RZ/G2 Group

Secure IP Driver API Specification

Introduction

This document describes Secure IP Driver API provided for RZ/G2 group.

This document is a manual intended for users who plan to develop their own security software using Secure IP Driver. Please also refer to "RZ/G2 Trusted Execution Environment Porting Guide".

Target Device

RZ/G2E

RZ/G2M

RZ/G2N

RZ/G2H

Contents

1. Overview

Secure IP Driver provides security features using the on-chip Trusted Secure IP (hereinafter referred to as “TSIP”) included with RZ/G2 group.

1.1 Features

Secure IP Driver provides APIs for the following purpose:

- Secure Boot
- Secure Update
- Basic Cryptographic

- Secure Boot

Secure Boot is a function that verifies user data loaded during a boot process. This function ensures that loaded user data is trusted.

The user data is encrypted before it is stored in the boot device ROM. Secure Boot detects unauthorized tampering with user data by decrypting and verifying the encrypted user data.

Secure IP Driver provides the following functions for the Secure Boot mechanism.

- Encryption of keyring and user data
- Decryption and verification of keyring and user data

Secure Boot is required to use the features of Secure Update and Basic Cryptographic. Since TSIP is not activated if Secure Boot fail, these features are not available.

- Secure Update

Secure Update is a function to securely update keyring and user data. This function is used to update keyring and user data in products shipped to the market.

- Basic Cryptographic

Secure IP Driver provides the following for Basic Cryptographic functions:

- Random Number Generation.

1.1.1 Key Features

In Secure IP driver, TSIP is used to implement security functions. TSIP has a unique key for each device (device-specific key) and this device-specific key is used to encrypt and decrypt various data. The device-specific key in TSIP cannot be accessed from outside, so it has a high tamper resistance.

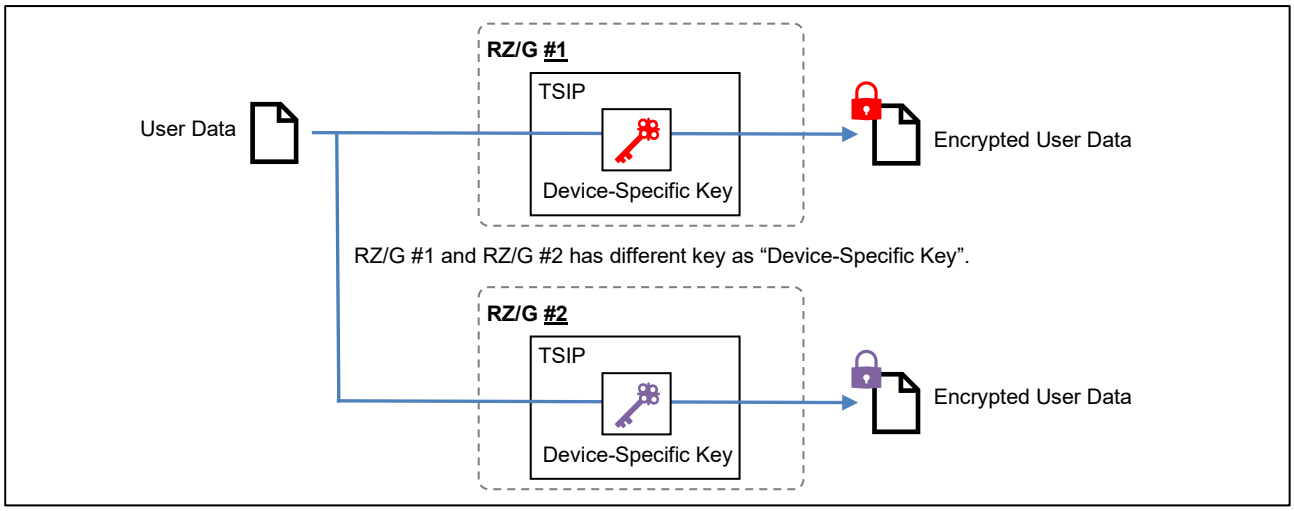

Figure 1-1. Schematic View of Encryption with Device-Specific Key

Data encrypted with the device-specific key can be decrypted only by the device that encrypted the data. Even when keyring and user data are stored in non-volatile memory outside the device, encrypting each data with the device-specific key ensures safe operation. Unauthorized copying of keyring or user data in non-volatile memory to another product cannot be decrypted.

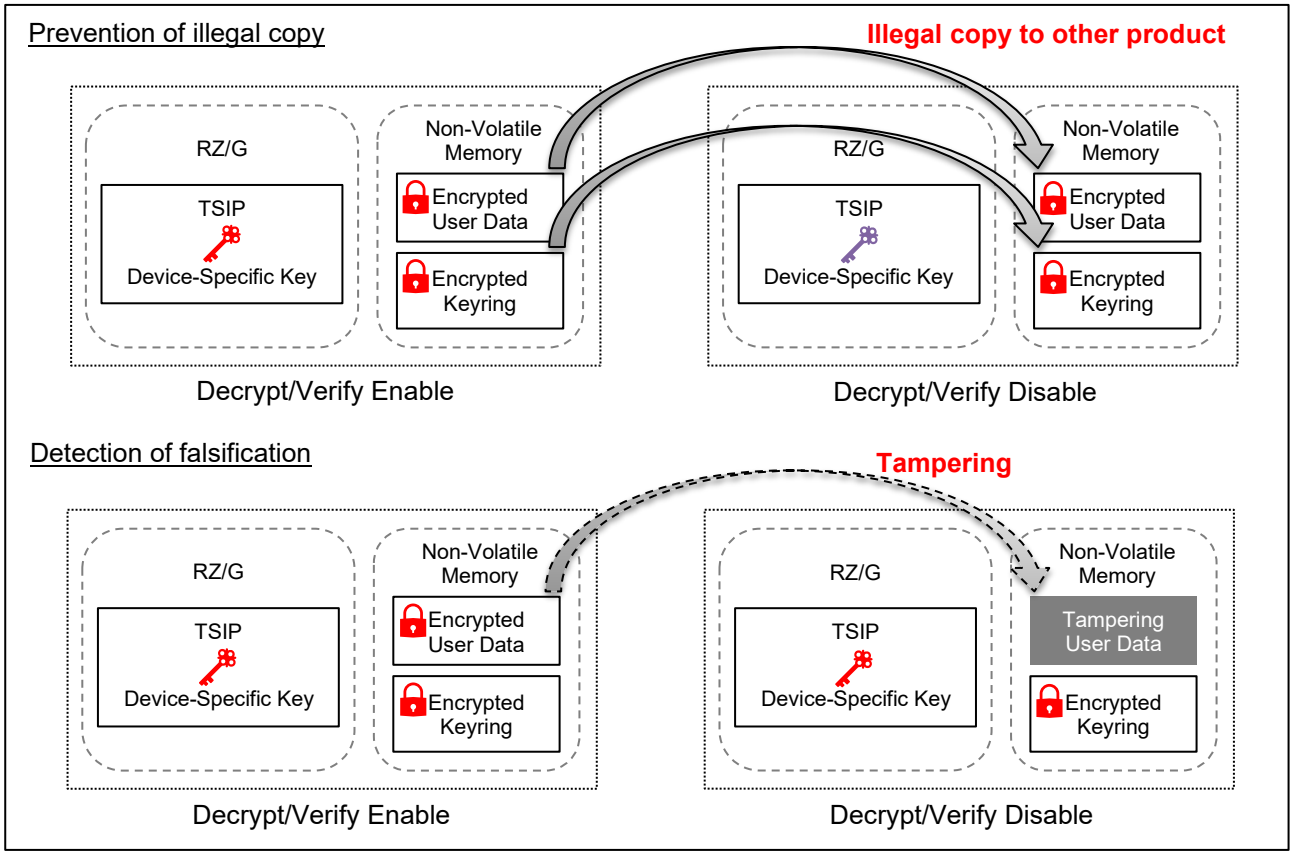

Figure 1-2. Schematic View of Encryption with Device-Specific Key

1.2 Related Documents

Table 1-1. Related Documents

No	Issue	Title
1	Renesas Electronics	RZ/G2 Trusted Execution Environment Start-Up Guide
2	Renesas Electronics	RZ/G2 Trusted Execution Environment Porting Guide

1.3 Terms

Table 1-2. Terms

No		
1	Trusted Secure IP (TSIP)	Security IP in RZ/G devices.
2	Temporary Encryption	Encryption to convert keyring and user data to be verified by Secure Boot to the format handled by TSIP.
3	Re-encryption	Decryption the temporarily encrypted data with TSIP and encrypt again with device-specific key with TSIP.
4	Provisioning Process	Process of Temporary Encryption and Re-encryption with TSIP and store re-encrypted data to non-volatile memory. For using Security IP Driver, provisioning process must be performed in advance.

2. Provided Software

Secure IP Driver source code is included in Yocto build environment provided for RZ/G2 group. For the Yocto build environment, please refer to “Related Documents No1”.

Security Module and OP-TEE OS have a built-in Secure IP Driver. In the Yocto build environment, the driver is stored in the following directory.

- Security Module
  [${WORK}/build/tmp/work/<work-sub-directories>/secmod/<Properties-of-yocto-environment>/git/tsip]
- OP-TEE OS
  [${WORK}/build/tmp/work/<work-sub-directories>/optee-os/<Properties-of-yocto-environment>/git/core/arch/arm/plat-rzg/driver/tsip]

3. Functions

The following functions can be implemented using the API provided by the Secure IP Driver.

3.1 Secure Boot

Secure Boot is a mechanism for detecting data tampering by decrypting and verifying signed and encrypted data. The following is an example of an environment that implements Secure Boot.

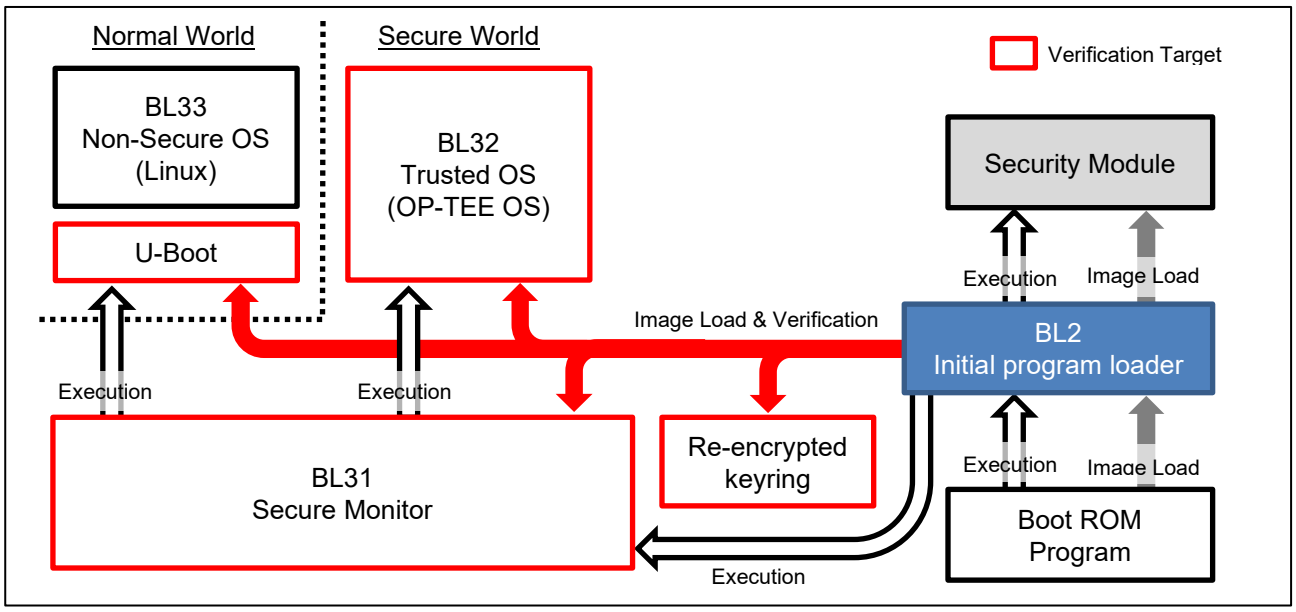

Figure 3-1. Secure Boot sequence example

BL2 is a program for loading user data (firmware image) and implements Secure Boot. BL2 decrypts and verifies user data loaded from the boot device ROM and places it in RAM.

Security Module is a program that accesses TSIP and includes Secure IP Driver. BL2 calls Security Module to decrypt and validate encrypted user data.

BL31, BL32, U-Boot, and re-encrypted keyring are decrypted and verified by Secure Boot. These data are signed and encrypted before being stored in non-volatile memory.

Secure IP Driver provides the following functions for the Secure Boot mechanism.

Table 3-1. Provided functions of Secure IP driver for the Secure Boot mechanism

Function	Description
Re-encryption of Keyring	Re-encrypt temporarily encrypted keyring using the device-specific key in TSIP.
Re-encryption of User Data	Re-encrypt temporarily encrypted user data using the device-specific key in TSIP.
Verification of Keyring	Verify re-encrypted keyring and activate TSIP.
Decryption and Verification of User Data	Decrypt and Verify re-encrypted user data and activate TSIP.

3.2 Provisioning

To implement Secure Boot in a user product environment, the encrypted keyring and encrypted user data must be prepared and stored in non-volatile memory. The process from preparing keyring and user data in an external environment to encrypting data with the device-specific key in a user product environment is called provisioning.

The following is an example of provisioning sequence.

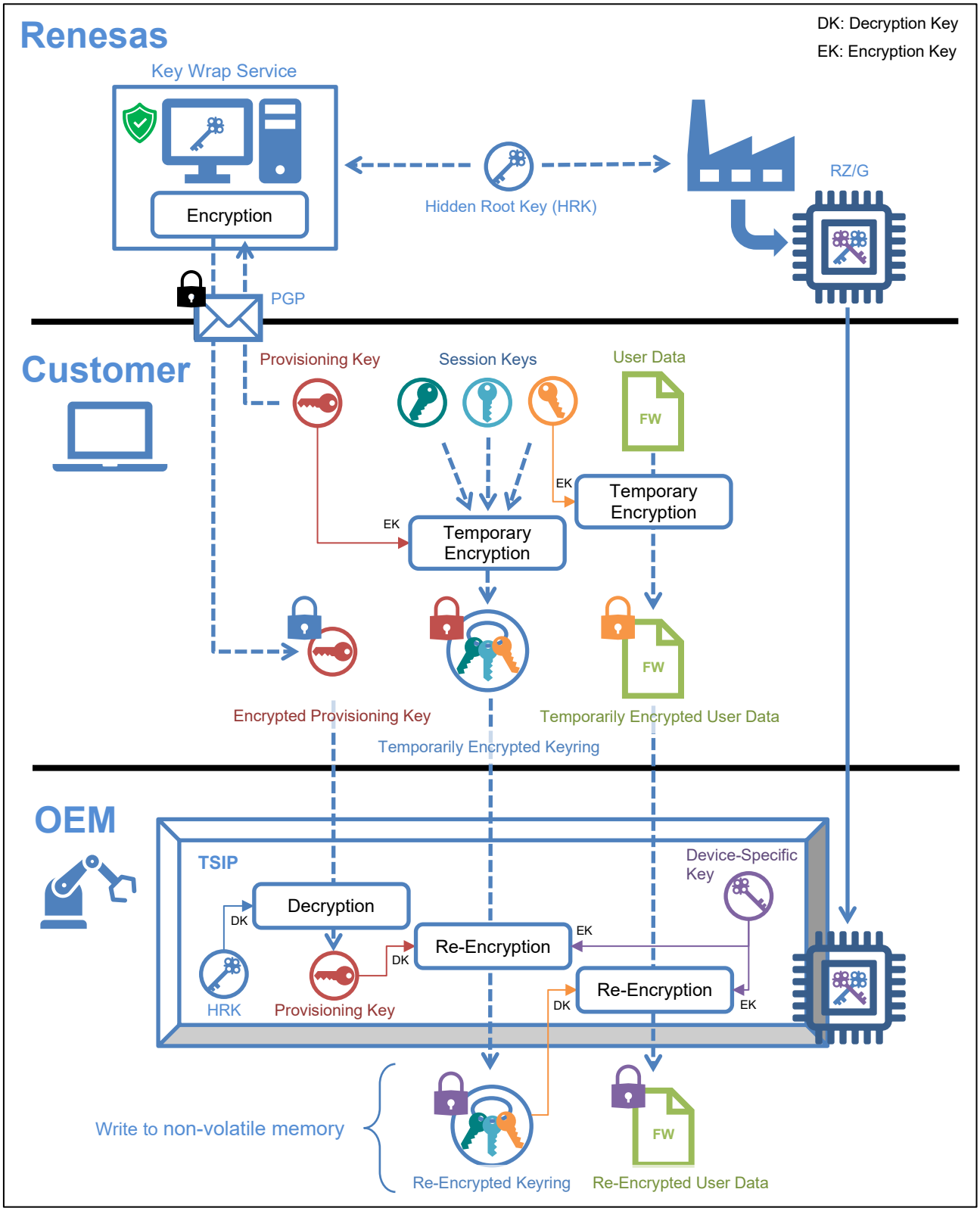

Figure 3-2. Provisioning sequence example

3.2.1 Key and Keyring

The keys used for provisioning are shown below.

Table 3-2. Keys used for Provisioning

Name	Description	Creator
Hidden Root Key (hereinafter referred to as HRK)	A key managed within Renesas, and this key is not provided to user. HRK exists only inside the Key Wrap Service and TSIP. Used to encrypt and decrypt the provisioning key.	Renesas
Device-Specific Key	A unique key that exists only inside TSIP. All data verified by Secure Boot is encrypted with this key.	
Provisioning Key	A key used to temporarily encrypt the keyring. By encrypting keyring with this key prevents the session keys from leaking between the external environment and the user product environment.	User
Encrypted Provisioning Key	A provisioning key encrypted with the HRK. It can be encrypted using Key Wrap Service provided by Renesas. Used to re-encrypt temporarily encrypted keyring in the user product environment.	
Session Keys	Keys used to temporarily encrypt the keys and user data prepared in the external environment. By encrypting the data with this key prevents leakage and tampering data between the external environment and the user product environment.	
Keyring	A bunch of the session keys.	
Temporarily Encrypted Keyring	Keyring encrypted with the provisioning key.	
Re-Encrypted Keyring	Keyring re-encrypted with the device-specific key.	

The following sections describe key to create in the user environment.

(a) Provisioning Key

The provisioning key is a concatenation of two keys. The format of the provisioning key is shown below.

Table 3-3. Format of Provisioning Key

Key		Algorithm	Size (Byte)
Provisioning Key	Temporary Encryption Key for Keyring	AES-128	16
	MAC Key for Keyring	AES-128	16

(b) Encrypted Provisioning Key

The encrypted provisioning key is required to re-encrypt temporarily encrypted keyring in the user product environment. It can be created by encrypting the provisioning key using the Key Wrap Service provided by Renesas. For information about how to encrypt the provisioning key, please refer to “Related Documents No.2”.

(c) Session Keys and Keyring

Keyring is a bunch of the session keys, is 672 bytes of binary data. The session key is used to temporarily encrypt the key and user data prepared in the external environment. The format of keyring is shown below.

Table 3-4. Format of Keyring

Session Keys		Algorithm	Size (Byte)
Reserved		(fixed 0)	32
for Secure Boot*1	Temporary Encryption Key for User Data	AES-128	16
		IV0	16
	Temporary Verification Key for Signature of User Data*2	RSA Public Key(n)	256
		RSA Public Key(0^15 Padding || e || 0^96 Padding)	16
Reserved		-	272
for Secure Update	Temporary Encryption Key for Keyring	AES-128	16
	MAC Key for Keyring	AES-128	16
Reserved		-	32

Notes: 1. If decryption/verification of user data is not needed in Secure Boot (it is means that Secure Boot is performed with only Verification of Keyring), pad this area with 0.
2. The private key that is pair of this key (Temporary Verification Key for Signature of User Data) is used for signing user data in the provisioning process.

(d) Temporarily Encrypted Keyring

The temporarily encrypted keyring is data in which keyring is encrypted using the provisioning key.

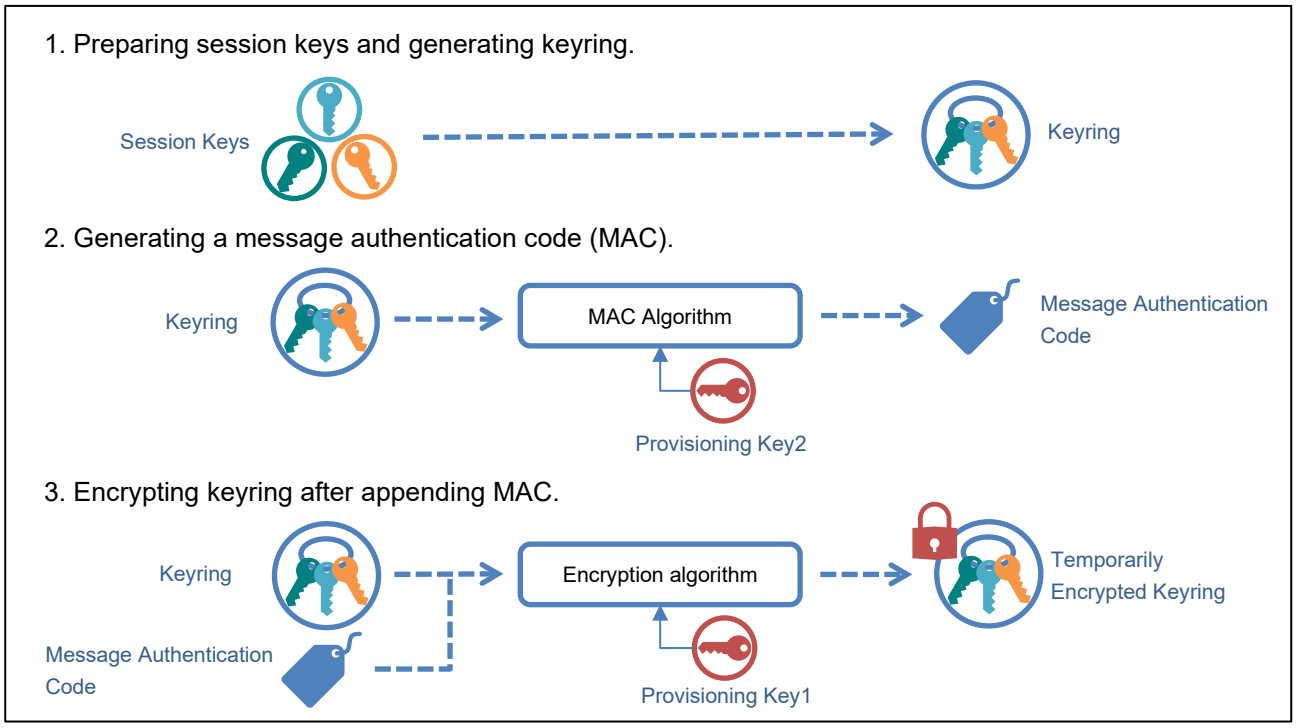

Figure 3-3. Schematic View of Temporary Encryption of Keyring

The following show the algorithm used for MAC generation and encryption in Temporary Encryption of Keyring.

Table 3-5. Algorithm and Key in Temporary Encryption of Keyring

Process	Algorithm	Key	IV
MAC	CBC-MAC with AES-128	Provisioning Key2 (MAC Key for Keyring)	0
Encryption	AES-128-CBC	Provisioning Key1 (Temporary Encryption Key for Keyring)	IV0*

Notes: IV0 = 0x85c1673483d5d291f0d0713e3ea434a3

(e) Re-Encrypted Keyring

The re-encrypted keyring is data in which the temporarily encrypted keyring is re-encrypted. The re-encryption process is performed using Secure IP Driver on the target product.

The re-encrypted keyring is used for Re-encryption of User Data, Verification of Keyring and Decryption and Verification of User Data.

3.2.2 User Data

User data used for provisioning are shown below.

Table 3-6. User Data for Provisioning

Name	Description
Temporarily Encrypted User Data	User data encrypted with the session keys for Secure Boot.
Re-Encrypted User Data	User data re-encrypted with the device-specific key.

(a) Temporarily Encrypted User Data

The temporarily encrypted user data is data in which user data is encrypted using a session key for Secure Boot.

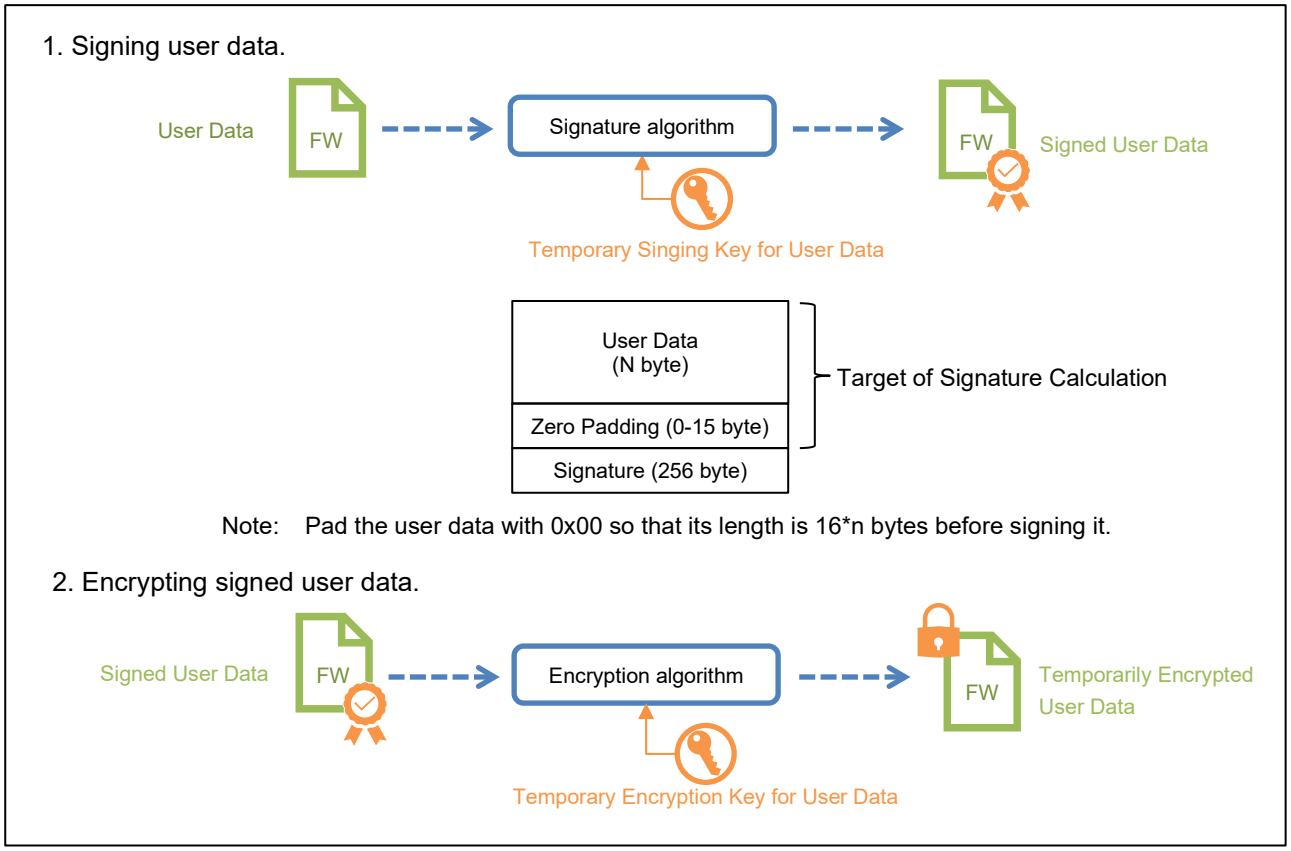

Figure 3-4. Schematic View of Temporary Encryption of User Data

The following show the encryption algorithm used for sign processing and encryption in Temporary Encryption of User Data.

Table 3-7. Algorithm and Key in Temporary Encryption of User Data

Process	Algorithm	Key	IV
Signing	RSASSA-PKCS1-v1_5 SHA256	Temporary Signing Key for User Data	0
Encryption	AES-128-CBC	Temporary Encryption Key for User Data	IV0*

Note: IV0 for Temporary Encryption Key for User Data (for Secure Boot) in keyring

(b) Re-Encrypted User Data

The re-encrypted user data is data in which the temporarily encrypted user data is re-encrypted. The re-encryption process is performed using Secure IP Driver on the target product. It is necessary to prepare the re-encrypted keyring in advance for this process.

3.3 Secure Update

After starting the operation of the product, it is possible to create data for updating each, to update keyring and user data re-encrypted. Keyring and user data to be updated are brought into the product after being temporarily encrypted in the external environment, re-encrypted with TSIP, and output. Update output data by replacing the existing data. New keyring and user data are brought into the product as encrypted data and re-encrypted inside TSIP, so keyring and user data update safely.

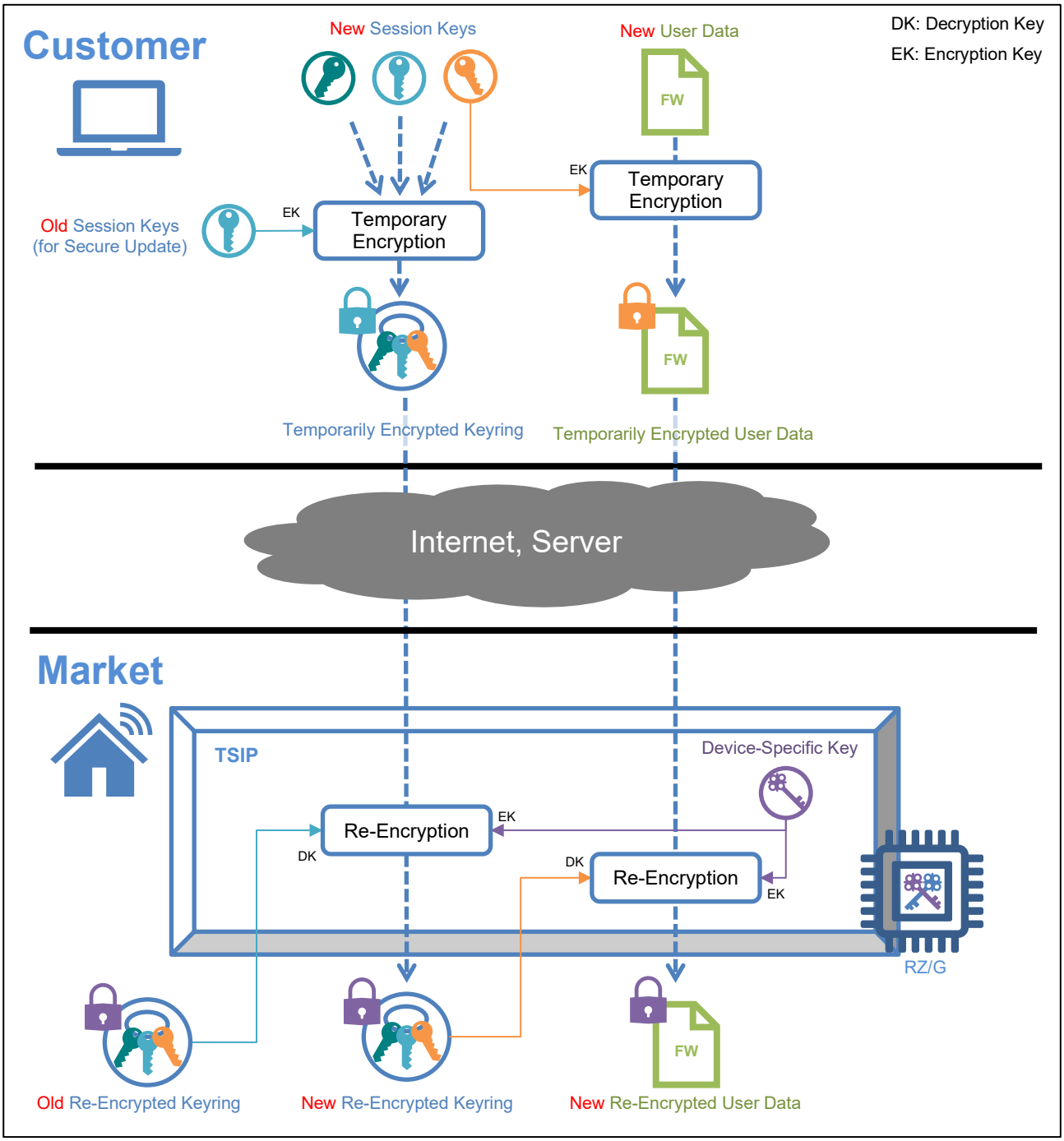

Figure 3-5. Secure Update sequence example

3.3.1 Temporary Encryption

(a) Temporarily Encrypted Keyring

Temporary encryption of Keyring for the update uses the session keys (for Secure Updates) of the re-encrypted keyring already included with the product. The temporarily encrypted keyring is created in the same way as MAC addition and encryption of keyring in the provisioning.

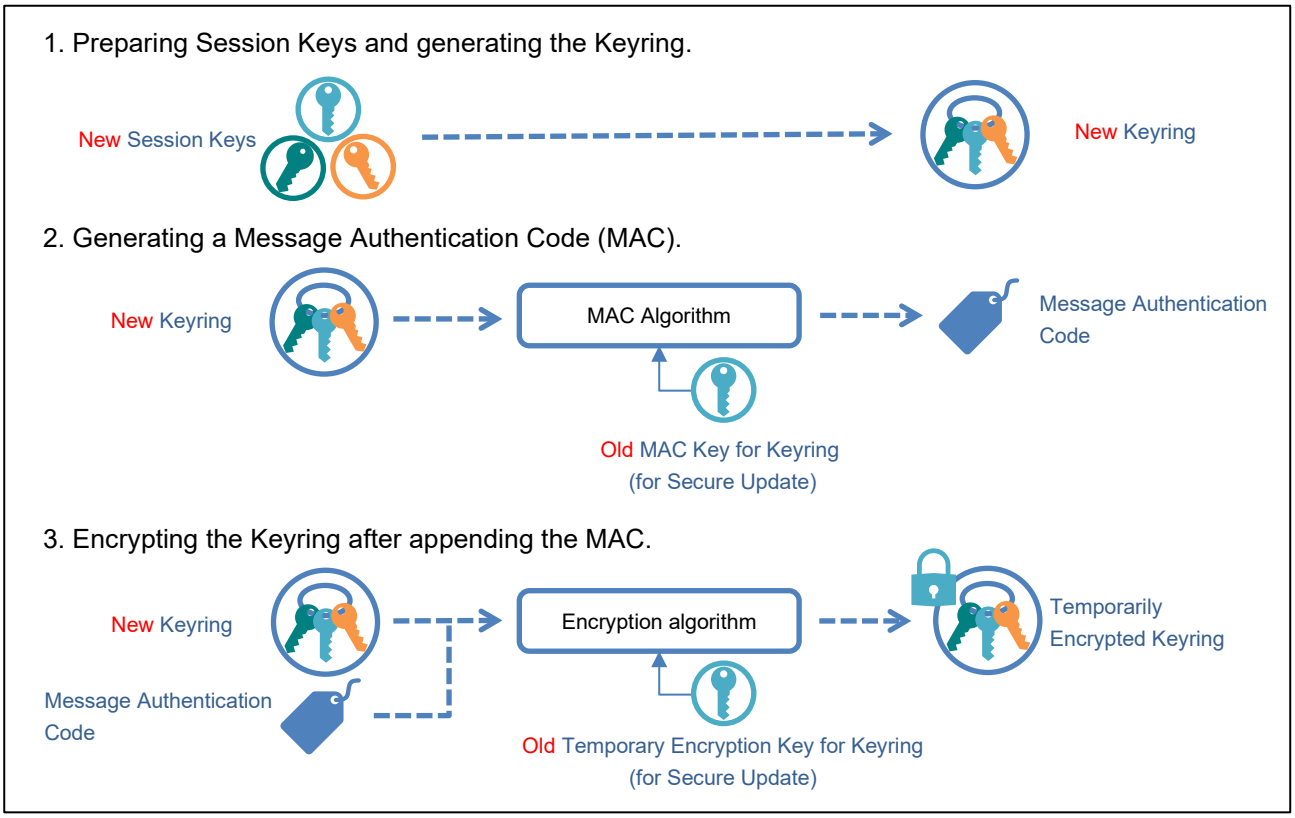

Figure 3-6. Schematic View of Temporary Encryption of Keyring for Secure Update

(b) Temporarily Encrypted User Data

The temporarily encrypted user data for Secure Update is encrypted in the same way as provisioning. If keyring is updated, Temporary Encryption of User Data for Secure Update uses the session keys of the updated keyring.

3.3.2 Re-Encryption

The temporarily encrypted keyring and user data are brought into the product and then re-encrypted in TSIP. The re-encryption process is performed using Secure IP Driver on the target product.

3.4 Basic Cryptographic

The Basic Cryptographic function is activated when all user data is successfully verified by Secure Boot.

Table 3-8. Basic Cryptographic Functions

Function	Standard
Random Number Generation	NIST SP800-90A

4. Specification

4.1 List of APIs

A list of APIs provided by Secure IP Driver to user application is shown below.

Table 4-1. Initialization API

No	Function Name	Description
1	R_TSIP_Init	Initialization of Secure IP Driver for Secure Boot.
2	R_TSIP_Lib_Init	Initialization of Secure IP Driver.

Table 4-2. Secure Boot API

No	Function Name	Description
1	R_TSIP_Inject_Key	Re-encryption of Keyring.
2	R_TSIP_ReEncBootData	Re-encryption of User Data.
3	R_TSIP_VerifyBootData	Decryption and Verification of User Data.
4	R_TSIP_KeyRing_Verify	Verification of Keyring.

Table 4-3. Secure Update API

No	Function Name	Description
1	R_TSIP_SU_Activate	Activation of Secure Update.
2	R_TSIP_UpdateBootData	Making Update Data for Secure Boot.
3	R_TSIP_SU_Key	Making Update Keyring for Secure Boot.

Table 4-4. Basic Cryptographic API

No	Function Name	Description
1	R_TSIP_BCF_GenerateRandom	Random Number Generation.

4.2 Data Definition

4.2.1 Return Value

A list of return values is shown below.

Table 4-5. Return Value

No	Definition	Value	Description
1	R_PASS	0x00000000	Success.
2	R_INITIALIZATION_FAIL	0x00000001	Failed to initialize Secure IP Driver. Retry if necessary.
3	R_PARAMETER_FAIL	0x00000002	Invalid argument. Specify correct argument.
4	R_SEQUENCE_FAIL	0x00000003	Invalid driver status. Execute after the correct state transition.
5	R_RESOURCE_CONFLICT_FAIL	0x00000004	Resource conflict. Resolve the resource conflict and retry.
6	R_VERIFICATION_FAIL	0x00000005	Failed to verify. Execute with correct verification data.
7	R_USEKEY_FAIL	0x00000009	Invalid key data Execute with correct key data.
8	R_PROVISIONING_KEY_FAIL	0x0000000C	Invalid a provisioning key. Use the correct provisioning key.
9	R_KEYRING_FORMAT_FAIL	0x0000000E	Invalid keyring format. Use the correct keyring.
10	R_MMAP_FAIL	0x00000021	Failed to map memory. Map system resources correctly.
11	R_FALSIFICATION_ERROR	0x00000022	Detected tampering. Reset hardware.
12	R_INITIALIZATION_ERROR	0x00000080	TSIP Hardware initialization error. Reset hardware.

4.2.2 Constants

A list of constant values is shown below.

Table 4-6. Constants

No	Definition	Value	Description
1	MSTP_BASE_ADDR	0xE6150000	Address of register for Module Standby.
2	MSTP_SIZE	0x00001000	Mapping size of register for Module Standby.
3	TSIP_BASE_ADDR	0xE7800000	Address of register for TSIP.
4	TSIP_SIZE	0x00000200	Mapping size of register for TSIP.
5	REENC_BOOT_DATA_MAX	16	Maximum value of user data for Secure Boot.
6	UPDATE_BOOT_DATA_MAX	16	Maximum value of user data for Secure Update.

4.2.3 Structure

(1) TSIP_REENC_BOOT_DATA

Format

```
typedef struct str_tsip_reenc_boot_data {
    unsigned char *InData_BootData;
    unsigned long  InData_BootData_ByteSize;
    unsigned char *OutData_BootData;
} TSIP_REENC_BOOT_DATA[REENC_BOOT_DATA_MAX];
```

Member

Member Name	IN/OUT	Description
InData_BootData	IN	Pointer to target data of re-encryption.
InData_BootData_ByteSize	IN	Size of target data of re-encryption.
OutData_BootData	OUT	Pointer to store re-encrypted data.

Description

This structure specifies data to re-encrypt the target to be decrypted and verified in Secure Boot.

It is necessary to prepare a member for each user data and specify it as a 16-element array. Each member is stored the address and size of each user data. If the number of user data is less than 16, the remaining element members must be set to NULL or 0.

Relation

- R_TSIP_ReEncBootData

(2) TSIP_UPDATE_BOOT_DATA

Format

```
typedef struct str_tsip_update_boot_data {
    unsigned long InData_BootData_UpdateFlag;
    unsigned char *InData_BootData;
    unsigned long InData_BootData_ByteSize;
    unsigned char *OutData_BootData;
} TSIP_UPDATE_BOOT_DATA[UPDATE_BOOT_DATA_MAX];
```

Member

Member Name	IN/OUT	Description
InData_BootData_UpdateFlag	IN	Flag that indicates whether to update.
InData_BootData	IN	Pointer to target data of re-encryption.
InData_BootData_ByteSize	IN	Size of target data of re-encryption.
OutData_BootData	OUT	Pointer to store re-encrypted data.

Description

This structure specifies data to update the target to be decrypted and verified in Secure Boot.

It is necessary to prepare a member for each user data and specify it as a 16-element array. Each member stores the address and size of each user data to update. The index of the element that stores the information of update data must be the same as the index when it was re-encrypted in provisioning.

Each user data can be updated individually. Set the InData_BootData_UpdateFlag of the user data to be updated to 1. Set the InData_BootData_UpdateFlag of user data that is not to be updated to 0.

Relation

- R_TSIP_UpdateBootData

4.3 Initialization API Details

This section describes APIs related to initialization.

4.3.1 R_TSIP_Init

R_TSIP_Init			Initialization
Initialization of Secure IP Driver for Secure Boot			
Header	R_TSIP_Boot_Lib.h		
Declaration	unsigned long R_TSIP_Init(unsigned char *S_RAMData, unsigned char *S_INSTData);		
Argument	IN	*S_RAMData	Pointer to the start address of the area in RAM used by TSIP.
	IN	*S_INSTData	Pointer to the start address of the area where the re-encrypted keyring is placed.
Return Value	R_PASS		Success.
	R_INITIALIZATION_FAIL		Failed to initialize Secure IP Driver.
	R_INITIALIZATION_ERROR		Failed to initialize TSIP.
	R_RESOURCE_CONFILICT_FAIL		Resource conflict.
	R_PARAMETER_FAIL		Invalid argument.
Description	This function initializes Secure IP Driver. Executing this function also initializes TSIP. This function must be executed before the function of Secure Boot API. The area of the S_RAMData requires 480 bytes. The area of the S_INSTData requires 1296 bytes. When the Secure Boot API are executed, this area is referred. How to use this area depends on the functions of Secure Boot API. For more information, please refer to each functions of Secure Boot API.		

4.3.2 R_TSIP_Lib_Init

R_TSIP_Lib_Init Initialization

Initialization of Secure IP Driver

Header	R_TSIP_Core_Lib.h		
Declaration	unsigned long R_TSIP_Lib_Init(unsigned char *S_RAMData, unsigned char *S_INSTData, unsigned char *TSIP_BaseAddress, unsigned char *MSTP_BaseAddress);		
Argument	IN	*S_RAMData	Pointer to the start address of the area in RAM used by TSIP.
	IN	*S_INSTData	Pointer to the start address of the area where the re-encrypted keyring is placed.
	IN	*TSIP_BaseAddress	Pointer to the TSIP base address that is mapped to the address space of the execution environment.
	IN	*MSTP_BaseAddress	Pointer to the MSTP base address that is mapped to the address space of the execution environment.
Return Value	R_PASS		Success.
	R_PARAMETER_FAIL		Invalid argument.
	R_SEQUENCE_FAIL		Invalid driver state.
	R_MMAP_FAIL		Failed to map memory.
	R_FALSIFICATION_ERROR		Detected tampering.
Description	This function prepares to start the use of the Secure Update API and Basic Cryptographic API. This function must be executed after validation by the Secure Boot API's R_TSIP_VerifyBootData() or R_TSIP_KeyRing_Verify() functions. The area of the S_RAMData requires 480 bytes. The area of the S_INSTData requires 1296 bytes. For TSIP_BaseAddress, specify the address that maps TSIP_BASE_ADDR to the execution environment. For MSTP_BaseAddress, specify the address that maps MSTP_BASE_ADDR to the execution environment.		

4.4 Secure Boot API Details

This section describes APIs related to Secure Boot.

State Transitions of Secure Boot APIs is shown below.

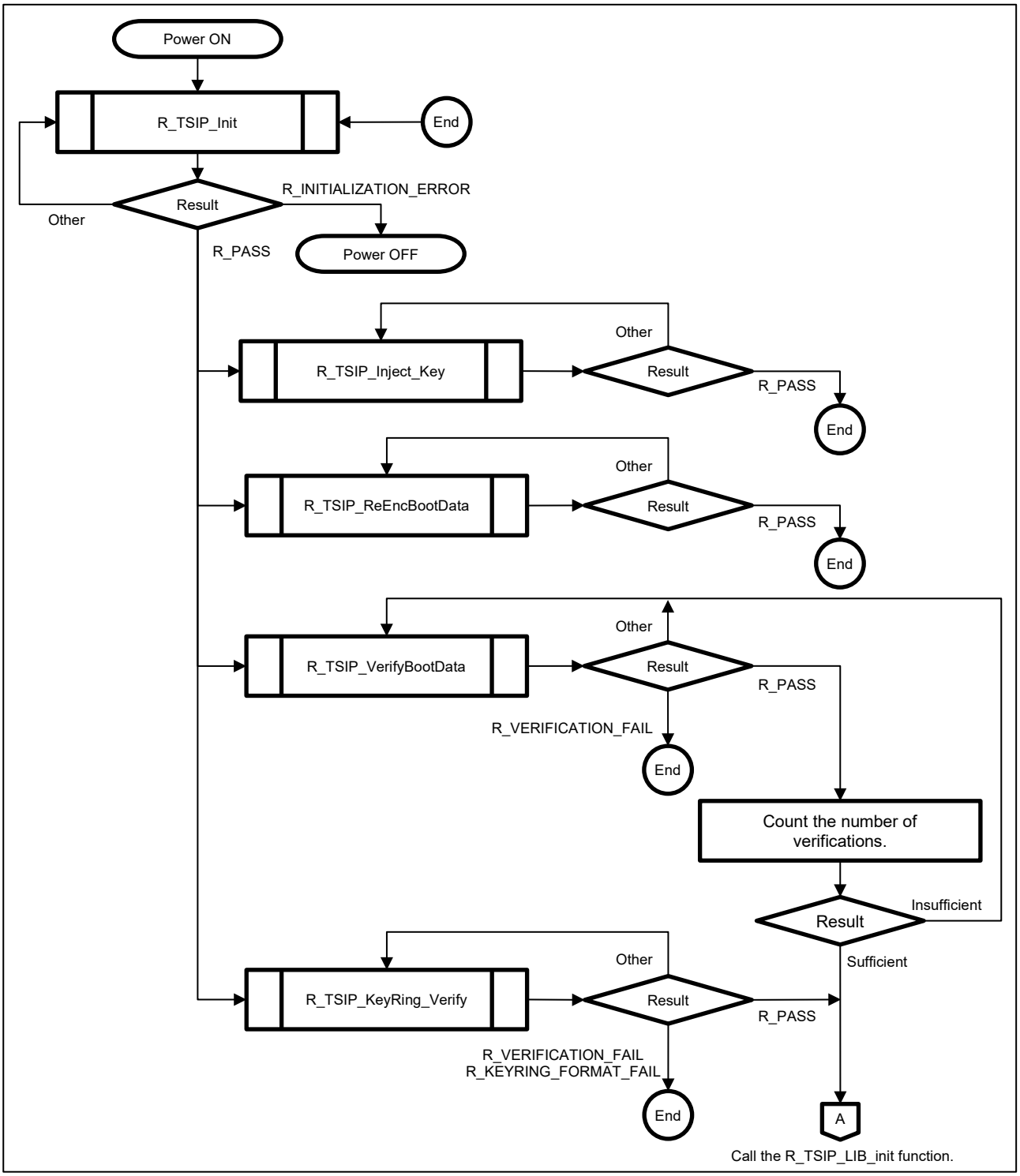

Figure 4-1. State Transitions of Secure Boot APIs

4.4.1 R_TSIP_Inject_Key

R_TSIP_Inject_Key	Secure Boot
Re-encryption of Keyring	

Header	R_TSIP_Boot_Lib.h		
Declaration	unsigned long R_TSIP_Inject_Key(unsigned char *InData_ProvisioningKeyOperationCode, unsigned char *InData_KeyRingOperationCode);		
Argument	IN	*InData_ProvisioningKeyOperationCode	Pointer to the encrypted provisioning key.
	IN	*InData_KeyRingOperationCode	Pointer to the temporarily encrypted keyring.
Return Value	R_PASS		Success.
	R_PROVISIONING KEY_FAIL		Invalid the provisioning key.
	R_VERIFICATION_FAIL		Failed to verify the keyring.
	R_PARAMETER_FAIL		Invalid argument.
	R_SEQUENCE_FAIL		Invalid driver state.
	R_RESOURCE_CONFLICT_FAIL		Resource conflict.
Description	This function decrypts the temporarily encrypted keyring and re-encrypts it by using the device-specific key within the TSIP. After executing this function, the re-encrypted keyring is stored in the area of the S_INSTData specified when the R_TSIP_Init() function is called.		

4.4.2 R_TSIP_ReEncBootData

R_TSIP_Inject_Key Secure Boot

Re-Encryption of User Data

Header	R_TSIP_Boot_Lib.h	
Declaration	unsigned long R_TSIP_ReEncBootData(TSIP_REENC_BOOT_DATA *tsip_reenc_bootdata);	
Argument	IN / OUT *tsip_reenc_bootdata	Pointer to the parameter structure that specifies user data for Secure Boot.
Return Value	R_PASS	Success.
	R_VERIFICATION_FAIL	Failed to verify the user data.
	R_PARAMETER_FAIL	Invalid argument.
	R_SEQUENCE_FAIL	Invalid driver state.
	R_RESOURCE_CONFLICT_FAIL	Resource conflict.

Description

This function decrypts the temporarily encrypted user data and re-encrypts user data with the device-specific key in TSIP. User data re-encrypted with this function can be decrypted and verified with the R_TSIP_VerifyBootData() function.

Before executing this function, the re-encrypted keyring must be stored in the area of the S_INSTData specified when the R_TSIP_Init() function is called.

The tsip_reenc_bootdata is a pointer to the parameter structure that specifies user data for Secure Boot. The parameter structure is defined as an array of structures with 16 elements (See 4.2.3(1) TSIP_REENC_BOOT_DATA). The information of temporarily encrypted user data to be input and the information of re-encrypted user data to be output are stored in this structure in order from the first element. The order of the user data stored in each element must be the same as the order in which the R_TSIP_VerifyBootData function verifies.

InData_BootData_ByteSize is the size of the data stored in the area of the InData_BootData. InData_BootData_ByteSize is a multiple of 16 with a minimum size of 272 bytes. Re-encrypted data is stored at address specified OutData_BootData. OutData_BootData of TSIP_REENC_BOOT_DATA[0] must be specified area larger than InData_BootData_ByteSize + 64 bytes. OutData_BootData of TSIP_REENC_BOOT_DATA[1 to 15] must be specified area larger than InData_BootData_ByteSize + 16 bytes.

If the number of user data is less than 16, set the remaining element members to NULL or 0.

Re-encryption is performed at once, and it is not possible to re-encrypt each data separately.

4.4.3 R_TSIP_VerifyBootData

R_TSIP_VerifyBootData	Secure Boot
Decryption and Verification of User Data	

Header	R_TSIP_Boot_Lib.h		
Declaration	unsigned long R_TSIP_VerifyBootData(unsigned char *InData_BootData, unsigned long InData_BootData_ByteSize, unsigned char *InData_BootData_HeapArea, unsigned char *OutData_BootData);		
Argument	IN	*InData_BootData	Pointer to the re-encrypted user data.
	IN	InData_BootData_ByteSize	Size of the re-encrypted user data.
	OUT	*InData_BootData_HeapArea	Pointer to heap area used by TSIP in decryption.
	OUT	*OutData_BootData	Pointer to store the decrypted user data.

Return Value	R_PASS	Success.
	R_VERIFICATION_FAIL	Failed to verify user data.
	R_PARAMETER_FAIL	Invalid argument.
	R_SEQUENCE_FAIL	Invalid driver state.
	R_RESOURCE_CONFLICT_FAIL	Resource conflict.

Description

This function decrypts and verifies the re-encrypted user data with TSIP. The re-encrypted user data is the data output by R_TSIP_ReEncBootData() function.

Before executing this function, the re-encrypted keyring must be stored in the area of the S_INSTData specified when the R_TSIP_Init() function is called. By executing this function, the re-encrypted keyring is also verified.

If multiple user data is re-encrypted, this function must be executed multiple times. Decryption and validation with this function is performed for each re-encrypted user data. The order in which the re-encrypted user data is validated must be the same as the order of the elements specified in the TSIP_REENC_BOOT_DATA array when the R_TSIP_ReEncBootData() function is called.

The area of the InData_BootData_HeapArea requires InData_BootData_ByteSize - 272 bytes. To decrypt the user data specified in TSIP_REENC_BOOT_DATA [0], the area of the OutData_BootData requires InData_BootData_ByteSize - 320 bytes. To decrypt the user data specified in TSIP_REENC_BOOT_DATA [1~15], the area of the OutData_BootData requires InData_BootData_ByteSize - 272 bytes.

After verifying all the re-encrypted user data with this function, the R_TSIP_Lib_Init() function can be called.

4.4.4 R_TSIP_KeyRing_Verify

R_TSIP_KeyRing_Verify Secure Boot

Verification of Keyring

Header	R_TSIP_Boot_Lib.h	
Declaration	unsigned long R_TSIP_KeyRing_Verify(void);	
Argument	None	
Return Value	R_PASS	Success.
	R_KEYRING_FORMAT_FAIL	Invalid keyring format.
	R_PARAMETER_FAIL	Invalid arguments.
	R_SEQUENCE_FAIL	Invalid driver state.
	R_VERIFICATION_FAIL	Failed to verify the keyring.
	R_RESOURCE_CONFLICT_FAIL	Resource conflict.
Description	This function verifies only the re-encrypted keyring. This function is used to verify only re-encrypted keyring in Secure Boot (it is means that Secure Boot does not validate user data). Before executing this function, the re-encrypted keyring must be stored in the area of the S_INSTData specified when the R_TSIP_Init() function is called. After verifying the re-encrypted keyring with this function, the R_TSIP_Lib_Init() function can be called.	

4.5 Secure Update API Details

This section describes APIs related to Secure Update.

State Transitions of Secure Update APIs is shown below.

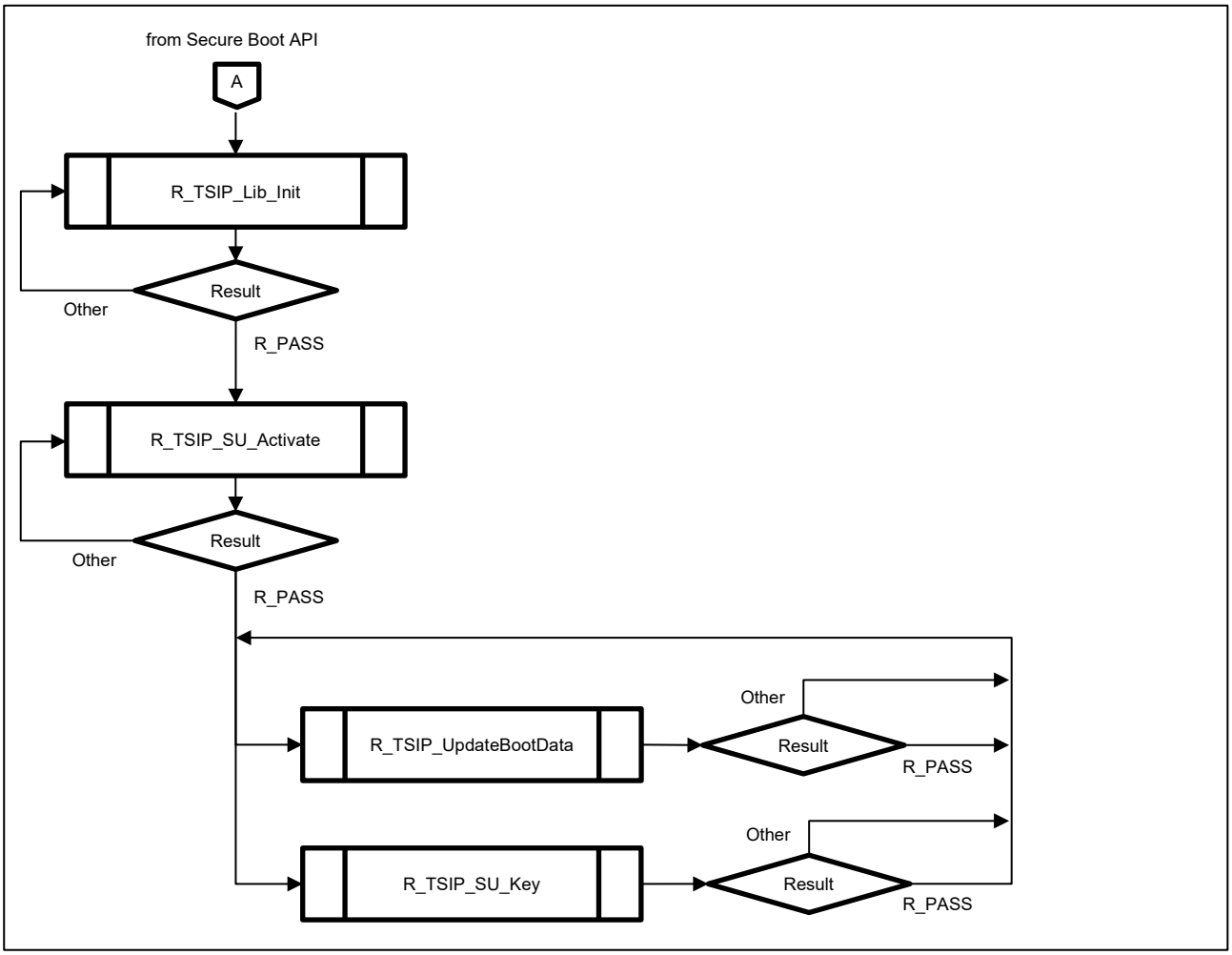

Figure 4-2. State Transitions of Secure Update APIs

4.5.1 R_TSIP_SU_Activate

R_TSIP_SU_Activate	Secure Update
Activation of Secure Update	

Header	R_TSIP_Core_Lib.h	
Declaration	unsigned long R_TSIP_SU_Activate(void);	
Argument	None	
Return Value	R_PASS	Success.
	R_SEQUENCE_FAIL	Invalid driver state.
	R_RESOURCE_CONFLICT_FAIL	Resource conflict.
	R_FALSIFICATION_ERROR	Falsification of internal data.
Description	This function prepares to start the Secure Update. After executing this function, APIs other than Secure Update cannot be executed. To run other APIs, start over with the sequence from R_TSIP_Init() function.	

4.5.2 R_TSIP_UpdateBootData

R_TSIP_UpdateBootData Secure Update

Making Update Data for Secure Boot

Header	R_TSIP_Core_Lib.h		
Declaration	unsigned long R_TSIP_UpdateBootData(TSIP_UPDATE_BOOT_DATA *tsip_update_bootdata);		
Argument	IN / OUT	TSIP_UPDATE_BOOT_DATA *tsip_update_bootdata	Pointer to the parameter structure that specifies the user data to update.
Return Value	R_PASS		Success.
	R_PARAMETER_FAIL		Invalid argument.
	R_SEQUENCE_FAIL		Invalid driver state.
	R_VERIFICATION_FAIL		Failed to verify the user data.
	R_RESOURCE_CONFLICT_FAIL		Resource conflict.
	R_FALSIFICATION_ERROR		Falsification of internal data.

Description

This function makes update the user data to be decrypted and verified at Secure Boot. The user must replace the existing re-encrypted user data with the re-encrypted user data created by this function. At the next boot, the updated user data will be verified and decrypted by Secure Boot.

Before executing this function, the re-encrypted keyring must be stored in the area of the S_INSTData specified when the R_TSIP_Lib_Init() function is called. This re-encrypted keyring contains the session key used to temporarily encrypt user data for update.

The tsip_update_bootdata is a pointer to the parameter structure that specifies the user data for update. The parameter structure is defined as an array of structures with 16 elements (See 4.2.3(2) TSIP_UPDATE_BOOT_DATA). The index of the element that stores the information of update data must be the same as the index when it was re-encrypted in provisioning.

InData_BootData_ByteSize is the size of the data stored in the area of the InData_BootData. InData_BootData_ByteSize is a multiple of 16 with a minimum size of 272 bytes. Re-encrypted data is stored at address specified OutData_BootData. OutData_BootData must be specified area larger than InData_BootData_ByteSize + 16 bytes.

Note: The re-encrypted user data output to TSIP_UPDATE_BOOT_DATA[0] requires the user to combine with 48 bytes of additional information at the end of the data. The additional information is the last 48 bytes of the re-encrypted user data output to TSIP_REENC_BOOT_DATA[0] when the R_TSIP_ReEncBootData function is called.
If the additional information is not combined, the R_TSIP_VerifyBootData function will fail to verify the data.

The users can choose to update or not update for each user data. For user data that is not updated, set the update flag in element to 0.

Secure Update cannot increase or decrease the user data verified by Secure Boot.

4.5.3 R_TSIP_SU_Key

R_TSIP_SU_Key Secure Update

Making Update Keyring for Secure Boot

Header	R_TSIP_Core_Lib.h		
Declaration	unsigned long R_TSIP_SU_Key(unsigned char *InData_KeyRingOperationCode);		
Argument	IN	InData_KeyRingOperationCode	Pointer to the temporarily encrypted keyring.
Return Value	R_PASS		Success.
	R_VERIFICATION_FAIL		Failed to verify the keyring.
	R_PARAMETER_FAIL		Invalid argument.
	R_SEQUENCE_FAIL		Invalid driver state.
	R_RESOURCE_CONFLICT_FAIL		Resource conflict.
	R_FALSIFICATION_ERROR		Falsification of internal data.
Description	This function makes update the keyring to be verified at Secure Boot. The user must replace the existing re-encrypted keyring with the re-encrypted keyring created by this function. At the next boot, the updated keyring will be verified by Secure Boot. Before executing this function, the existing re-encrypted keyring must be stored in the area of the S_INSTData specified when the R_TSIP_Lib_Init() function is called. The temporarily encrypted keyring for update is specified in InData_KeyRingOperationCode. After executing this function, the area of S_INSTData will be overwritten with the re-encrypted keyring for update. The user must store this re-encrypted keyring in non-volatile memory.		

4.6 Basic Cryptographic API Details

This section describes APIs related to Basic Cryptographic.

State Transitions of Basic Cryptographic APIs is shown below.

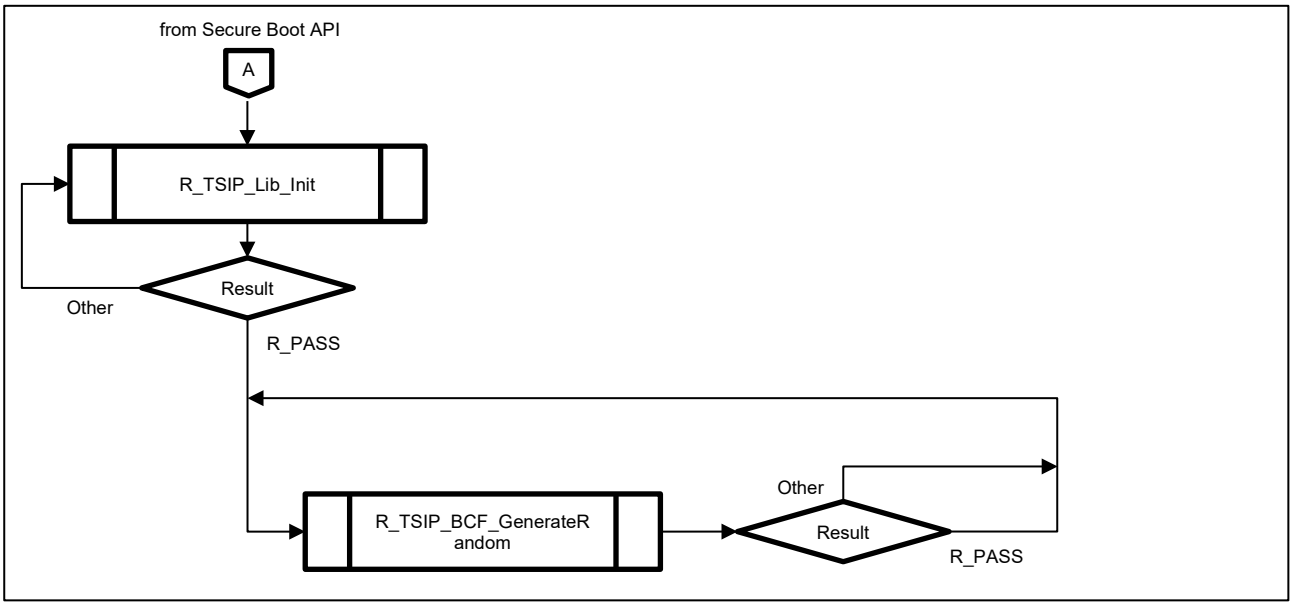

Figure 4-3. State Transitions of Basic Cryptographic APIs

4.6.1 R_TSIP_BCF_GenerateRandom

R_TSIP_BCF_GenerateRandom	Basic Cryptographic
Random Number Generation	

Header	R_TSIP_Core_Lib.h		
Declaration	unsigned long R_TSIP_BCF_GenerateRandom(unsigned long InData_GenByteSize, unsigned long InData_UseCASE, unsigned char *OutData);		
Argument	IN	InData_GenByteSize	Byte size of the random number.
	IN	InData_UseCASE	Fixed to 0.
	OUT	*OutData	Pointer to the area where random numbers are output.
Return Value	R_PASS		Success.
	R_PARAMETER_FAIL		Invalid argument.
	R_SEQUENCE_FAIL		Invalid driver state.
	R_RESOURCE_CONFLICT_FAIL		Resource conflict.
Description	This function generates random numbers.		

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	May 28, 2021	-	First 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/.