer.

A non-trivial key that is not duplicated from Renesas examples or other publicly available cryptography references must be used for the UFPK.

Application keys are injected by providing both the W-UFPK and the UFPK-encrypted application key(s) to the RSIP-E51A. The security engine unwraps the W-UFPK to obtain the UFPK, then decrypts the encrypted application key. It then wraps the application key with the MCU's HUK. Figure 3. Secure Key Injection shows an overview of this process.

Key preparation steps where keys are exposed in plaintext must be performed in a secure environment.

It is recommended to use different UFPKs for prototype development with test keys and for production programming of mass production keys.

The RA8M1 cannot be personalized to accept only specific W-UPFKs for application key injection; therefore, the production programming flow must protect against malicious key injection. To guard against supply chain attacks, it is recommended to issue the "Initialize" boot firmware command to erase the chip prior to key injection.

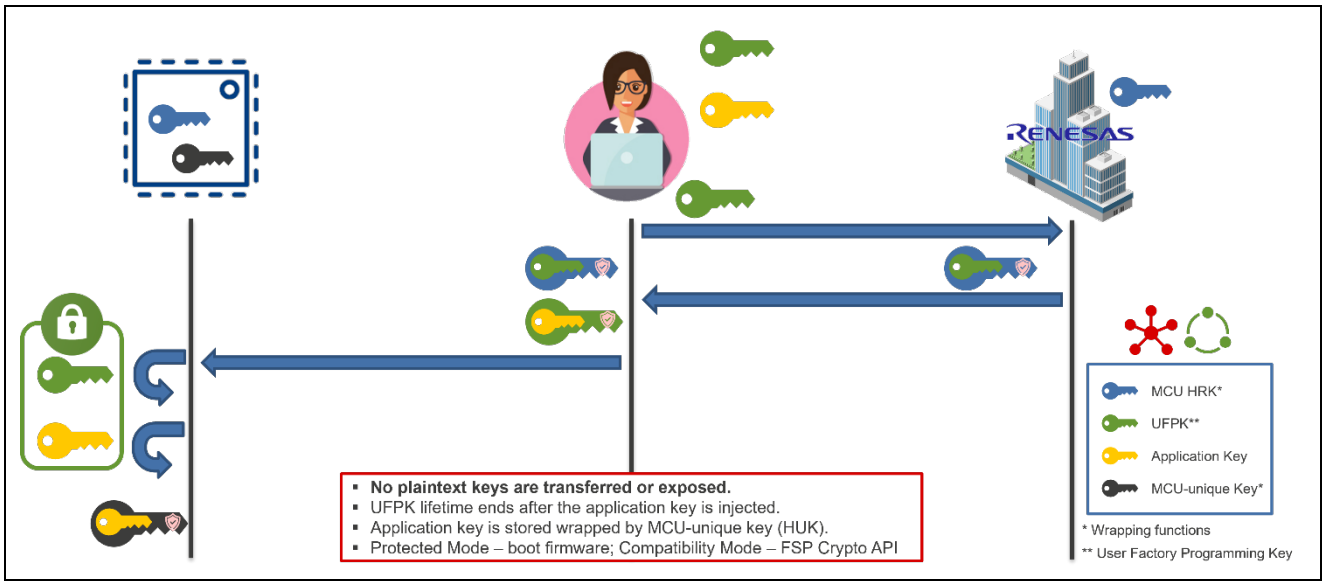

Figure 3. Secure Key Injection

On the RA8M1, Protected Mode secure key injection is performed using the factory boot firmware, and Compatibility Mode secure key injection is performed using APIs contained in the FSP.

Renesas provides the following tools to support this process:

- Security Key Management Tool (SKMT) - This tool can create sample keys and prepare keys for secure key injection and secure key update. [SKMT]
- Renesas Flash Programmer (RFP) - This tool can securely inject Protected Mode keys using the MCU's factory boot firmware. [RFP]

Select third-party programming tools also support secure key injection.

Plaintext key injection is supported only by Compatibility Mode and is performed using APIs contained in the FSP.

The following sections provide more details about the secure and plaintext key injection and update processes.

For more details, refer to the following documents:

- [RA8M1 User Manual]
- [FSP User Manual]
- [Boot Firmware]
- [RSIP Modes]
- [Secure Key Injection]
- [Plaintext Key Injection]
- [SKMT]
- [RFP]

4.3.1 RSIP-E51A Protected Mode Key Injection

Keys used with Protected Mode must be securely injected or securely updated. The wrapped keys can then be used with the FSP Crypto APIs.

Secure key injection is performed using the programming interface and the MCU's factory boot firmware. As shown in Figure 4. Protected Mode Secure Key Injection, the W-UFPK, and the encrypted application key are all provided to the MCU using the appropriate factory boot firmware command. The factory boot firmware

uses the security engine to decrypt the application key and wrap it with the MCU's HUK. The boot firmware then stores the key at the designated location in memory.

Keys injected via this mechanism cannot be used with the PSA Certified Crypto APIs, which use RSIP-E51A in Compatibility Mode.

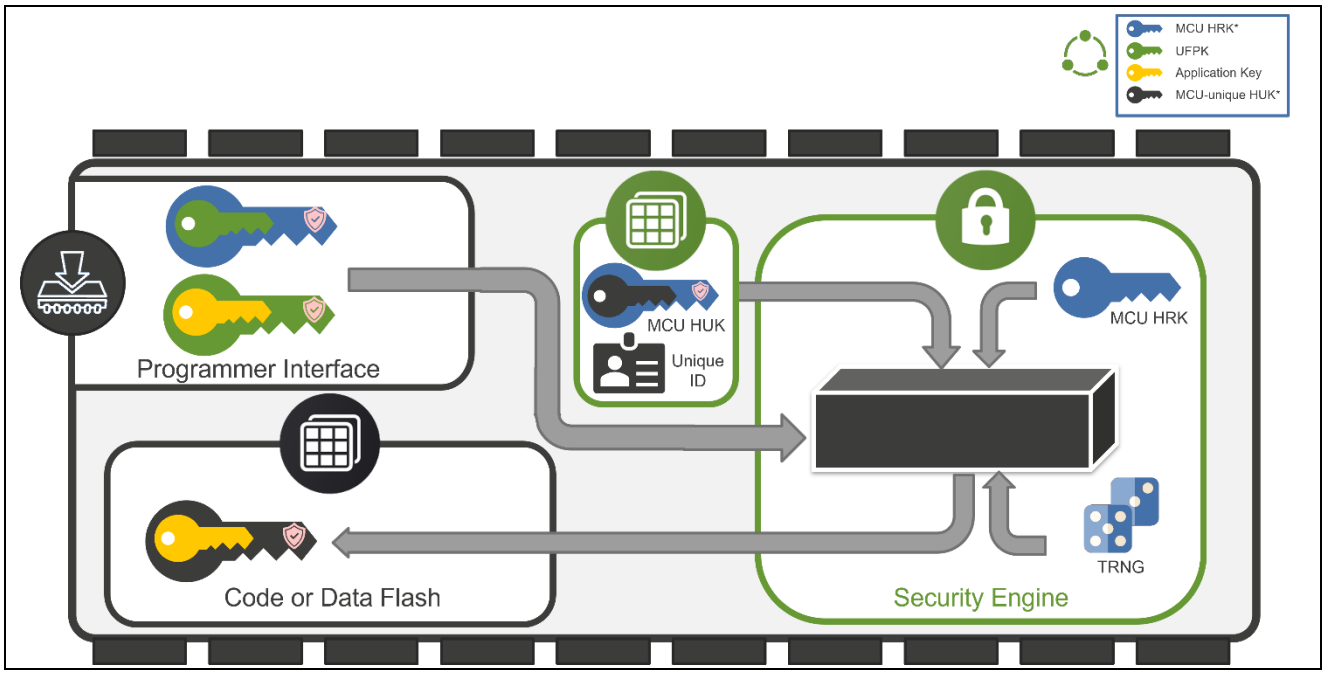

Figure 4. Protected Mode Secure Key Injection

The production programming flow should be designed such that the programming interface is locked after secure key injection to prevent malicious key injection.

After the programming interface is locked (see section 13.1 Device Lifecycle Management and Debug Authentication), application keys can no longer be injected. Instead, the application must be updated with new keys via Key-Update Keys (KUKs).

During initial provisioning, an application should be provisioned with one or more KUKs. As shown in Figure 5. Key-Update Key Injection, Key-Update Key injection functions identically to application key injection.

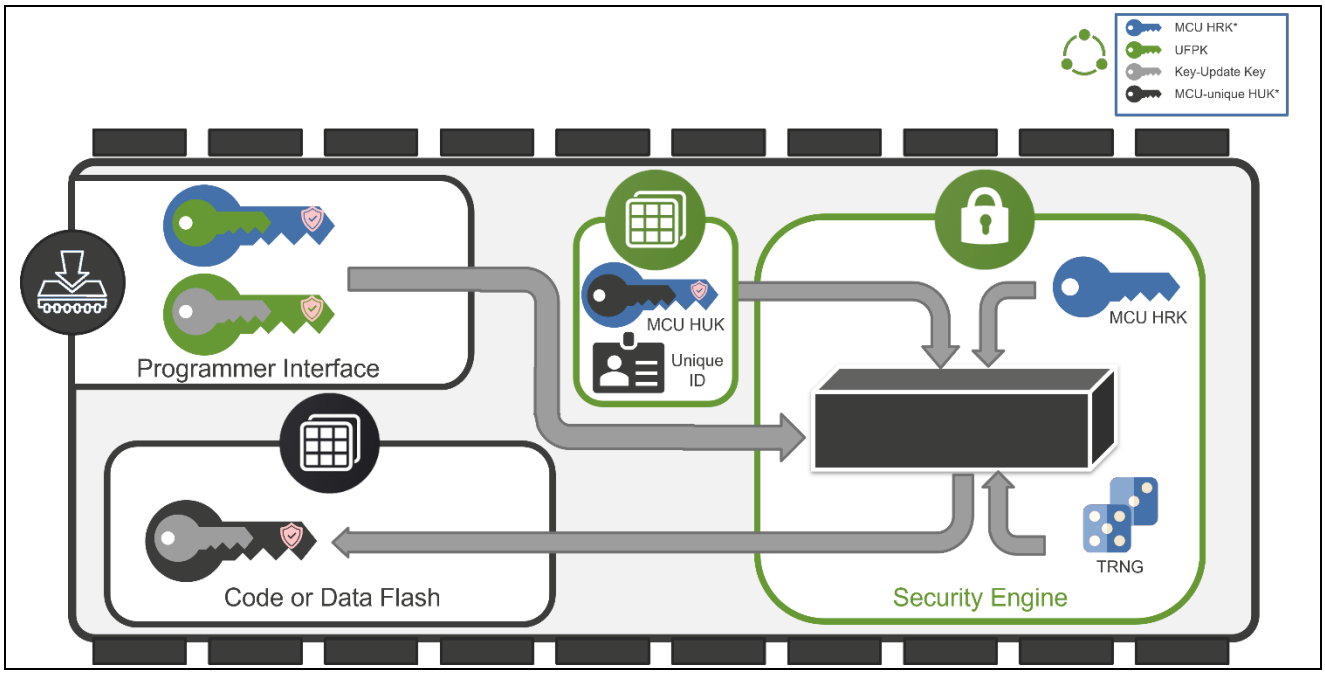

Figure 5. Key-Update Key Injection

To update an application key, the new application key must be wrapped with the Key-Update Key. The application code then uses the appropriate FSP Crypto API to update the specific key type of the new application key. As shown in Figure 6. Protected Mode Key Update, this enables secure key update, with no plaintext key exposure outside the security engine.

Keys updated via this mechanism cannot be used with the PSA Certified Crypto APIs, which use RSIP-E51A in Compatibility Mode.

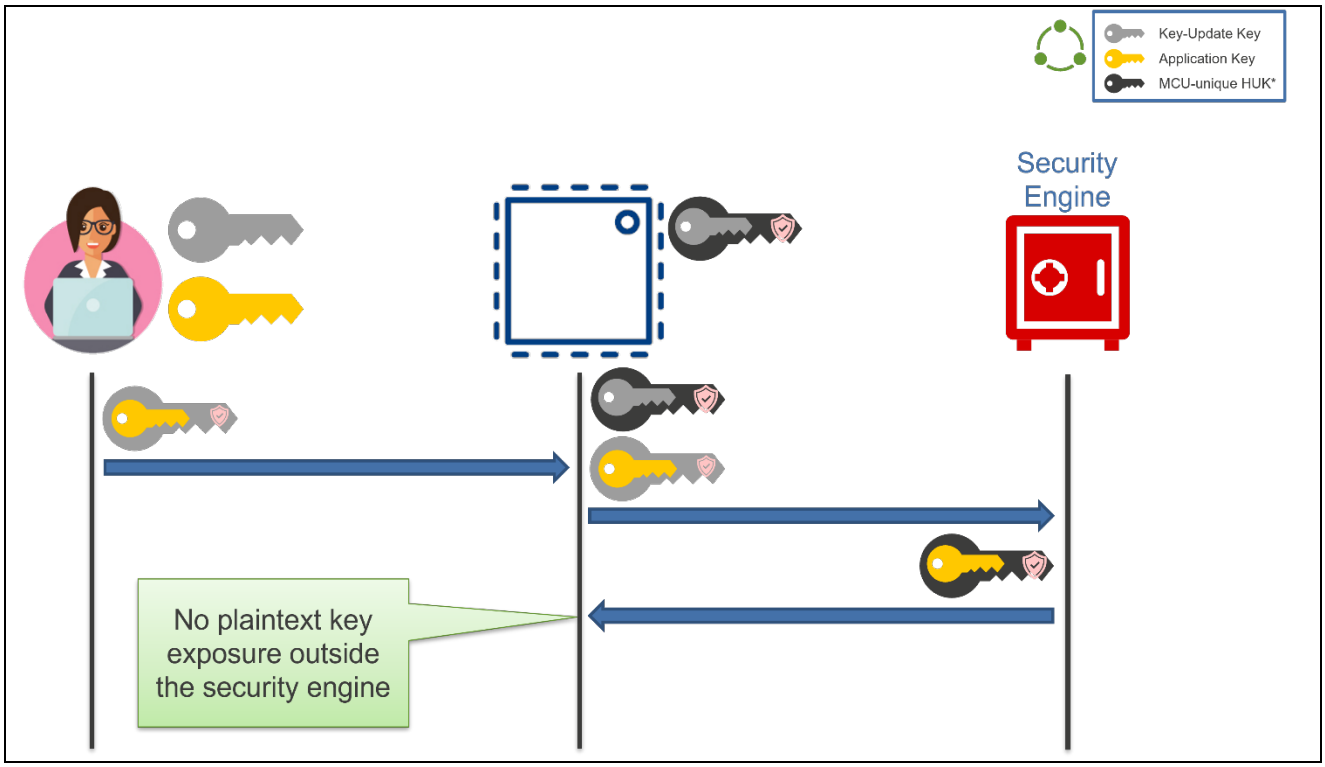

Figure 6. Protected Mode Key Update

4.3.2 RSIP-E51A Compatibility Mode Key Injection

Keys used with Compatibility Mode must use the FSP Key Injection module and be either securely injected or injected as plaintext. The wrapped keys can then be used with the PSA Certified Crypto APIs.

Keys injected via this mechanism cannot be used with the FSP Crypto APIs, which use RSIP-E51A in Protected Mode.

Secure key injection is performed using the FSP Key Injection module. As shown in Figure 7. Compatibility Mode Secure Key Injection, the W-UFPK and encrypted application key are both provided to the API. The RSIP driver uses the security engine to decrypt the application key and wrap it with the MCU's HUK. The application code must then store the key at a designated location in memory.

It is recommended not to include either secure key injection API code or any W-UFPKs in end-product firmware. This can be done by utilising a programming tool that executes the secure key injection code from the MCU RAM or by erasing the flash memory that contains the secure key injection code.

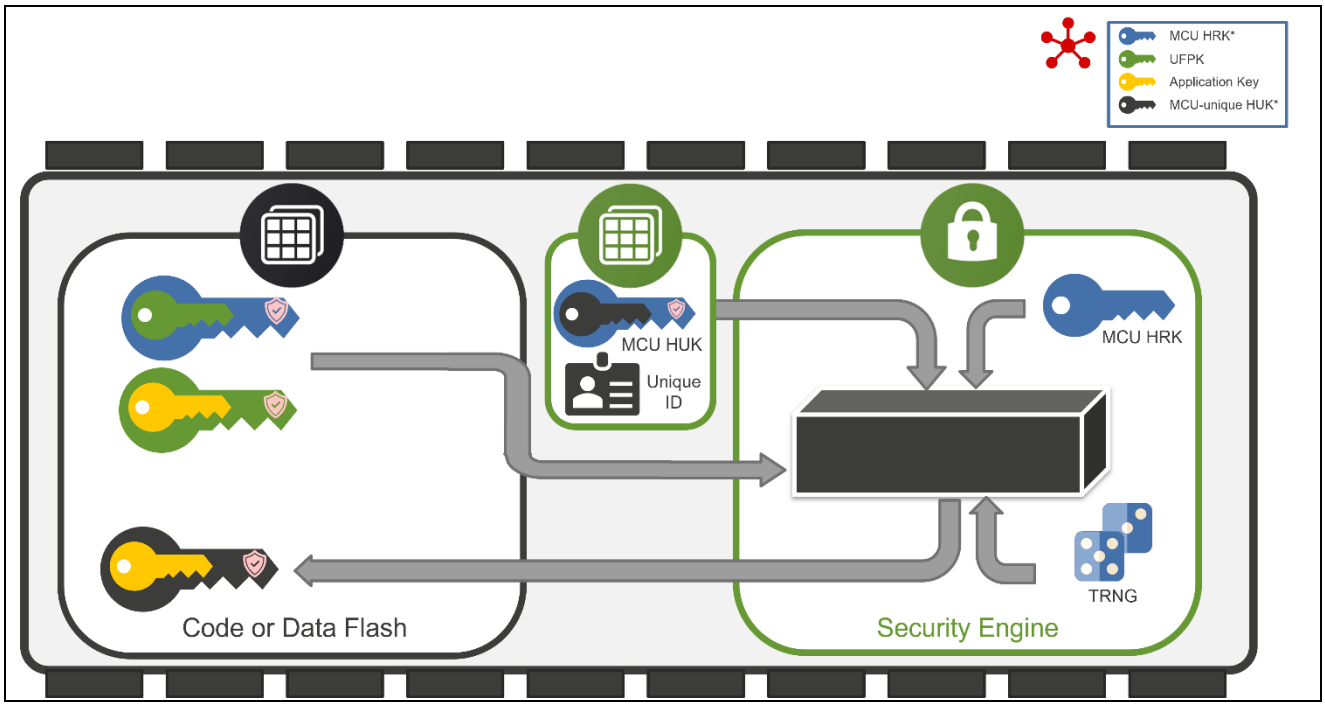

Figure 7. Compatibility Mode Secure Key Injection

Plaintext key injection is also performed using the FSP Key Injection module. As shown Figure 8. Compatibility Mode Plaintext Key Injection, the plaintext application key is provided to the API. The RSIP driver uses the security engine to wrap the plaintext key with the MCU's HUK. The application code must then store the wrapped key at a designated location in memory.

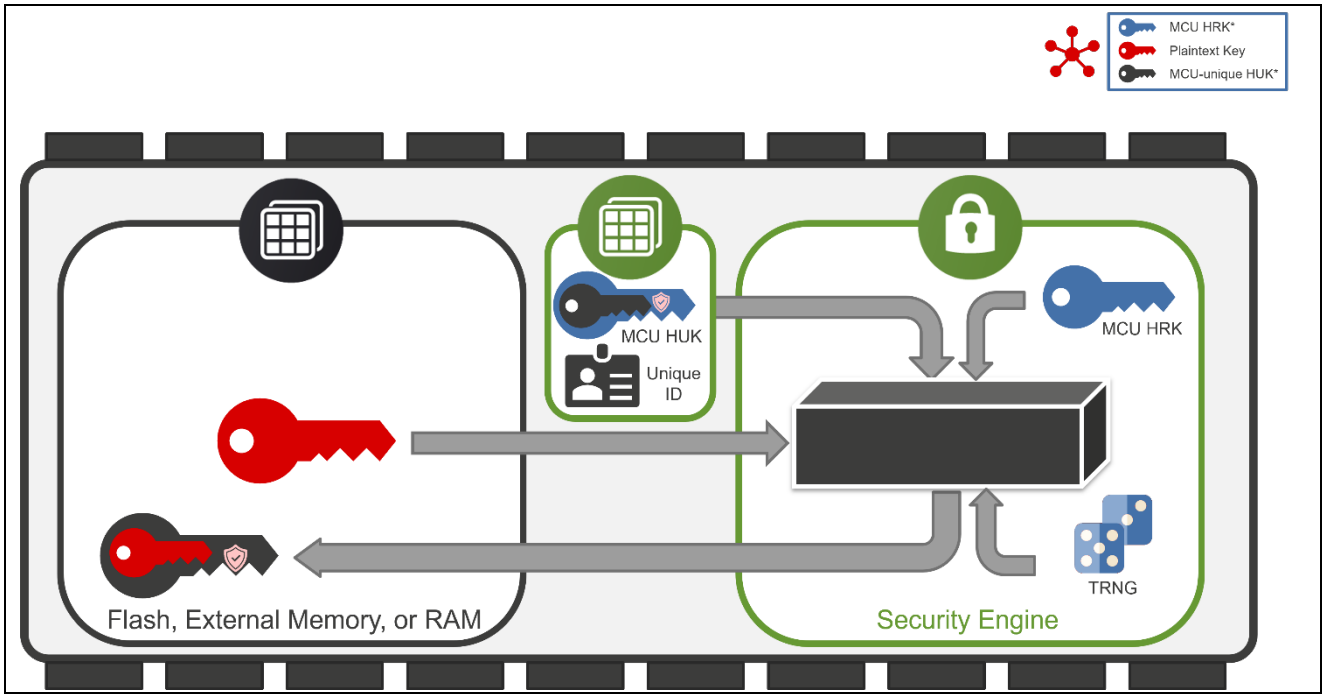

Figure 8. Compatibility Mode Plaintext Key Injection

There is no explicit support in Compatibility Mode for Key Update. It is recommended to securely inject one or more keys that will serve the role of Key-Update Keys and manually perform the decryption and plaintext key injection of the new key. See Figure 9. Compatibility Mode Key Update for an overview of this process.

Since there is plaintext key exposure outside the security engine, it is recommended to perform the entire key update operation in the Secure region.

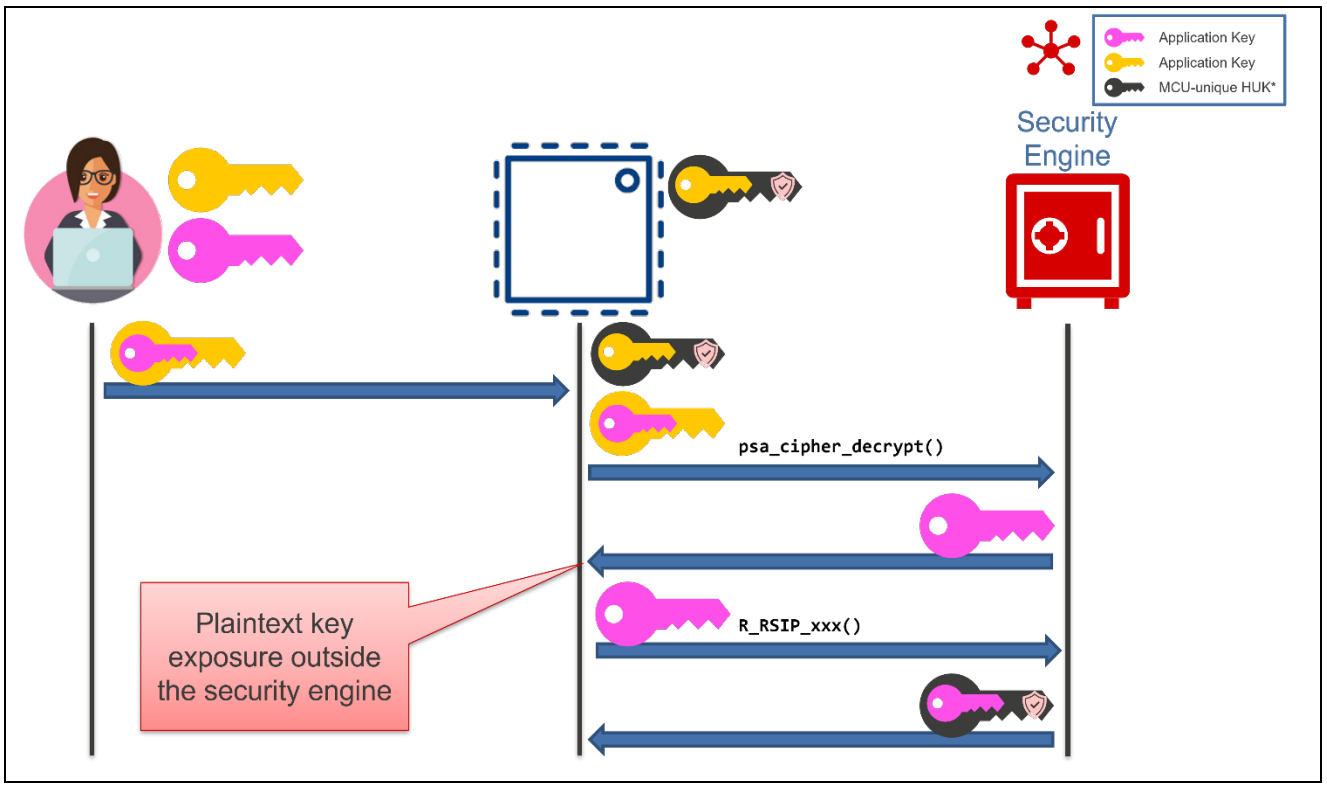

Figure 9. Compatibility Mode Key Update

5. Cryptographic Functions and Key Generation

The RSIP-E51A security engine provides hardware acceleration for basic cryptographic functions. The FSP Crypto APIs (Protected Mode) and PSA Certified Crypto APIs (Compatibility Mode) provide any needed software support.

Table 1 RSIP-E51A Cryptographic Operations lists the cryptographic algorithms supported by the security engine. Table 2 RSIP-E51A Key Generation lists the cryptographic key generation supported by the security engine.

Application code should use cryptographic algorithms with sufficient strength as appropriate to the application's threat model.

Table 1 RSIP-E51A Cryptographic Operations

Algorithm	Operations	Specification	Key lengths	Modes
AES	Encryption, decryption, MAC	NIST FIPS PUB 197 NIST SP800-38A (ECB, CBC, CTR) NIST SP800-38B (CMAC) NIST SP800-38C (CCM) NIST SP800-38D (GCM, GMAC)	128 and 256 bits	ECB, CBC, CTR, CCM, GCM, CMAC, GMAC
ECC	Signature generation, signature verification	NIST FIPS PUB 186-5	Curves: NIST P-256, P-384, P-521; secp256k1; Brainpool P256r1, P384r1, P512r1	N/A
RSA	Signature generation, signature verification, encryption, decryption	IETF RFC 8017 FIPS PUB 186-5	2048, 3072, 4096 bits	N/A
Ed25519(1)	Signature generation (PureEdDSA), signature verification	RFC8032	N/A	N/A
HMAC	Keyed Hash	NIST FIPS PUB 198-1	256 bits	SHA-256, SHA-384, SHA-512
KDF	Key Agreement	NIST SP 800-56A	128 and 256 bits	SHA-256, SHA-384
Key Wrapping	Wrap/Unwrap	RFC3394	128 and 256 bits	N/A
SHA	Hash	NIST FIPS PUB 180-4	N/A	SHA-224, SHA-256, SHA-384, SHA-512

(1) Protected Mode only

Table 2 RSIP-E51A Key Generation

Algorithm	Specification	Key lengths
AES	NIST FIPS PUB 197	128 and 256 bits
RSA	IETF RFC 8017 FIPS PUB 186-5	2048, 3072, 4096 bits
ECC – SEC prime curves	FIPS PUB 186-5	256 and 384 bits
ECC – Brainpool prime curves	RFC5639	256 and 384 bits
ECC – Koblitz curves	SEC 2: Recommended Elliptic Curve Domain Parameters	256 bits
Ed25519	RFC8032	N/A

For more details, refer to the following document:

- [RA8M1 User Manual]

6. Random Number Generation

The RA8M1 MCU Group implements random number generation by utilising the True Random Number generation capability of the RSIP-E51A security engine. The TRNG implementation consists of an SP800-90A compliant DRBG that is fed by an SP800-90B compliant seed, which is generated from an entropy source. Each TRNG request generates a 128-bit random number.

The TRNG is initially seeded by calling the APIs to initialize the security engine:

- Protected Mode: `R_RSIP_Open()`
- Compatibility Mode: `mbedtls_platform_setup()`

Periodic reseeding of the TRNG is typically recommended for most applications that utilise a TRNG. Reseeding the TRNG can be performed as follows:

The reseeding interval for Protected Mode must be manually implemented by the application code. The application code must call `R_RSIP_Close()` followed by `R_RSIP_Open()` to reseed the TRNG.

The reseeding interval for Compatibility Mode is configured by defining the macro `MBEDTLS_CTR_DRBG_RESEED_INTERVAL` and setting it to the number of calls that can be made to `psa_generate_random()` before reseeding is required. The macro can be defined and set using either the e^2 studio IDE or the RA Smart Configurator (*MbedTLS (Crypto Only) module> RNG*). Reseeding will occur automatically if needed when `psa_generate_random()` is called.

For more details, refer to the following documents:

- [RA8M1 User Manual]
- [SP800-90B Test Report]

7. Immutable Memory

The code flash blocks of the MCU can be configured to be immutable. This property is commonly used to protect critical assets that define the identity of the device and control access to the device through a secure firmware update mechanism. Refer to the [RA8M1 User Manual] for the sizes and addresses of the MCU's code flash blocks.

An application that intends to be configured with immutable flash blocks can be prototyped by using the Block Protect Setting Register (BPS) and the Bank Select Register Secure (BPS_SEC) to temporarily configure the blocks as read-only. These settings can be modified using either self-programming or issuing the "Initialize" boot firmware command.

Production programming of the end-product should also use the equivalent bits in the Permanent Block Protect Setting Register (PBPS) and the Permanent Block Protect Setting Register Secure (PBPS_SEC) to make these blocks permanently immutable. Any programming of the PBPS or PBPS_SEC bits from 1 to 0 is permanent, and the chip can no longer be erased using the "Initialize" boot firmware command.

The BPS, BPS_SEC, PBPS, and PBPS_SEC registers can be set using the FSP Configurator in the e^2 studio IDE or the RA Smart Configurator.

The code flash blocks that contain the Root of Trust should be made immutable. The immutable Root of Trust contains items such as a secure boot solution, root keys, and certificates.

If a chip is returned to Renesas for failure analysis, the DLM state transition to RMA_REQ (Return Material Authorization Request) will not erase any blocks that have their associated PBPS or PBPS_SEC bit set. Information in these blocks will be visible to Renesas during the RMA process.

The Protection Programming Flag (FSPR) bit is used to prevent modification of the startup area selection and to disallow the “Chip Erase” command. Any programming of the FSPR bit from 1 to 0 is permanent, and the chip can no longer be erased using the “Initialize” boot firmware command.

For more details, refer to the following documents:

- [RA8M1 User Manual]
- [FSP User Manual]

8. Secure External Storage

The Decryption On-The-Fly peripheral of the MCU enables secure external storage of application code and/or data on OSPI memory. Figure 10. DOTF with OSPI Block Diagram shows a block diagram of the DOTF peripheral and its relationship to OSPI and the RSIP-E51A security engine. This enables transparent execution of code or reading of data that is stored externally in a secure, encrypted format.

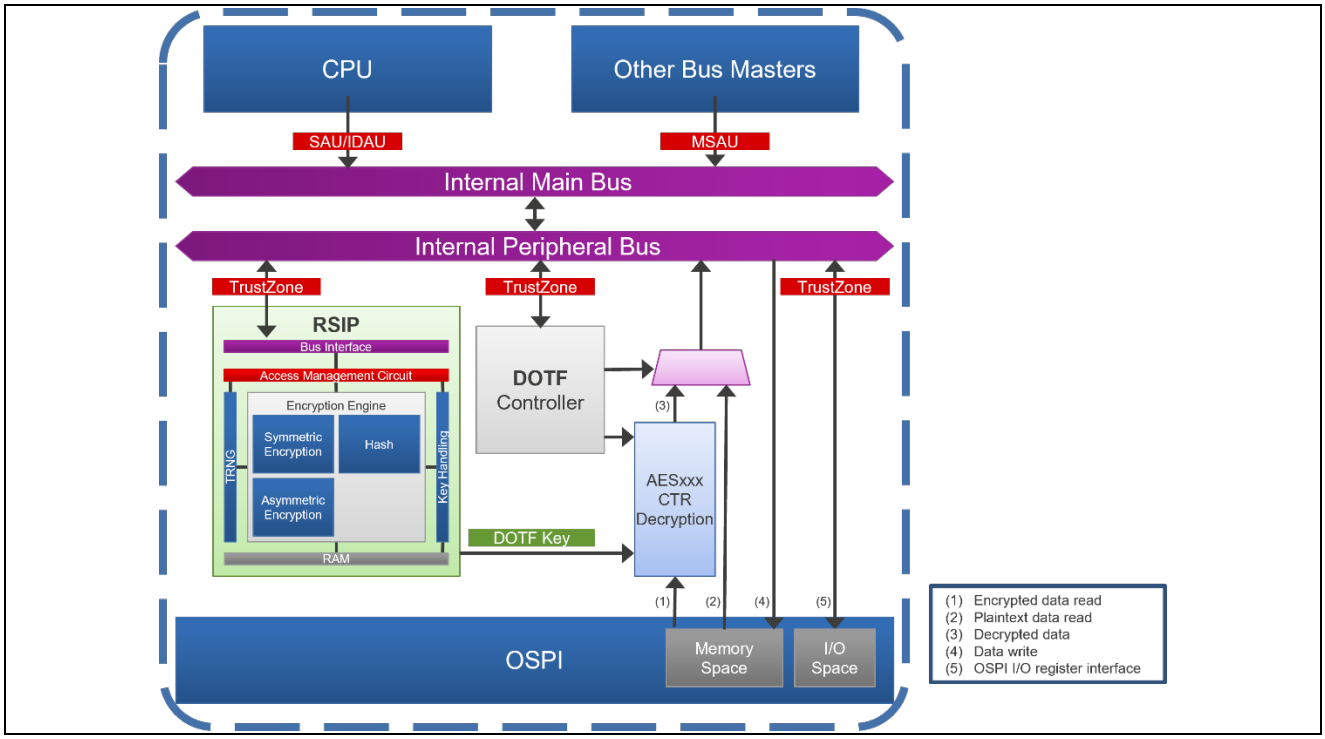

Figure 10. DOTF with OSPI Block Diagram

There are two basic use cases for DOTF:

1. Code or data is independently encrypted and stored on the OSPI memory. The decryption key is provisioned on the MCU such that the application can decrypt the external OSPI memory on the fly.
2. The MCU internally generates a key. The MCU then uses that key to encrypt data, writing the encrypted data to the OSPI memory for secure storage and later use.

The DOTF peripheral works in conjunction with the RSIP-E51A security engine. The RSIP-E51A operational mode that is in use when the DOTF feature of the OSPI peripheral is initialised must match the operation mode used to inject or generate the DOTF key.

The DOTF peripheral supports AES encryption in Counter mode with 128, 192, or 256-bit keys. The counter is calculated as:

Counter[127:0] = IV[127:28] || (Address[31:4] >> 4)

The Security Key Management Tool (SKMT) supports encrypting raw code/data for use with DOTF. SKMT takes an S-record file (*.mot or *.srec) and produces an output S-record file, supporting many use cases, including:

- Application debugging. The input file contains plaintext of both internal flash and OSPI memory application code and/or data, such as would be generated by compilation of application code. In the output file, all or a portion of the plaintext OSPI information is replaced with encrypted information, retaining all other information in plaintext. All addresses from the original input file are retained. See Figure 11. SKMT Support for DOTF Debugging.

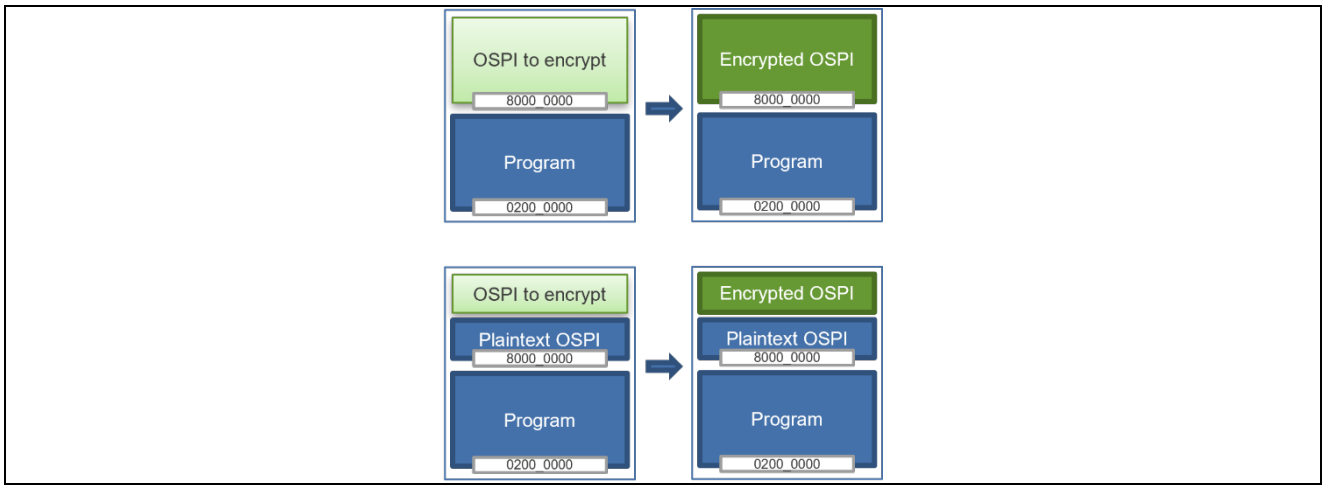

Figure 11. SKMT Support for DOTF Debugging

- Production programming. The input file contains both internal flash and OSPI memory application code and/or data, such as would be generated by compilation of application code. The output file contains the encrypted OSPI information as a separate file with addressing starting at 0 for external programming of the OSPI. This file can then be used for external programming of an OSPI memory chip. See Figure 12. SKMT Support for DOTF Production Programming, Partial File.

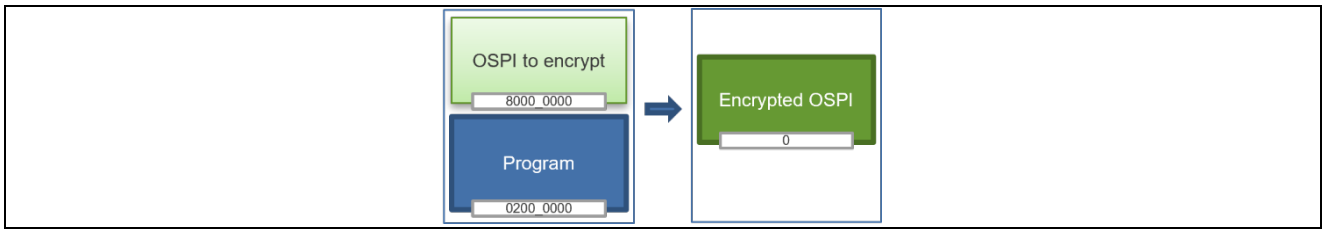

Figure 12. SKMT Support for DOTF Production Programming, Partial File

- Production programming. The input file contains separately generated data that will be stored on OSPI memory (for example, large graphical images). The output file contains the encrypted OSPI data as a separate file with addressing starting at 0 for external programming of the OSPI. This file can then be used for external programming of an OSPI memory chip. See Figure 13. SKMT Support for DOTF Production Programming, Complete File.

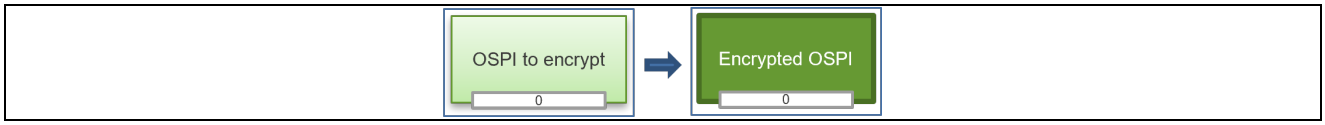

Figure 13. SKMT Support for DOTF Production Programming, Complete File

Any range of valid OSPI address space can be configured as the region to be decrypted by DOTF. Multiple keys with separate address ranges can be used by reconfiguring and re-initialising the OSPI peripheral.

Encryption On-The-Fly is not supported. To employ the DOTF peripheral to work with securely stored data generated by the application, the application must (1) configure the OSPI to use a key that the application will generate, (2) generate the AES key, (3) open the OSPI peripheral, (4) encrypt the plaintext data using the generated AES key and correct counter and IV values, (5) write the cipher text to the OSPI memory. Subsequent reads of the encrypted data from OSPI memory will be transparently decrypted.

For more details, refer to the following documents and tools:

- [RA8M1 User Manual]
- [FSP User Manual]
- [DOTF]
- [SKMT]

9. Tamper Protection

9.1 Passive Tamper Detection

Passive tamper detection is provided by the Realtime Clock (RTC) peripheral of the MCU. The RTC can be configured such that a change on one of the input capture pins, RTCICn, triggers a capture of the current RTC value (i.e., the current time) and an optional interrupt. The captured value is retained after reset.

Refer to the detailed description of this peripheral and its associated electrical characteristics as stated in the [RA8M1 User Manual] for the requirements and limitations of this hardware feature.

Figure 14. Passive Tamper Detect shows the passive tamper components included in the RTC.

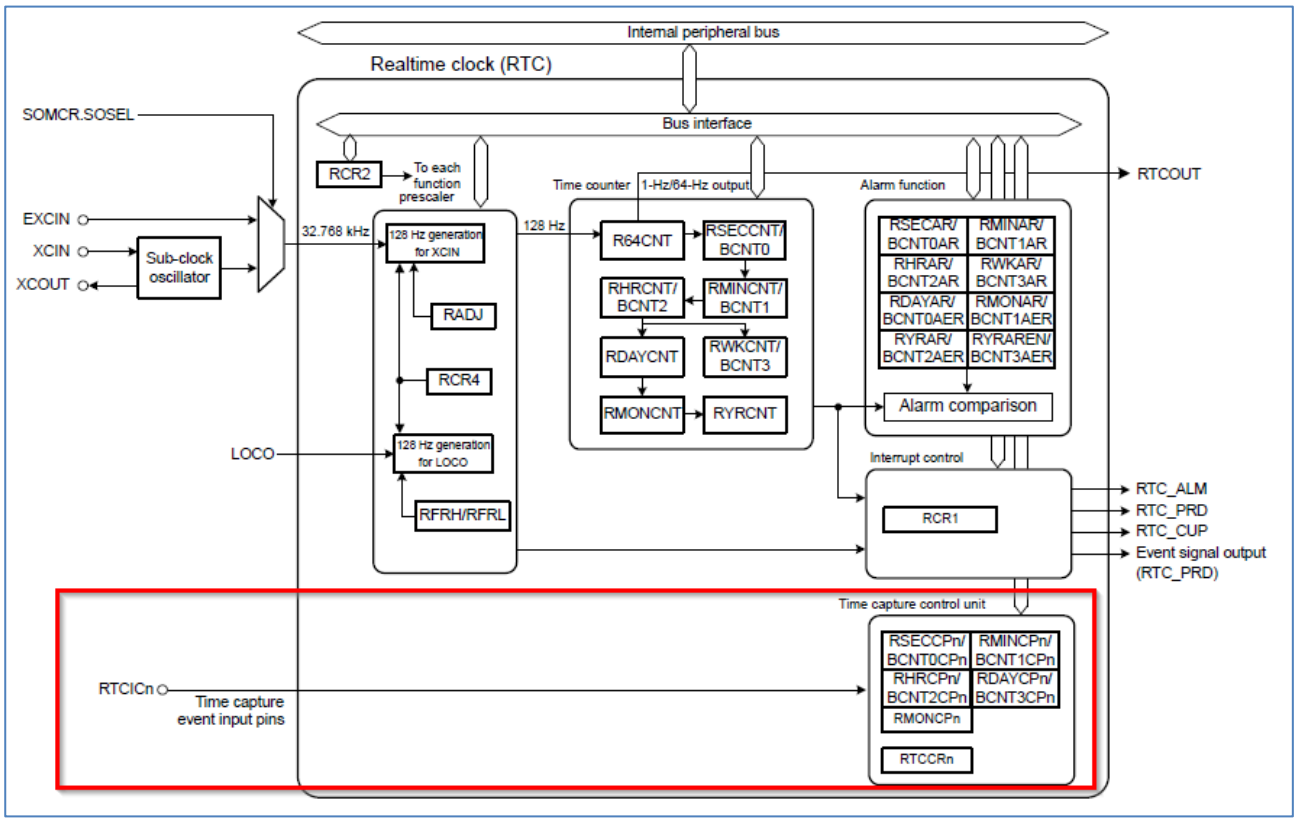

Figure 14. Passive Tamper Detect

Use of the passive tamper detection feature is application-specific and dependent on the end product's threat model. A typical use of passive tamper detection is to detect enclosure breaches that give an attacker physical access to a sensitive area of the end-product.

It is recommended to place the code that interacts with this peripheral in the Secure region if the application utilises TrustZone.

For more details, refer to the following documents:

- [RA8M1 User Manual]
- [FSP User Manual]

9.2 RSIP-E51A Protection Features

RSIP-E51A includes protection features that are designed to protect key material from physical and logical attacks.

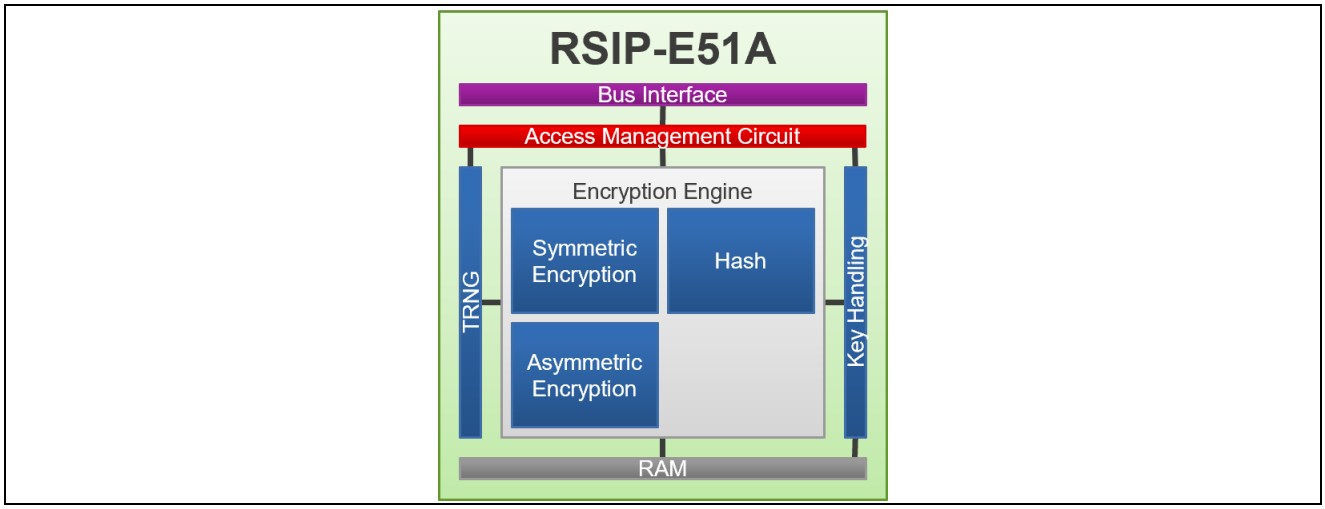

Figure 15. RSIP-E51A Block Diagram

As shown in Figure 15. RSIP-E51A Block Diagram, access to the internal components of the security engine is protected by the Access Management Circuit. The software driver provided in the FSP must be used to perform the various cryptographic functions supported by RSIP-E51A. This driver performs the proper authenticated access sequence to interface with the RSIP-E51A's Access Management Circuit. Improper access attempts via the CPU or the debugger will result in the Access Management Circuit generating an interrupt (RSIP_TADI) and locking the security engine so it can no longer be used to perform cryptographic functions. A device reset is required to clear the error.

The application should use the e^2 studio IDE or the RA Smart Configurator to enable the RSIP_TADI interrupt event and call an application-defined ISR. The RSIP signal will be armed during the RSIP initialization process. The interrupt should be enabled immediately after RSIP initialization and disabled prior to RSIP shutdown. It is important to note that repeating the initialization process will trigger the interrupt, even if the shutdown process is performed. To prevent any issues, clear the interrupt status prior to enabling the interrupt; otherwise, the interrupt will trigger when RSIP is initialized a second time.

It is recommended to place the code that interacts with this peripheral in the Secure region if the application utilises TrustZone.

The following sections provide more details about the additional hardware protection features that are available for each operational mode of the security engine. For more details, refer to the following documents:

- [RA8M1 User Manual]
- [FSP User Manual]
- [RSIP Modes]
- Section 3.3 RSIP-E51A
- Section 4 Secure Key Injection, Storage, and Usage
- Section 5 Cryptographic Functions

9.2.1 RSIP-E51A Protected Mode Protection Features

Protected Mode provides timing attack protection on symmetric and asymmetric cryptographic algorithms, implementing constant-time algorithms when dealing with key material.

Protected Mode also provides SPA/DPA protections on AES operations.

In Protected Mode, all keys, both public and private, must be securely injected such that they have an associated MAC that binds them to the specific MCU chip. When performing cryptographic operations with either a public or a private key, the MAC associated with the key is checked prior to performing any cryptographic operation with that key. If the MAC verification fails, an error is returned.

9.2.2 RSIP-E51A Compatibility Mode Protection Features

Compatibility Mode provides timing attack protection on symmetric and asymmetric cryptographic algorithms, implementing constant-time algorithms when dealing with key material.

SPA/DPA protections are not provided for AES operations. This security risk must be analysed and accepted when considering Compatibility Mode.

When performing cryptographic operations with a wrapped private key, the MAC associated with the key is checked prior to performing any cryptographic operation with that key. If the MAC verification fails, an error is returned.

Public keys in Compatibility Mode have no MAC. The security risk of using non-authenticated public keys must be analysed and accepted when considering Compatibility Mode.

9.3 Operational Temperature Monitoring

Some attacks involve operating the MCU outside its specified temperature range to try to induce erroneous behavior. The on-chip Temperature Sensor (TSN) peripheral can be used to monitor the die temperature.

Refer to the detailed description of this peripheral and its associated electrical characteristics as stated in the [RA8M1 User Manual] for the requirements and limitations of this hardware feature.

Application code must periodically check the die temperature, compare it to the device’s operating range, and trigger a fault if the temperature is out of range.

If temperature attacks are in scope for the application’s threat model, this check should be performed at a rate that is consistent with the threat model of the application.

It is recommended to place the code that interacts with this peripheral in the Secure region if the application utilises TrustZone.

For more details, refer to the following documents:

- [RA8M1 User Manual]
- [FSP User Manual]

9.4 Operational Voltage Monitoring

Some attacks involve operating the MCU outside its specified voltage range to try to induce erroneous behavior. The Programmable Voltage Detection (PVD) modules can be used to monitor the voltage level input to the VCC and EXLVD pins and alert if a selectable voltage threshold is crossed.

Refer to the detailed description of this module and its associated electrical characteristics as stated in the [RA8M1 User Manual] for the requirements and limitations of this hardware feature.

There are three programmable voltage monitors on the MCU.

Voltage Monitor 0 can be configured to be active immediately upon reset using the voltage level selected in the Option-Setting Memory, resetting the MCU if the voltage drops below the selected value. Option Function Select Register 1 can be set to be in the Secure region to effectively place Voltage Monitor 0 in the Secure region.

It is recommended to configure Voltage Monitor 0 to be active immediately upon reset at a level that is appropriate for the application, and to configure Option Function Select Register 1 to be in the Secure region to mitigate voltage fault injection attacks.

Voltage Monitor 1 and Voltage Monitor 2 can be configured for more advanced voltage detection scenarios.

It is recommended to place the code that interacts with the PVDs in the Secure region if the application utilises TrustZone.

For more details, refer to the following documents:

- [RA8M1 User Manual]
- [FSP User Manual]

9.5 Clock Protection

Some attacks involve varying the frequency of the MCU clock to try to induce erroneous behaviour. The Clock Frequency Accuracy Measurement Circuit (CAC) peripheral counts pulses of the measurement target clock using a measurement reference clock. If the number of pulses is not within an allowable range, an interrupt is generated.

Refer to the detailed description of this peripheral and its associated electrical characteristics as stated in the [RA8M1 User Manual] for the requirements and limitations of this hardware feature.

If the application uses an external clock and clock attacks are in scope for the application’s threat model, the application should utilise the CAC.

It is recommended to place the code that interacts with this peripheral in the Secure region if the application utilises TrustZone.

For more details, refer to the following documents:

- [RA8M1 User Manual]
- [FSP User Manual]

10. Internet Connectivity

The RA8M1 MCU Group includes a one-channel Ethernet Controller (ETHERC) compliant with the Ethernet or IEEE802.3 Media Access Control (MAC) layer protocol, plus one channel for the Ethernet DMA Controller (EDMAC) for the Ethernet Controller (ETHERC). These peripherals can be used to provide wired internet connectivity.

The majority of cybersecurity threats faced by a device that incorporates an internet-connected microcontroller will involve the internet connection. It is vital to understand both the threat model and the regulatory compliance requirements of the end-product.

The EU Cyber Resilience Act (CRA) requires that any product placed on the EU market that incorporates an RA6M5 microcontroller and may have a direct or indirect logical or physical data connection to a device or network must employ adequate security measures.

Secure connectivity requirements apply to both public and private infrastructure. A connected device should not be considered exempt from security requirements simply because it is connected to private infrastructure.

TCP connections should include the TLS protocol; UDP connections should include the DTLS protocol.

Ethernet stacks are highly complex software that are a prime target for cyberattacks. Ethernet stack providers, including the RA Family FSP, regularly provide updates containing fixes for security vulnerabilities. The Ethernet stack in the end-product should be updatable such that these fixes can be deployed to the end-product. Anti-rollback protection on these updates should also be provided to prevent an attacker from regressing the end-product firmware to an older version with exploitable vulnerabilities.

For more details, refer to the following documents and tools:

- [RA8M1 User Manual]
- [FSP User Manual]
- [Cloud Connectivity]
- [Cloud OTA]
- [FSP]

11. Secure Boot

To ensure secure initialization of the application, it is recommended to utilise a secure boot solution.

In high-end computing systems, “bootloading” is the process of loading code from external storage into memory so it can be executed. It is typically done in stages, with a small first-stage bootloader residing in immutable memory to ensure that the Root of Trust cannot be tampered with. The first-stage boot typically authenticates a second-stage bootloader, which is loaded from external memory and contains significantly more functionality. The second-stage bootloader typically contains more system information and authenticates additional code prior to that code’s execution.

When translated to microcontrollers, the first-stage bootloader (if present) is stored in non-modifiable memory, often masked ROM. The advantage of this architecture is that it enables secure update of a second-stage bootloader.

11.1 First Stage Bootloader (FSBL)

The RA8M1 offers a First Stage Bootloader (FSBL), which is masked in ROM. When configured and enabled, the FSBL executes after a reset to either authenticate or integrity check a specified region of on-chip flash prior to its execution. Figure 16. FSBL Used with SSBL and TrustZone shows an architecture where the FSBL is used with a second-stage bootloader in a TrustZone-enabled application.

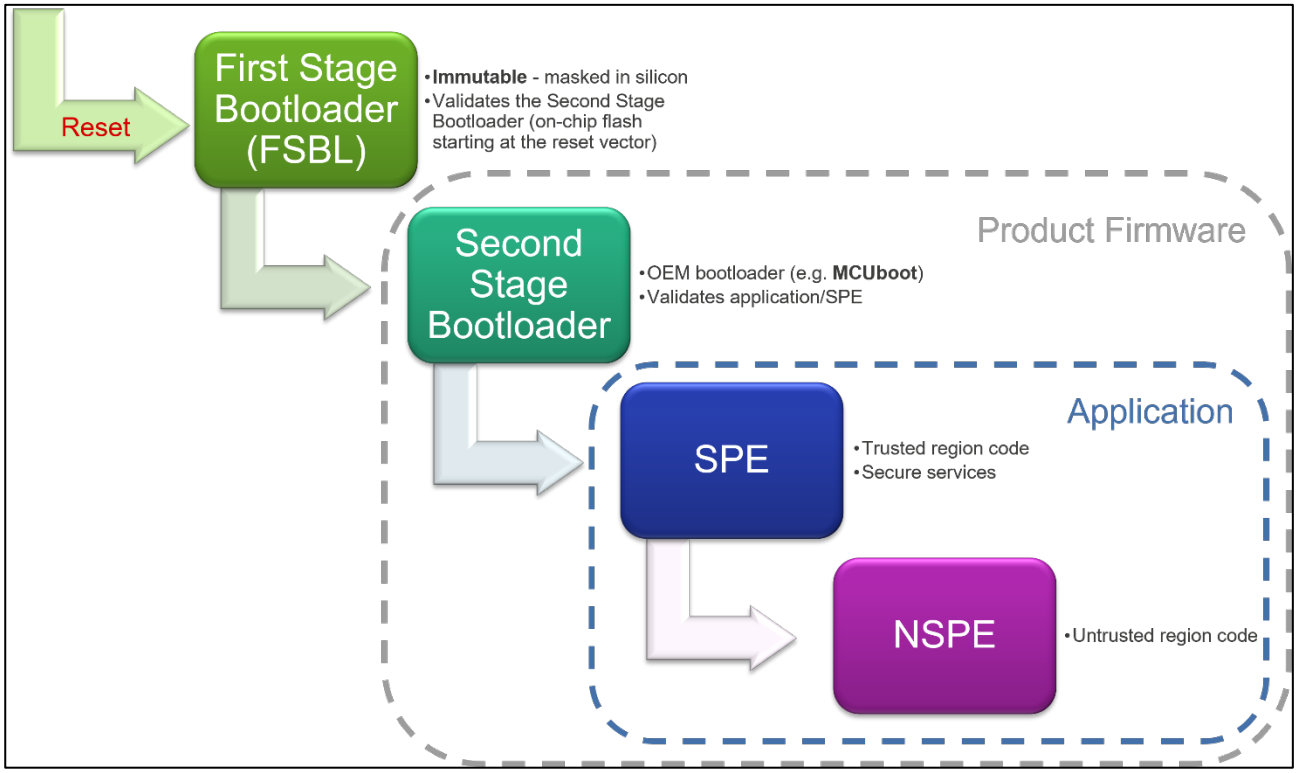

Figure 16. FSBL Used with SSBL and TrustZone

The RA8M1 FSBL is optional and highly configurable, including:

Operational Mode

- Signature authentication - When code starting at the beginning of program memory (often the second-stage bootloader) is programmed, it is signature checked for authenticity using an ECDSA secp256r1 signature. If the signature check is successful, an HMAC-SHA256 digest is created using the MCU's Hardware Unique Key (HUK) and stored on-chip. This digest is then verified upon subsequent resets of the MCU. This has the advantages that:
  - HMAC-SHA256 digest verification is significantly faster than ECDSA secp256r1 signature verification.
  - The HUK is securely stored on the MCU. It is unknown anywhere outside the MCU.
  - The HMAC-SHA256 algorithm is considered to be quantum-safe.
- Integrity check - When code starting at the beginning of program memory is programmed, the CRC of the specified region of code is calculated and confirmed. Upon subsequent resets of the MCU, this integrity check is repeated.
- Bypass – The FSBL is not enabled. Application execution starts at the beginning of program memory with no verification of that program memory.

The FSBL does impact application start-up time. CRC integrity check is faster than HMAC-SHA256 verification; however, a CRC check should not be considered a security solution.

Since all code flash on the RA8M1 is stored inside the chip, and the programmer and debugger interface should be locked upon deployment, use of the FSBL may not be required to maintain the security of the end-product. Be sure to examine the threat model of the application to determine what solution provides the best fit.

Verification Region

Since both verification methods impact application start-up time, the amount of program memory that is verified by the FSBL can be specified.

In an optimal security solution, all executable code would be authenticated by either the first-stage bootloader or the second-stage bootloader prior to execution. However, this is often not practical for microcontroller applications. Limiting the verification region and placing vital assets in that region is a potential way to balance security with efficiency.

Reset Handling

The FSBL can be configured such that verification is skipped after a software reset or deep sleep.

This option is provided with acknowledgment that the internal flash will likely not be unexpectedly modified under those conditions. Be sure to examine the threat model of the application to determine if this threat is out of scope.

Firmware Update

If the verified program memory region is updated, a new digest/CRC must be generated/provided and stored. The RSIP Protected Mode API `R_RSIP_FSBL_OEM_BL_Digest_Generate()` is provided in the FSP to generate the digest.

The FSBL utilises RSIP Protected Mode; therefore, any new digest must be created using RSIP in Protected Mode. There is no Compatibility Mode equivalent of `R_RSIP_FSBL_OEM_BL_Digest_Generate()`.

The FSBL employs an anti-rollback counter to prevent firmware downgrade. The ARC_OEMBL allows 64 versions (i.e., 63 potential updates) of the code flash that will be authenticated. It is vital to consider the operating environment of the end-product and determine how to architect a secure boot and secure update solution.

For more details, refer to the following documents:

- [RA8M1 User Manual]
- [FSP User Manual]
- [FSBL]

11.2 Second Stage Bootloader

The RA8M1 does not have a built-in (i.e., First Stage) secure bootloader; however, it is possible to implement an immutable bootloader using the secure bootloader solution provided in the FSP.

The FSP includes a port of the TrustedFirmware MCUboot project, an open-source secure boot and firmware update solution for 32-bit microcontrollers. Secure boot is usually very application-specific; therefore,

Renesas provides multiple examples of how the secure boot solution can be used. Note that secure boot and secure firmware updates are provided as a unified solution.

Review the threat model and start-up time requirements of the application when implementing secure boot. Since all code flash on the RA8M1 is stored inside the chip, and the programmer and debugger interface should be locked upon deployment, it may be sufficient to implement only the secure firmware update portion of the secure bootloader solution.

For more details, refer to the following documents:

- [Bootloader Basic]
- Section 7 Immutable Memory

12. Secure Firmware Update

The ability to update firmware in an end-product is becoming a standard requirement, especially if the product is connected to public infrastructure such as the internet. Any firmware update, whether through public infrastructure or a private connection, should be performed in a secure manner.

The FSP includes a port of the TrustedFirmware MCUboot project, an open-source secure boot and firmware update solution for 32-bit microcontrollers. Secure firmware update is usually very application-specific; therefore, Renesas provides multiple examples of how the secure firmware update solution can be used. Note that secure boot and secure firmware updates are provided as a unified solution.

There are many items to consider when devising a secure firmware update strategy, including:

How will the new image be received? Is it received over a secure channel, or does the image need to be encrypted for transfer to the product? A TLS connection can usually be considered a secure channel.

The authenticity of the image needs to be verified before it is programmed for execution. Therefore, the entire image must be received and stored locally. Will the new image be stored on-chip or in an external memory?

If the new image is stored externally prior to installation, does it need to be encrypted so that an attacker cannot physically extract the plaintext binary?

Be sure to consider the threat model for the end product when architecting a firmware update strategy.

If MCUboot is used without also using the FSBL, it is recommended to make the code flash blocks that contain MCUboot immutable to form the application's immutable Root of Trust.

For more details, refer to the following documents:

- [Bootloader Basic]
- Section 7 Immutable Memory

13. Provisioning

13.1 Device Lifecycle Management and Debug Authentication

The RA8M1 supports authenticated Device Lifecycle Management (DLM) and debug authentication to provide IP protection and support product failure analysis. Figure 17. Device Lifecycle Management Overview shows an overview. Note that debugger/programmer access is configurable only when in the OEM state.

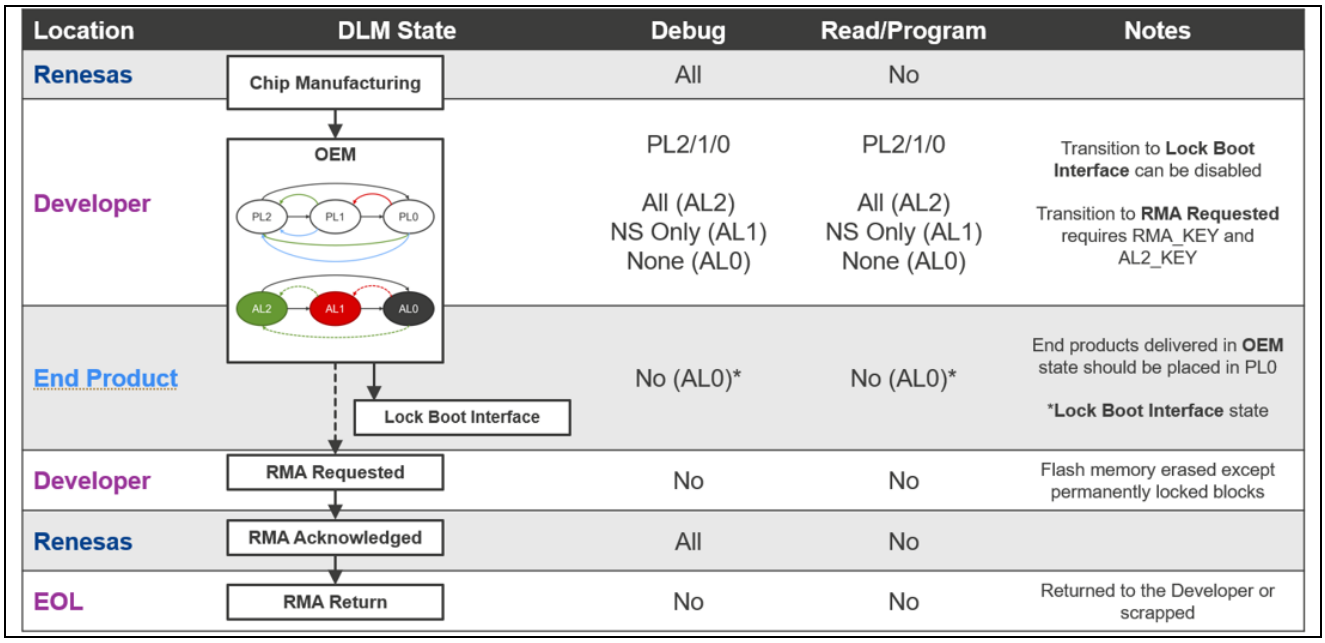

Figure 17. Device Lifecycle Management Overview

Protection Level (PL) and Authentication Level (AL) determine the availability of the debugger functionality and the programming interface. AL is the currently active authentication status. It is set to PL after an MCU power-on reset. Table 3. Available Debug and Programmer Functionality per Authentication Level describes the debugger and programmer functionality available in each Authentication Level.

Table 3. Available Debug and Programmer Functionality per Authentication Level

AL	Debug Function	Serial Programming Interface
AL2	Non-secure and Secure debug functions are enabled and accessible	All functions are available
AL1	Only non-secure debug functions are enabled. Debugger can access only defined non-secure debug accessible regions	Cannot program, erase, or read secure code or data flash areas
AL0	No debug functions are available	Cannot access code or data flash areas

Figure 18. PL and AL States and Transitions describes the conditions for changing the Authentication Level and the Protection Level.

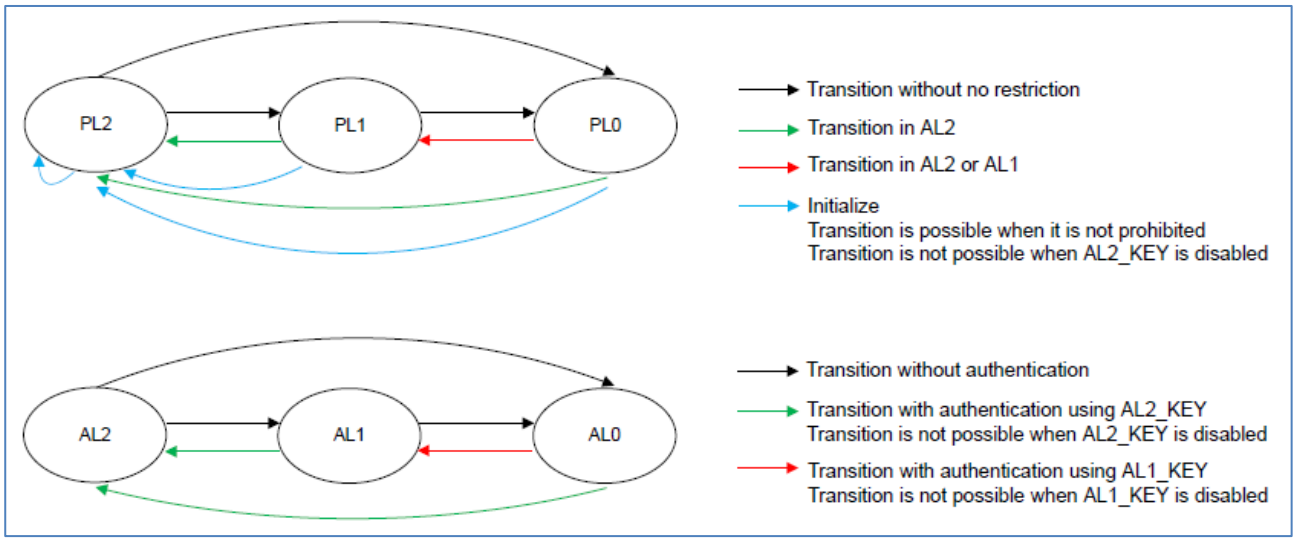

Figure 18. PL and AL States and Transitions

Blank MCUs are delivered in the OEM state, with the debug Protection Level set to PL2. In this state, the product developer has complete access to all user-accessible memory and peripherals. If the application does not utilise TrustZone as an isolation mechanism, the entire application is developed in this PL state. If the application does utilise TrustZone, the Secure region application code is programmed and debugged in this state.

If the application utilises TrustZone, the MCU should be transitioned to the Protection Level PL1 state to program and debug the Non-secure region application code.

When the MCU is programmed for incorporation into an end-product, there are several available options:

- If there is no desire to either re-enable the debugging capability of the MCU or maintain the ability to send the MCU to Renesas for failure analysis in the event of possible MCU failure, then the MCU should be transitioned to the Lock Boot Interface state. This transition is permanent.
- The MCU can remain in the OEM state to retain the ability to re-enable the debugging capability of the MCU and/or maintain the ability to send the MCU to Renesas for failure analysis.
    - To maintain the ability to send the MCU to Renesas for failure analysis in the event of possible MCU failure, an RMA_KEY must be injected when the device is in the PL2 state.
    - To maintain the ability to re-enable debugging of the Non-secure region, an AL1_KEY must be injected when the device is in the AL1 or AL2 state.
    - To maintain the ability to re-enable debugging of the Secure region, including re-enabling debugging of an application that does not utilise TrustZone as an isolation mechanism, then an AL2 _KEY must be injected when the device is in the AL2 state.
    - The MCU must be transitioned to Protection Level PL0 at the completion of production programming.

If a chip is returned to Renesas for failure analysis, the DLM state transition to RMA Requested (RMA_REQ) will not erase any blocks that have their associated PBPS or PBPS_SEC bit set. Information in these blocks will be visible to Renesas during the RMA process.

Key preparation steps where mass production keys are exposed in plaintext must be performed in a secure environment. Test keys used for prototype development can be created and prepared on a development PC.

The process of preparing and injecting DLM/AL keys is similar to the secure key injection process for Protected Mode, described in section. 4.3 Secure and Plaintext Key Injection and Update.

The most secure configuration is Lock Boot Interface; however, the application use case may require the ability to re-enable programming and/or debugging of the end-product. In this case, non-trivial keys that are not duplicated from Renesas examples or other publicly available cryptography references must be used.

Renesas provides the following tools to support this process:

- Security Key Management Tool (SKMT) - This tool can create sample keys and prepare the DLM keys for secure injection. [SKMT]
- Renesas Flash Programmer (RFP) - This tool can perform DLM state transition using the MCU's factory boot firmware. [RFP]

Select third-party programming tools also support secure key injection.

For more details, refer to the following documents and sections of this document:

- [RA8M1 User Manual]
- [DLM]
- [SKMT]
- [RFP]
- Section 4.3 Secure and Plaintext Key Injection and Update

13.2 Secure Factory Programming

The RA8M1 supports Secure Factory Programming (SFP), enabling secure production programming of the MCU in a non-secure environment.

It is not required to use Secure Factory Programming to create a secure end-product. However, Secure Factory Programming can be used as a component of a secure supply chain solution when production programming is performed in either a secure or non-secure environment.

Figure 19. Secure Factory Programming Overview shows an overview of the secure programming process. The developer must create an Image Encryption Key. This key must be used to encrypt the firmware image. The Image Encryption Key must be wrapped by a UFPK.

Key preparation steps where keys are exposed in plaintext must be performed in a secure environment.

At the completion of the SFP process, the MCU can be either completely locked or left in the OEM DLM state with a provisioned AL2_KEY. If an AL2_KEY is being provisioned, it must be wrapped with the same UFPK that was used to wrap the Image Encryption Key.

Only AL2_KEY injection is supported; AL1_KEY injection is not supported.

If the MCU is left in the OEM DLM state, it is recommended to disable the Initialize command during production programming.

If the MCU is left in the OEM DLM state, an RMA_KEY can be injected; however, RMA_KEY injection is not included within the Secure Factory Programming process. The security risk of a malicious RMA_KEY being injected during the programming process must be assessed.

The W-UFPK, the wrapped AL2_KEY (optional), the wrapped Image Encryption Key, and the encrypted firmware image are provided to the production programming facility. These items are then used in conjunction with the MCU's factory boot firmware to securely program the MCU.

Non-trivial keys that are not duplicated from Renesas examples or other publicly available cryptography references must be used for the UFPK, the Image Encryption Key, and the AL2_KEY (optional).

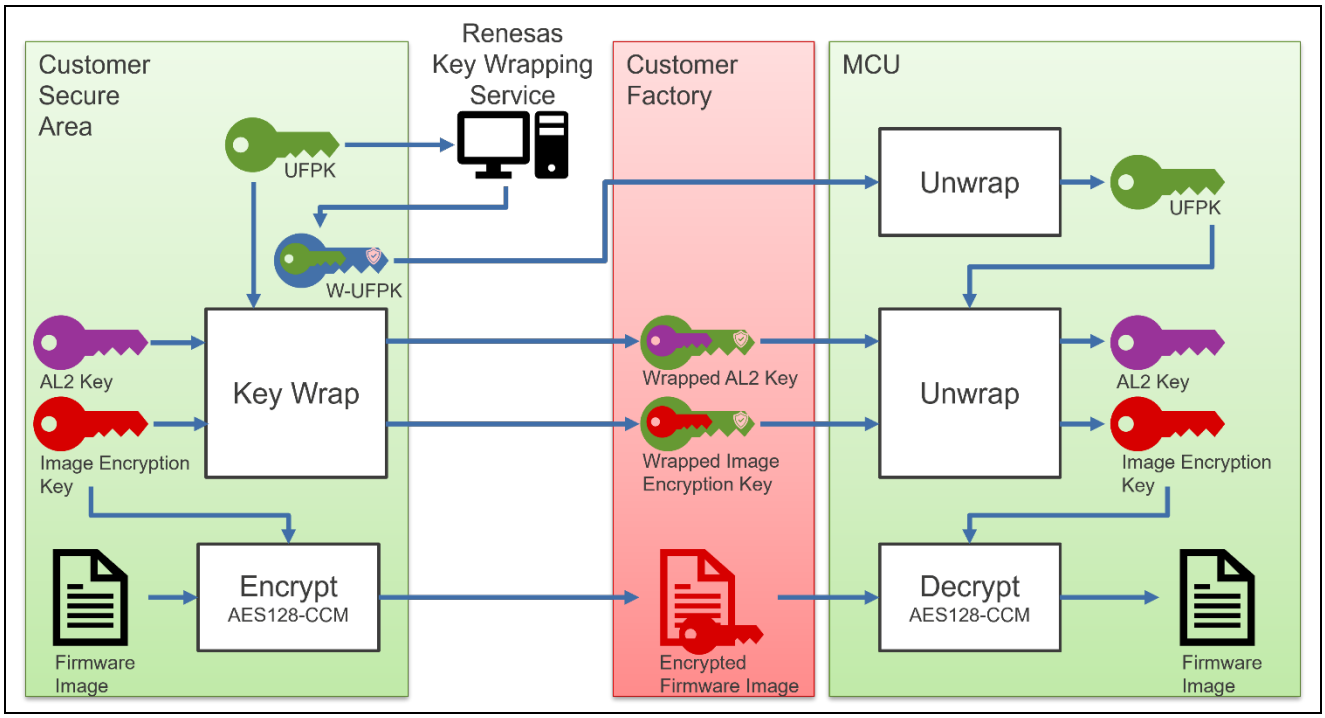

Figure 19. Secure Factory Programming Overview

Secure Factory Programming can be combined with RSIP-E51A Protected Mode secure key injection for provisioning and personalisation.

User application key injection can be performed with Secure Factory Programming; however, the W-UFPK used for user application key injection can be different from the W-UFPK used for Secure Factory Programming. The security risk of potential malicious key injection must be assessed.

Secure Factory Programming has a limitation that application keys cannot be made immutable via the Permanent Block Protect Setting registers. If securely injected RSIP-E51A Protected Mode keys need to be made immutable, the application code must set the Permanent Block Protect Setting registers when it executes for the first time.

RA8M1 Secure Factory Programming erases on-chip code flash and data flash prior to programming, but it does not perform an initialization. To ensure that the chip is programmed exactly as required, the complete Option-setting Memory contents should be included in the firmware image. This output is generated by changing the configurator settings from Disabled to Enabled and setting the individual options as appropriate (see Figure 20. Code Flash Option-setting Memory Configuration).

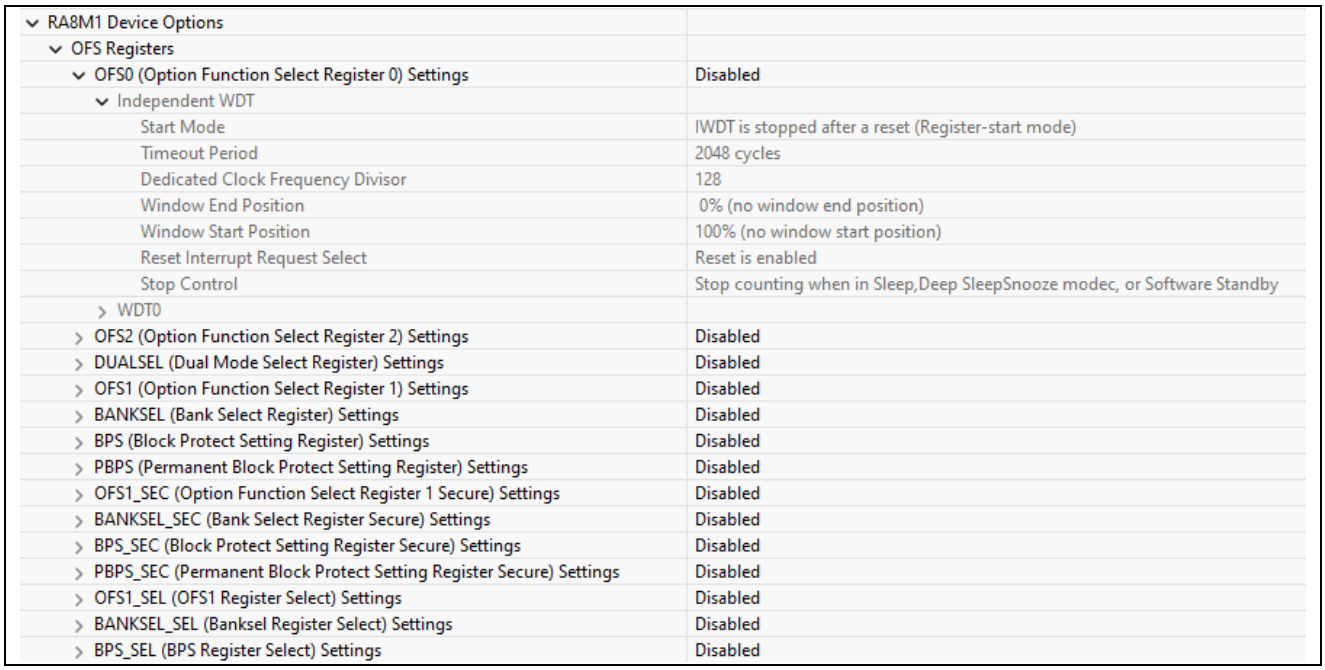

Figure 20. Code Flash Option-setting Memory Configuration

Renesas provides the following tools to support this process:

- Security Key Management Tool (SKMT) - This tool can create sample keys and prepare the firmware image for secure factory programming. [SKMT]
- Renesas Flash Programmer (RFP) - This tool can perform Secure Factory Programming using the MCU's factory boot firmware. [RFP]

Select third-party programming tools also support Secure Factory Programming.

For more details, refer to the following documents and sections of this document:

- [RA8M1 User Manual]
- [Boot Firmware]
- [Secure Factory Programming]
- [SKMT]
- [RFP]
- Section 4 Secure Key Injection, Storage, and Usage
- Section 13.1 Device Lifecycle Management and Debug Authentication

14. Website and Support

Visit the following URLs to learn about key elements of the RA family, download components and related documentation, and get support:

RA Product Information	renesas.com/ra
RA8M1 Product Information	renesas.com/RA8M1
RA Product Support Forum	renesas.com/ra/forum
RA Flexible Software Package	renesas.com/FSP
Renesas Support	renesas.com/support
Renesas PSIRT	renesas.com/psirt

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Nov.1.25	—	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/.