RZ/G1 Trusted Secure IP (TSIP) Software

Application Note (Function)

Introduction

This document describes the RZ/G1 Trusted Secure IP (TSIP) Software, which is provided for the Verified Linux Package of the RZ/G Linux platforms.

This document is a manual intended for users who plan to develop their own security solution software by using the Verified Linux Package of an RZ/G product.

Target Device

RZ/G1M-PF
RZ/G1E-PF
RZ/G1N-PF
RZ/G1C-PF

Contents

1. Overview 3
1.1 Key Features 3
1.2 Provided Items 5
1.3 References 7
1.4 Terms 8
2. Software Configuration 9
2.1 Software Interface 10
3. Functions 11
3.1 Secure Storage 11
3.2 Secure Software Update 11
3.3 Basic Cryptographic Function 11
3.4 Encrypted Kernel Booting 11
4. API (Security Library) 12
5. API (Security Driver for Boot/Provisioning) 13
6. Key and Keyring 14
6.1 Key Type 14
6.2 Keyring 14
7. Provisioning Process and Encrypted Kernel Booting 16
7.1 Provisioning Process 16
7.1.1 Temporary Encryption 17

7.1.2 Re-Encryption 18
7.2 Encrypted Kernel Booting 19

8. User Key Injection 20
8.1 Using user key 20

9. Secure Software Update 21
9.1 Temporary Encryption 21
9.2 Re-encryption 21

10. Product Operation and Disposal 22
10.1 Safe Operation 22
10.2 Product Disposal 22

11. Provisioning Tool 23
11.1 Functions 23
11.2 Environment for Operation 23
11.3 Configuration of Files 24
11.4 How to Use 27
11.4.1 Starting and Ending a Session with the Tool 27
11.4.2 Generating Keys 28
11.4.3 Generating the Keyring Data Required in Provisioning Process 29
11.4.4 Generating the Linux Kernel Required in Provisioning and Updating Process 30
11.4.5 Generating the Keyring Data Required in Updating 31
11.4.6 Key Data Injection 32
11.4.7 Display of an Error Message 33

Revision History 34

1. Overview

For RZ/G security, Renesas provides highly tamper-resistant security functions by using the on-chip Trusted Secure IP (hereinafter referred to as “TSIP”) included with RZ/G series products. The following security functions are provided.

- Secure Storage
- Secure Software Update
- Basic Cryptographic functions

These functions are achieved by “Security Daemon” and “Security Library” operating in user space of Linux. User application calls and uses the service provided by “Security Daemon” via “Security Library” API. “Security Daemon” includes “Security Driver” and uses TSIP to execute each function.

- Re-Encryption of Keyring and user data
- Decryption and Verification of Keyring and user data

These functions are used to decrypt and verify the Keyring and user data(assuming uImage/DeviceTree, the same applies thereafter) that have been encrypted in advance. In our solution, this process is called “Encrypted Kernel Booting”. This software package provides “Security Driver for Provisioning (Linux)” and “Security Driver for Boot (non-OS)”. Security Driver for Provisioning is the TSIP driver operated in user space of Linux for encryption in advance. Security Driver for Boot (non-OS) is the TSIP driver operated in non-OS environment for decryption when booting. For using function of TSIP, Encrypted Kernel Booting is needed. Since TSIP is not activated if decryption and verification process fail when booting, encrypted data can be protected more securely.

1.1 Key Features

In RZ/G1 Trusted Secure IP (TSIP) Software, 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. Device-specific key in TSIP cannot be accessed from outside, so it has a high tamper resistance.

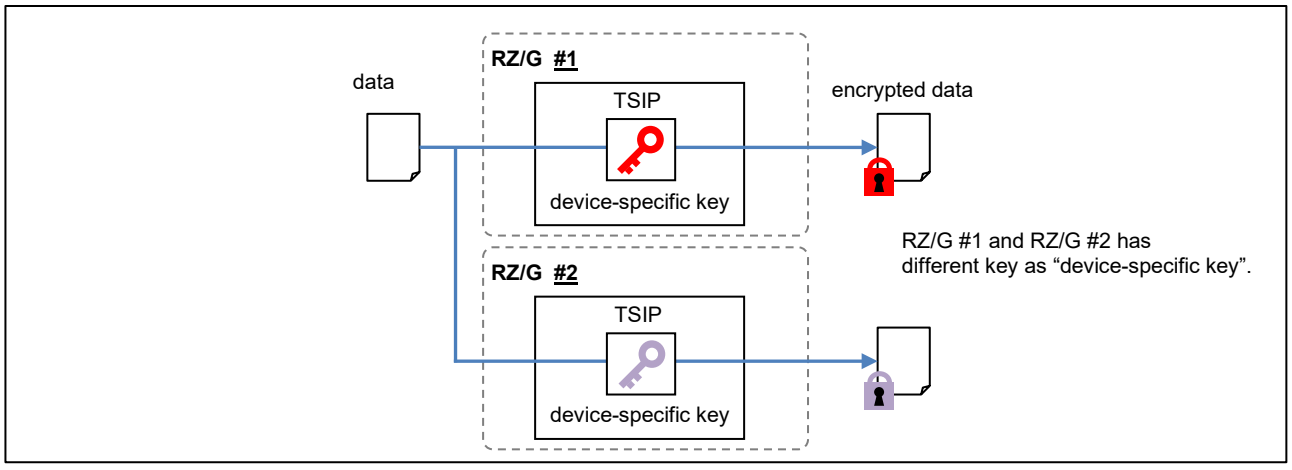


Figure 1-1 Schematic View of Encryption with Device-Specific Key

Data encrypted with the device-specific key can be decrypted only by the device that encrypted the data. Even when Linux kernel 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 Linux kernel or user data

in non-volatile memory to another product cannot be decrypted. Even Linux kernel on the same product cannot boot if it has been tampered, and tampering can be detected.

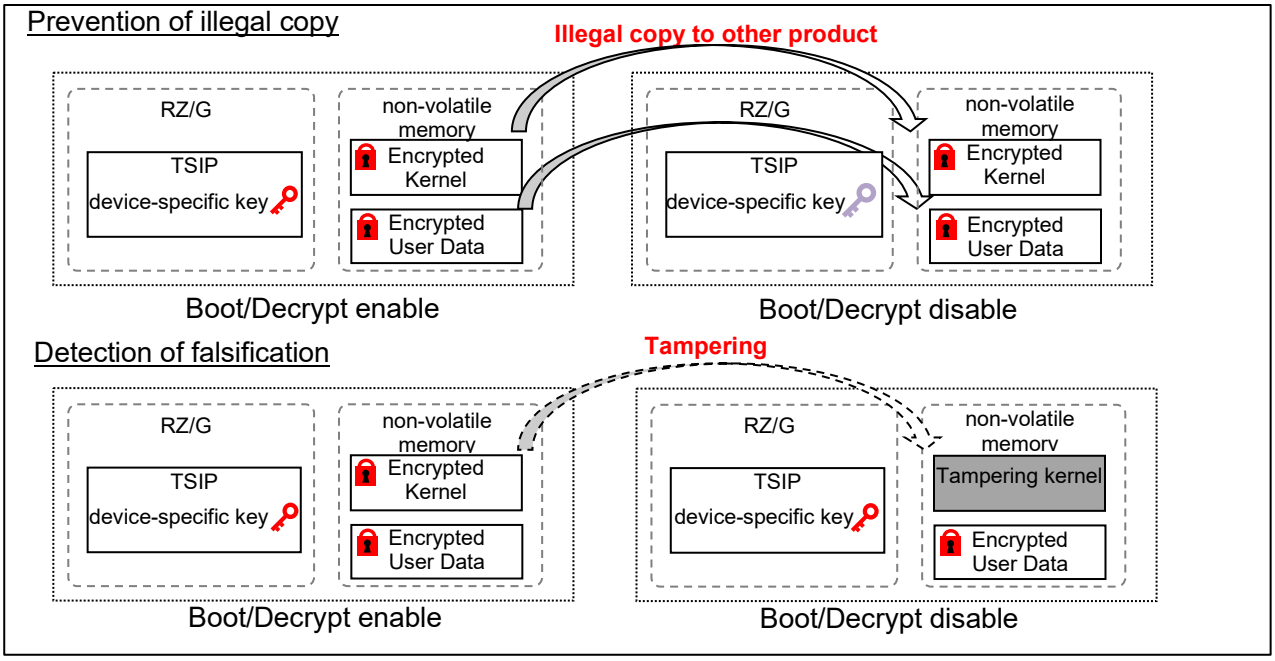


Figure 1-2 Schematic View of Encryption with Device-Specific Key

1.2 Provided Items

This package contains the following binaries.

Table 1-1 Provided Items (binaries)

type	name	description
Binary	Security Library	Application that runs in the user space of Linux. This application provides interface for user application. This application is used in conjunction with Security Daemon.
	Security Daemon	Application that runs in the user space of Linux. This application processes requests from Security Library. Security daemon includes Security Driver, and control TSIP via Security Driver.
	Security Driver for Provisioning (Linux)	TSIP Driver for provisioning in Linux environment. In provisioning process, this driver is used to encrypt keyring and user data with TSIP in advance for encrypted kernel booting. note: This driver is different from the driver included in the Security Daemon. Various functions of the Security Library/Security Daemon are not available in this driver.
	Security Driver for Boot (non-OS)	TSIP Driver for encrypted kernel booting in bare metal(non-OS) environment. In encrypted kernel booting, this driver is used to decrypt and verify keyring and user data. note: This driver is different from the driver included in the Security Daemon. Various functions of the Security Library/Security Daemon are not available in this driver.
	Security Module	Program using Security Driver for Boot(non-OS) in boot loader. This is u-boot application for decryption and verification re-encrypted key and user data that is re-encrypted with TSIP in advance in provisioning process. By execution this program from u-boot, TSIP is activated.

This package also contains the following sample program to make security environment and to check sequence of usage Security Library.

Table 1-2 Provided Items (Samples)

type	name	description
Sample Tool	Provisioning Tool	Windows application/source code to perform temporary encryption of data that to be bring to product in provisioning process. It is also used for temporary encryption of user key.
	Install Tool	Linux application/source code to perform re-encryption temporarily encrypted data with TSIP in provisioning process. This program uses Security Driver for Provisioning (Linux).
Sample Program	Injection sample	Linux sample code using Security Library. This is sample program for understanding the sequence of user key injection.
	Secure Storage Sample	Linux sample code using Security Library. This is sample program for understanding the sequence of secure storage function.
	Verification Sample	Linux sample code using Security Library. This is sample program for understanding the sequence of signature and verification with RSA.
	Encrypt/Decrypt Sample	Linux sample code using Security Library. This is sample program for understanding the sequence of encryption and decryption with AES.
	Secure Update Sample	Linux sample code using Security Library. This is sample program for understanding the sequence of Secure Update function.

File configuration of provided items shown below.

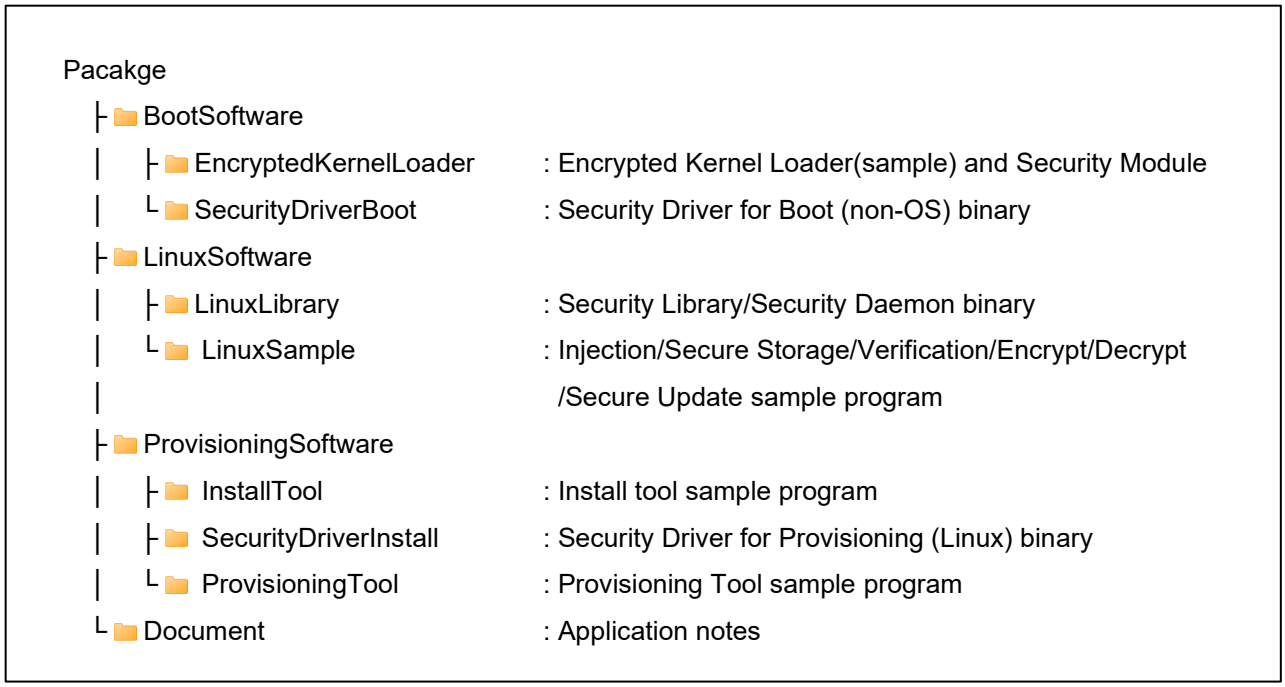


Figure 1-3 Configuration of Files for Items Provided

1.3 References

Reference documents of RZ/G1 Trusted Secure IP (TSIP) Software shown below.

- RZ/G1 Trusted Secure IP (TSIP) Software Application Note (function) (R01AN5788EJ), (This document)

This document describes functions of RZ/G1 Trusted Secure IP (TSIP) Software.

- RZ/G1 Trusted Secure IP (TSIP) Software Application Note (API) (R01AN5789EJ)

This document describes details of API of Security Library and Security Driver for RZ/G1 Trusted Secure IP (TSIP) Software. It also describes tools and samples using Security Library and Security Driver.

1.4 Terms

term	description
Trusted Secure IP (TSIP)	Security IP in RZ/G devices
Device-specific key	Unique key for the device inside TSIP. This key cannot be access from outside the TSIP.
temporary encryption	Encryption key, keyring and target data that has to be verified in booting, handled by TSIP once outside TSIP.
re-encryption	Decryption the temporarily encrypted data with TSIP and encrypt again with device-specific key with TSIP.
provisioning process	Process of temporary encryption and re-encryption keyring and user data with TSIP and store the re-encrypted data to non-volatile memory. For using TSIP, provisioning process must be performed in advance.
encrypted kernel booting	Process of decryption and verification the keyring and user data that has been re-encrypted in advance and make TSIP ready for use.
injection	Re-encryption of key and keyring.

2. Software Configuration

The following table shows the software configuration of the RZ/G security functions.

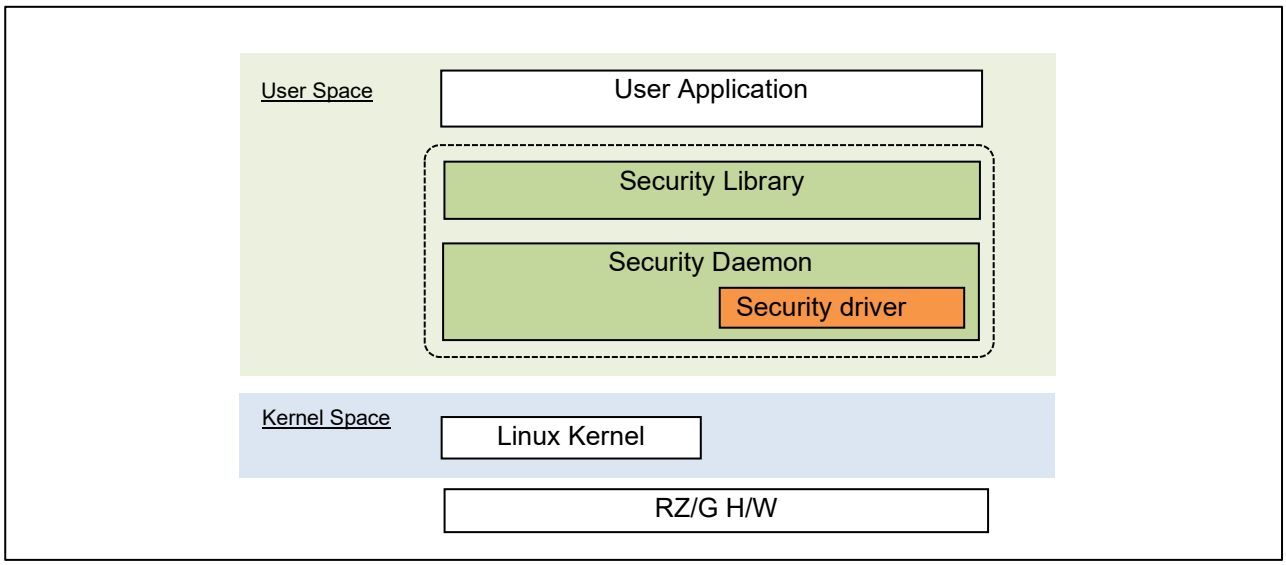


Figure 2-1 Software Configuration

Table 2-1 Components of the Software

name	description
Security Library	Provides the application program interface (API).
Security Daemon	Handles the execution of requests from applications.
Security Driver	Provides the driver for RZ/G security.

2.1 Software Interface

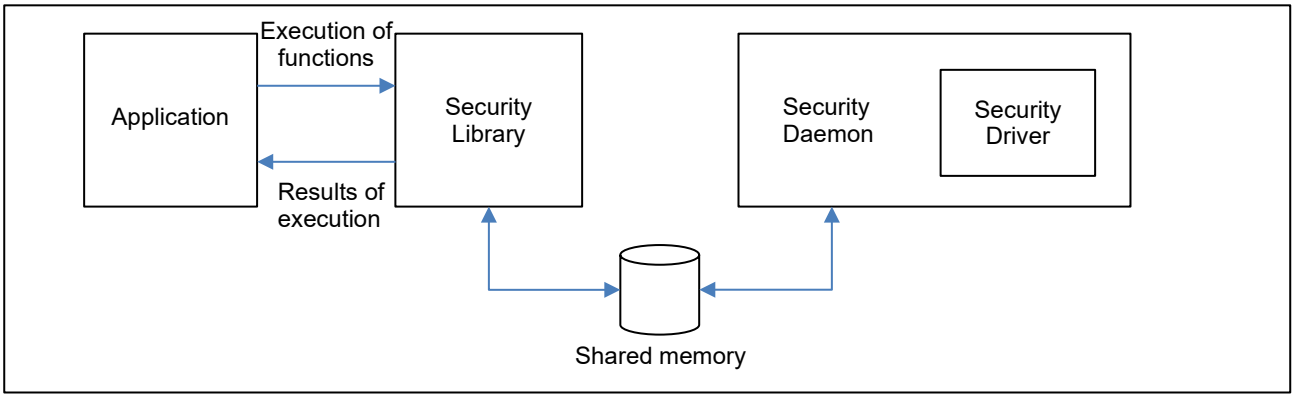


Figure 2-2 Schematic view of Interface between the software

Applications execute API functions through Security Library and obtain the results of execution.

Security Library selects either synchronous or asynchronous mode as a method of notification of the results of execution.

In synchronous mode, the results of execution are returned as return values of the API functions.

In asynchronous mode, a thread is generated in the Security Library and processing is executed in this thread. The API functions end successfully without waiting for processing to finish. After the completion of processing in a thread, the callback function specified by a given argument is executed, and the result of execution is returned to the application.

Table 2-2 Methods of Notification of the Results of Execution

mode	method of notification of the results of execution
Synchronous mode	Return values of API functions
Asynchronous mode	Callback functions

Shared memory are used in communications between the Security Library and Security Daemon. There are two types of shared memory, the shared memory created by Security Library and by the Security Daemon. Security Library creates the required shared memory each time an API function is executed and discards it when processing is complete. Security Daemon creates the required shared memory at startup.

3. Functions

3.1 Secure Storage

RZ/G Security Solution provides a secure storage function in the form of an API for using specific keys generated in the TSIP to protect data. The individual keys are generated for each TSIP, so only the specific RZ/G series product in which the protection was applied can release the data from protection by using its specific key.

Table 3-1 Provided functions for Secure Storage

function	description
Encryption	Encrypt user data using device-specific key in TSIP
Decryption	Decrypt user data using device-specific key in TSIP

3.2 Secure Software Update

RZ/G Security Solution provides function to make update data of keyring and user data that is decrypted and verified when booting.

Table 3-2 Provided functions for Secure Software Update

function	description
Update Keyring	Make data for update the keyring
Update User data	Make data for update the target of decryption and verification when booting

3.3 Basic Cryptographic Function

Renesas provides the cryptographic functionality of the TSIP shown in the table below in the form of a basic cryptographic API function.

Table 3-3 Provided Algorithms

type	algorithm
Symmetric cryptography	AES in CBC mode (128 or 256 bits)
Asymmetric cryptography	RSA (1024 or 2048 bits)
Hashing algorithm	SHA-1 or SHA-256
MAC	HMAC (SHA-1 or SHA-256) CMAC (AES-128 or AES-256)

3.4 Encrypted Kernel Booting

In Encrypted Kernel Booting, decrypt and verify the data which is provided in advance in provisioning process. RZ/G Security Solution provide TSIP driver for provisioning and encrypted kernel booting.

Table 3-4 Provided functions of TSIP driver for provisioning/encrypted kernel booting

function	description
re-encryption of keyring	Re-encrypt the temporarily encrypted keyring using device-specific key in TSIP.
re-encryption of user data	Re-encrypt the temporarily encrypted user data using 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.

4. API (Security Library)

A list of APIs provided by Security Library to user application is shown below.

Table 4-1 List of APIs (Security Library)

No	Function Name	Description
1	SEC_Initialize	Initialize Security Library and Security Daemon.
2	SEC_Finalize	Finalize Security Library and Security Daemon.
3	SEC_ContentsKeyConvert	Re-Encrypt user key with TSIP.
4	SEC_ContentsKeyGenerate	Create Key for encryption/decryption/signing/verification
5	SEC_ContentsDataEnc	Encrypt data with re-encrypted key
6	SEC_ContentsDataDec	Decrypt data with re-encrypted key
7	SEC_ContentsDataMakeVerifyCode	Make verification code with re-encrypted key
8	SEC_ContentsDataVerify	Verify data with re-encrypted key and verification code
9	SEC_SecureStorageEnc	Encrypt data using device-specific key
10	SEC_SecureStorageDec	Decrypt data using device-specific key
11	SEC_Kernel_Update	Make update uImage data which is decrypted/verified when booting
12	SEC_DevTree_Update	Make update DevTree data which is decrypted/verified when booting
13	SEC_Key_Update	Make update keyring
14	SEC_GetPackageVersion	Get version

About details of API, please refer to “RZ/G1 Trusted Secure IP (TSIP) Software Application Note (API)”.

5. API (Security Driver for Boot/Provisioning)

A list of APIs provided by Security Driver for Boot/Provisioning that is used in Encrypted Kernel Booting is shown below. Security Driver for Boot and Security Driver for Provisioning provide the same functions.

Table 5-1 List of APIs (Security Driver for Boot/Provisioning)

No	Function Name	Description
1	R_TSIP_Init	Initialize TSIP
2	R_TSIP_Inject_Key	Re-encrypt temporarily encrypted keyring using TSIP
3	R_TSIP_Install_uImage	Re-encrypt temporarily encrypted uImage data using TSIP
4	R_TSIP_Install_DeviceTree	Re-encrypt temporarily encrypted DeviceTree data using TSIP
5	R_TSIP_SB_uImage	Decrypt and verify the re-encrypted uImage data using TSIP
6	R_TSIP_SB_DeviceTree	Decrypt and verify the re-encrypted DeviceTree data using TSIP

About details of API, please refer to “RZ/G1 Trusted Secure IP (TSIP) Software Application Note (API)”.

6. Key and Keyring

6.1 Key Type

In this software package, the following keys are used.

Table 6-1 Key type

name	description
Hidden Root Key	Key managed within Renesas, and this key is not provided to user.
Provisioning Key	Keys shared by the user and Renesas This key is used for temporary encryption and message authentication of keyring. Provisioning Key is binary concatenation of two keys. (Provisioning Key = temporary encryption key for keyring || MAC key for keyring) *
EncProvisioning Key	Encrypted Provisioning Key with Hidden Root Key. TSIP uses this key for decrypt keyring that is temporarily encrypted with Provisioning Key.
Keyring	A bunch of keys used by the user in the product. For the formats in the keyring, see Table 6-3.
User Key	User key used for each cryptographic process.

note: “ || ” represents binary concatenation.

6.2 Keyring

TSIP in RZ/G uses multiple keys for each use. User need to create each user key. In this software package, this user keys for each use is managed collectively as keyring. Keyring is a file combination of each key and IV (Initialization Vector) in binary.

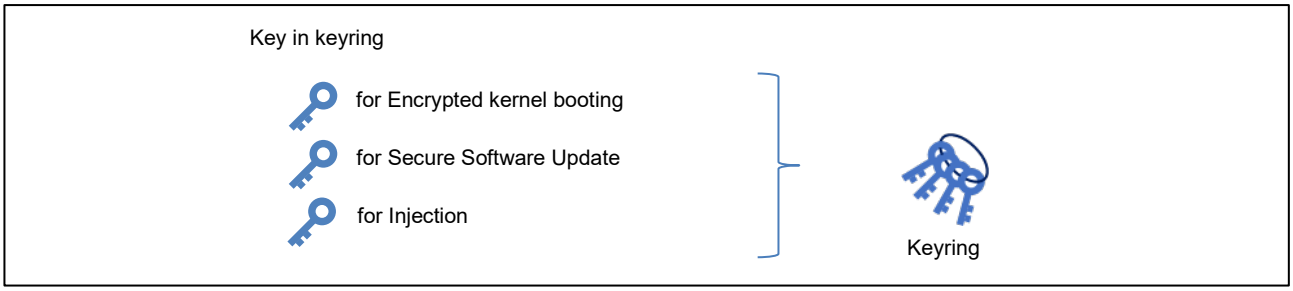


Figure 6-1 Schematic View of Keyring

Table 6-2 Kind of Keys

Key included in keyring	Description
for Encrypted Kernel Booting	Used for encryption and verification signature of user data to be decrypted and verified in Encrypted Kernel Booting.
for Security Software Update	Used to encrypt keyring, calculate MAC value, and verify MAC value for Secure Software Update
for Injection	Used to encrypt key, calculate MAC value, and verify MAC value for key injection

Formats in the Keyring is shown below. User need to prepare unique keys as each data included in the keyring. Provisioning Tool provides function of create these key and keyring.

Table 6-3 Formats in the Keyring

Key		**Algorithm**	**Size (Byte)**
for Encrypted Kernel Booting	Temporary encryption key for uImage	AES-128	16
		IV	16
for Encrypted Kernel Booting	Temporary encryption key for DeviceTree	AES-128	16
		IV	16
	Temporary verification key for signature of uImage/DeviceTree*	PublicKey (n)	256
		PublicKey (0^15 Padding || e || 0^96 Padding)	16
Reserved		(fixed 0)	272
for Secure Software Update	Temporary encryption key for MAC key	AES-128	16
	MAC key for keyring	AES-128	16
for Injection	Temporary encryption key for user key	AES-128	16
	MAC key for user key	AES-128	16

note * : Private key that is pair of this key (temporary signing key for signature of user data) is used for signing user data in provisioning process.

7. Provisioning Process and Encrypted Kernel Booting

To use various encryption/decryption functions of TSIP, it is necessary to execute Encrypted Kernel Booting. To perform Encrypted Kernel Booting, provisioning process must be performed in advance. Provisioning process is the process of re-encrypt keyring and user data with TSIP and storing them in non-volatile memory. Encrypted Kernel Booting is the process of decryption and verification the keyring and user data that has been re-encrypted in advance and make TSIP ready for use.

7.1 Provisioning Process

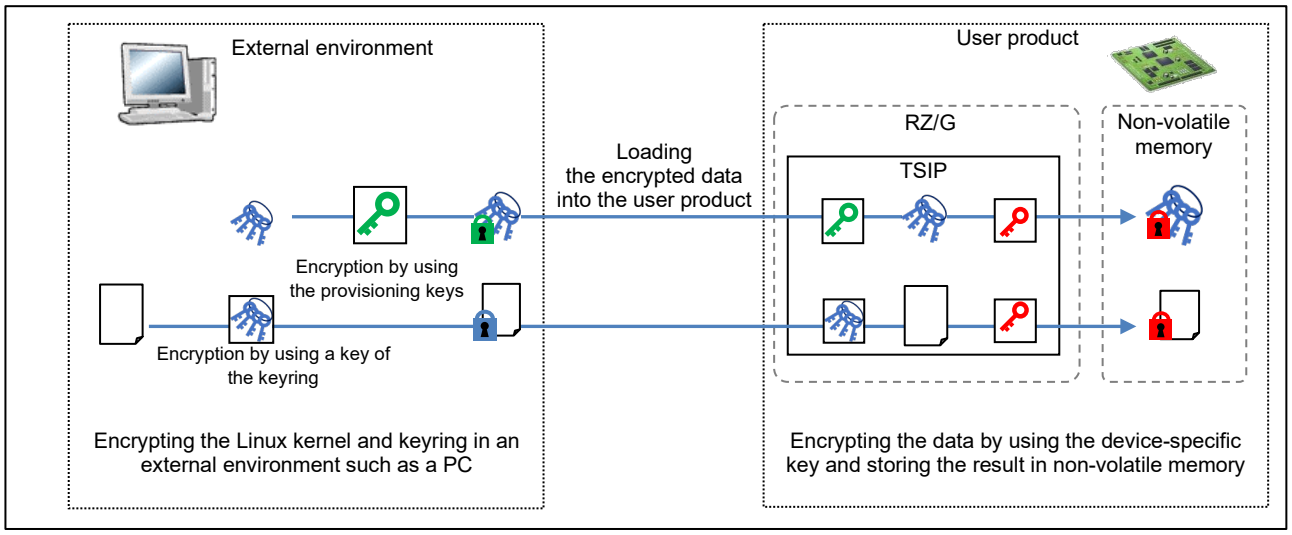


Figure 7-1 Schematic View of Provisioning Process

Proceed with the provisioning process by following the procedure below.

1. Encrypt the user data to be decrypted/verified and keyring in an external environment (temporary encryption) and loading the encrypted data into the user product.
2. Using the TSIP in an RZ/G device in the user product to proceed with decryption of the encrypted data and re-encryption of the result by using a device-specific key. After that, store the result in the non-volatile memory.

When bringing key data and user data to be decrypted/verified at booting into the product, the data is temporarily encrypted, the temporarily encrypted data is decrypted inside TSIP, and the re-encrypted data is output outside TSIP. This makes it possible to prevent information leakage even when data is brought into the product.

7.1.1 Temporary Encryption

This section describes keys used in provisioning process and temporary encryption keyring and user data in provisioning process. Follow the steps below to create each data to be brought into the product.

1. Encrypt Provisioning Key with Hidden Root Key. This process performed by Renesas, and Renesas provides Encrypted Provisioning Key (EncProvisioingKey) to user. About providing of EncProvisioningKey, please refer to “RZ/G1 Trusted Secure IP (TSIP) Software Application Note (API)”.

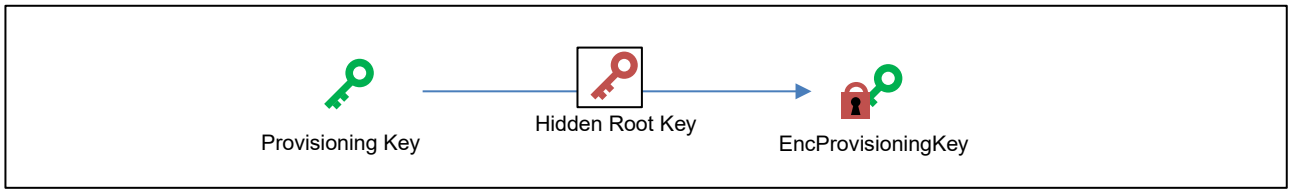


Figure 7-2 Schematic View of Encryption of Provisioning Key

2. Prepare the user keys. Generate the keyring from the prepared user keys and encrypt it. Users can use the provisioning tool for this processing. The provisioning keys (= Provisioning Key1 || Provisioning Key2 = Temporary encryption key for keyring || MAC key for keyring) are used to encrypt the keyring

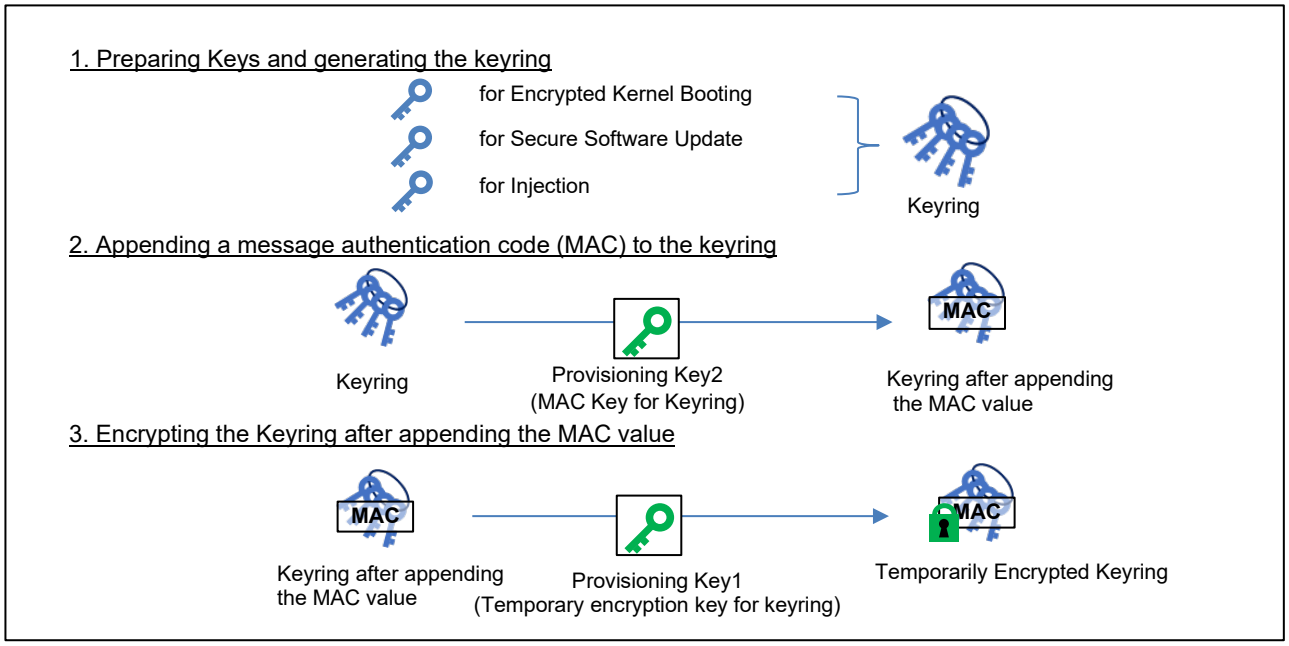


Figure 7-3 Schematic View of Temporary Encryption of Keyring

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

Table 7-1 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*

Note: IV0 = 0x85c1673483d5d291f0d0713e3ea434a3

3. Sign and encrypt user data (Linux Kernel etc.) with the key for encrypted kernel booting. Users can use the provisioning tool for this processing.

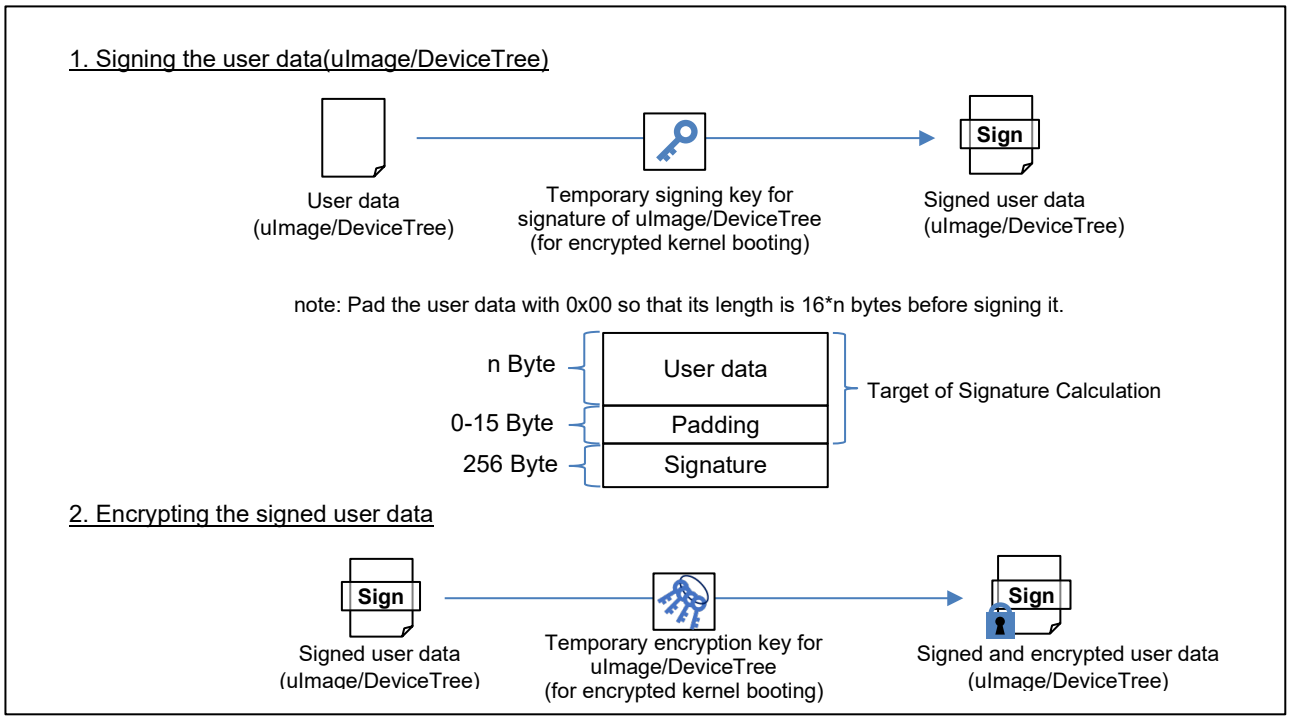


Figure 7-4 Schematic View of Temporary Encryption of user data

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

Table 7-2 Algorithm and key in temporary encryption of user data

process	algorithm	key	IV
signing	RSASSA-PKCS1-v1_5 SHA256	Temporary signing key for signature of uImage/DeviceTree	0
encryption	AES-128-CBC	Temporary encryption key for uImage/DeviceTree	IV*

Note: IV for temporary encryption key for user data (for Encrypted Kernel Booting) in keyring

This encryption/signing is performed for each uImage and Device data. This completes the creation of data to be brought into the product in advance.

7.1.2 Re-Encryption

Bring temporarily encrypted keyring and user data to product, then re-encrypt using TSIP. Since each data is brought in encrypted (temporarily encrypted), safe operation is possible. To re-encrypt each data with TSIP, Security Driver for Provisioning (Linux) is used. Re-encrypted data is stored to non-volatile memory, and these data is decrypted and verified using Security Driver for Boot (non-OS) in encrypted kernel booting.

7.2 Encrypted Kernel Booting

Encrypted Kernel Booting is the process that decrypts and verifies the keyring and user data brought into the product in the provisioning process, and enable the various encryption/decryption function of TSIP.

If decryption/verification process fail, TSIP is not activated and various encryption/decryption function of TSIP is not available.

In encrypted kernel booting, Security Driver for Boot (non-OS) is used for decryption/verification keyring and user data. For details of Security Driver for Boot (non-OS), please refer to “RZ/G1 Trusted Secure IP (TSIP) Software Application Note (API)”. This package provides sample encrypted kernel loader, which is an u-boot application using Security Module to decrypt and verify kernel data.

8. User Key Injection

If the encrypted kernel booting is completed successfully, various encryption/decryption functions of TSIP are available. When using user keys with TSIP, it is necessary to inject the user key. User key injection is the process of re-encryption temporarily encrypted user key with a device-specific key. When using various encryption/decryption function of TSIP with user key, user key must be injected.

Since the user key is brought into the product after being temporarily encrypted, the key can be operated safely. After injection, since user key is operated as re-encrypted key with device-specific key in TSIP, user key will not leak.

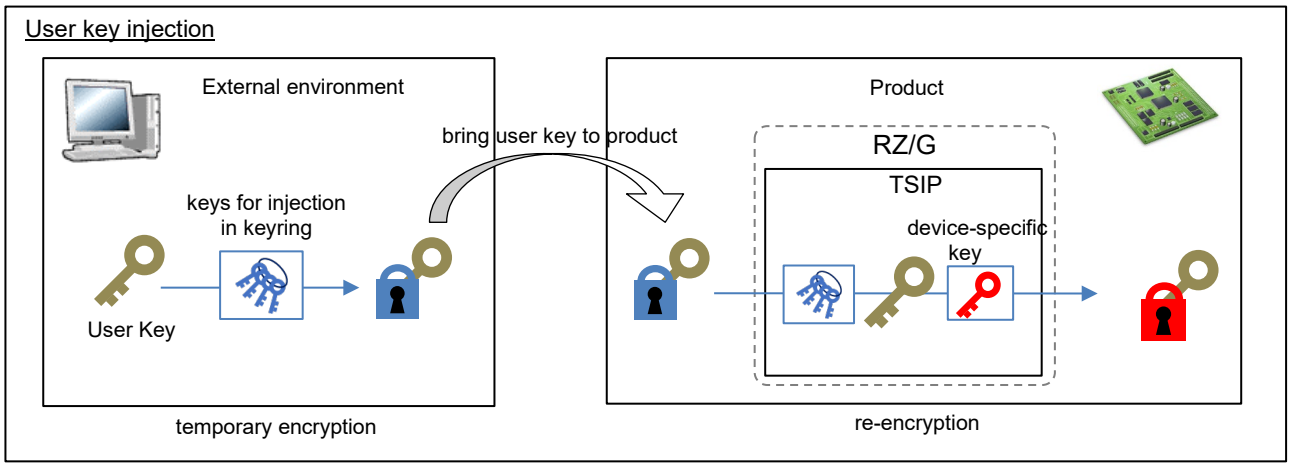


Figure 8-1 Schematic View of User Key Injection

In temporary encryption process for user key injection, temporary encryption key for user key (for Injection) and MAC key for user key (for Injection) are used. MAC value is calculated with MAC key for user key, then encryption is performed with temporary encryption key for user key.

Table 8-1 Algorithm and key in temporary encryption of user key for injection

process	algorithm	key	IV
MAC	CBC-MAC with AES-128	MAC key for user key	0
encryption	AES-128-CBC	Temporary encryption key for user key	IV*

note: When using Provisioning Tool, IV is created automatically by Provisioning Tool.

8.1 Using user key

Injected user key (that is re-encrypted user key) is used for encryption/decryption with TSIP. TSIP cannot encrypt/decrypt data using non-injected key.

Security Library and Security Damon enable to use various encryption/decryption function with TSIP using injected key.

9. Secure Software Update

After starting the operation of the product, to update the keyring and user data (uImage/DeviceTree) re-encrypted in the provisioning process, it is possible to create data for updating each. The 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 the 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.

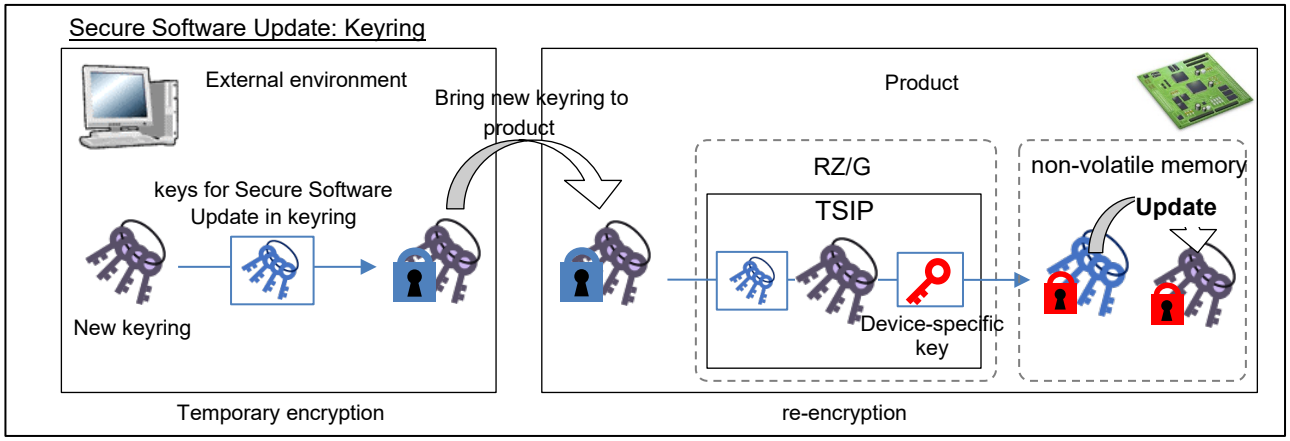


Figure 9-1 Schematic View of Secure Software Update (Keyring)

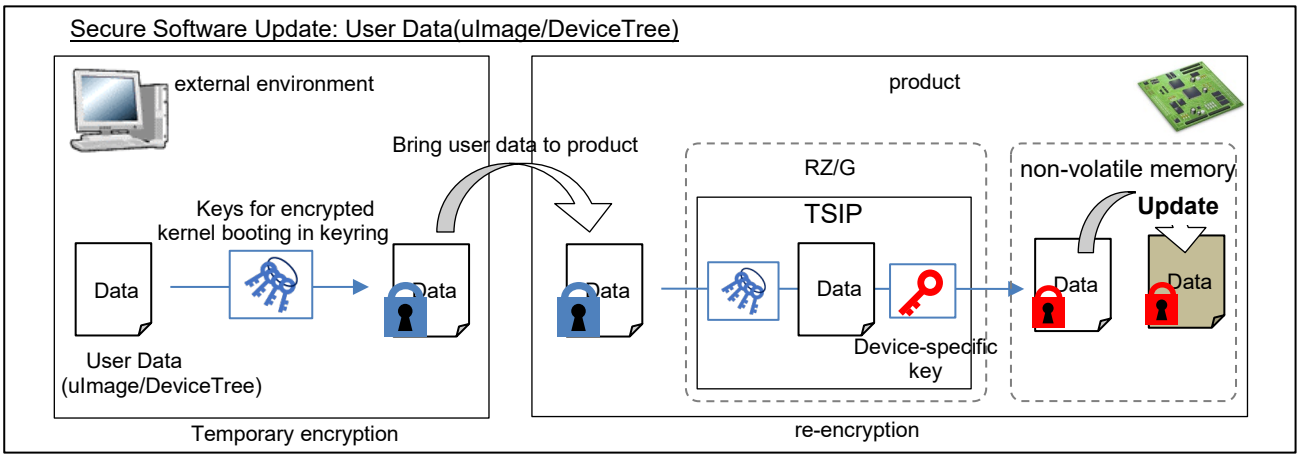


Figure 9-2 Schematic View of Secure Software Update (uImage/DeviceTree)

9.1 Temporary Encryption

For temporary encryption of the new keyring, use the keys for secure software update in the already provisioned keyring. Temporarily encrypted keyring is created in the same way as MAC addition and encryption of keyring in the provisioning process.

For temporary encryption of the new user data, use the keys for encrypted kernel booting in the already provisioned keyring. Create temporarily encrypted user data in the same way as adding signature and encryption in provisioning process. Note that you must use keys for encrypted kernel booting in the “new keyring” if new keyring re-encryption is performed already.

9.2 Re-encryption

Temporarily encrypted new keyring and user data are brought into the product and then re-encrypted in TSIP. Re-encryption process performed using Security Library and Security Daemon.

10. Product Operation and Disposal

10.1 Safe Operation

In RZ/G1 Trusted Secure IP (TSIP) Software, key data as plane text is handled only in TSIP. When key data is set to TSIP, key data must be encrypted (temporarily encrypted or re-encrypted). Therefore, it is possible to minimize the risk of user key leakage during product operation.

But in provisioning process, temporary encryption is performed with the key as plane text. User data/user key/provisioning key are also plane data. Temporary encryption in provisioning process must be performed in a secure environment.

Temporarily encrypted data is created in secure environment and provided to mass production bases/factories as ROM data containing them, then re-encrypted at user mass production bases/factories. Re-encrypted data with device-specific key can be created while minimizing the scope of the secure environment required.

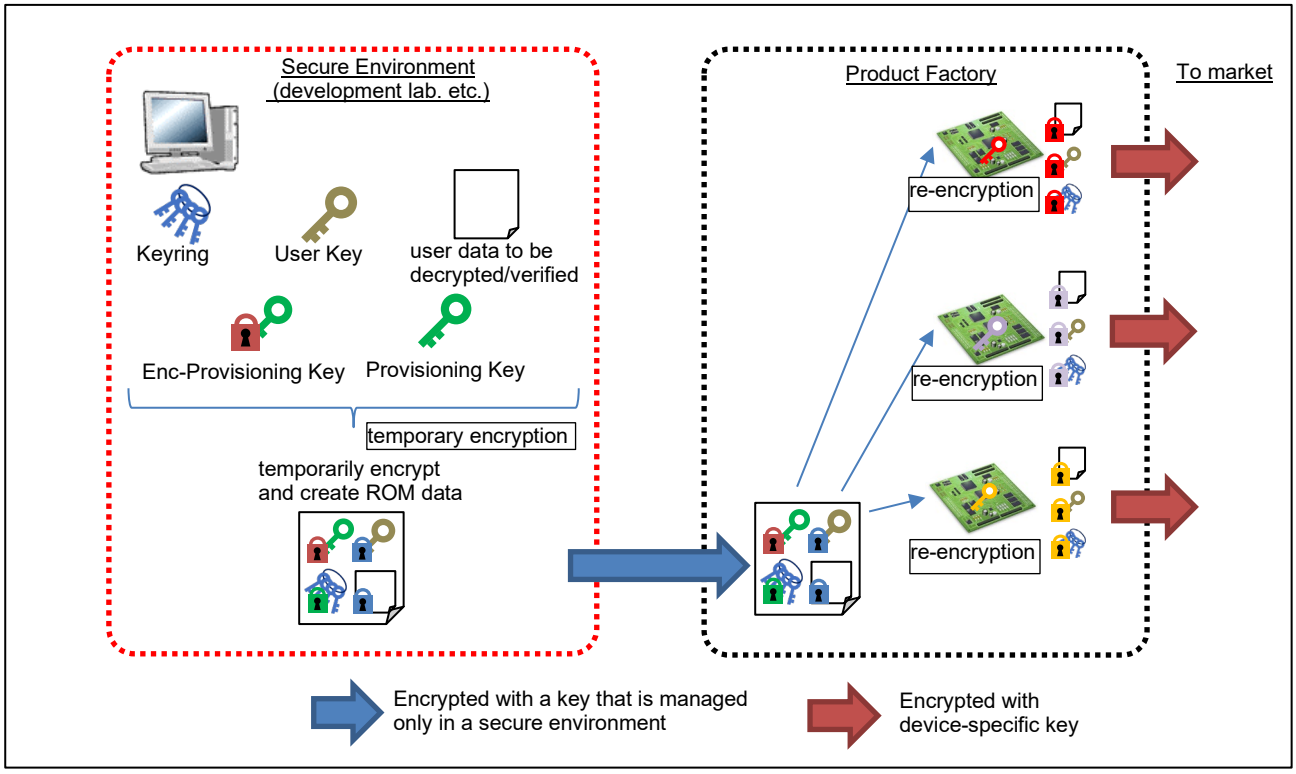


Figure 10-1 Schematic View of Shipment to market in safely

10.2 Product Disposal

Depend on the intended use of the product, it may be necessary to prevent leakage of important user data when disposing of the product. In RZ/G1 Trusted Secure IP (TSIP) Software, it is enabled to safe disposal by delete re-encrypted keyring in provisioning process. If re-encrypted keyring is deleted, TSIP is not activated and encrypted data cannot be decrypted. Make the keyring unusable by clearing the area of non-volatile memory that stores re-encrypted keyring in provisioning process to 0 or overwriting it with a fixed value. Since product cannot success encrypted kernel booting and TSIP is not activated, encrypted data is never decrypted.

11. Provisioning Tool

Provisioning Tool is Windows Application to provide functions of temporary encryption various data for provisioning process in RZ/G1 Trusted Secure IP (TSIP) Software. This tool also provides creation function of various key data by random number. This software package includes source code of Provisioning Tool.

11.1 Functions

This tool provides following functions.

- Making keyring and temporary encryption of keyring.
- Temporary encryption of user data(uImage/DeviceTree) to be decrypted and verified in Encrypted Kernel Booting.
- Making update keyring and temporary encryption of update keyring.
- Temporary encryption of update user data(uImage/DeviceTree) to be decrypted and verified in Encrypted Kernel Booting.
- Temporary encryption of user key.

11.2 Environment for Operation

This tool has been run in the following operating environment.

Table 11-1 Environment for Running Provisioning Tool

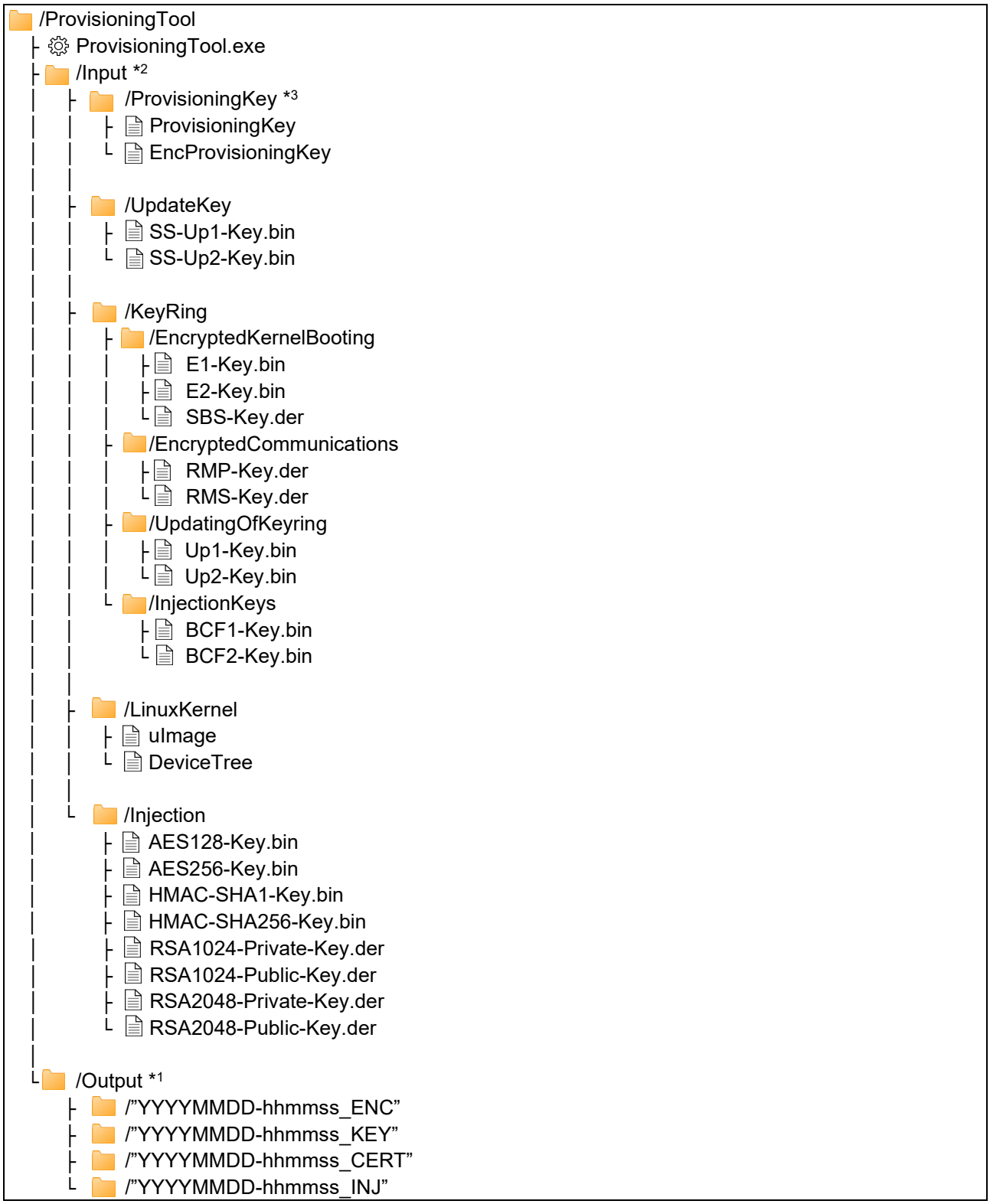
Environment	Description
OS	Windows 10 (64bit)
Using Library	Microsoft .net Framework 3.5

The GetBytes method of the RNGCryptoServiceProvider class of the .NET Framework is used as the random number generator for generating the keys.

11.3 Configuration of Files

The configuration of files for this tool is shown below.

Figure 11-1 Configuration of Files for Provisioning Tool



Note*1: The folders under the [Output] node are automatically generated after the respective tools are executed. “_KEY” is added to the end of the name of the folder to which the results of executing the tool for generating keys are output. Similarly, “_ENC” is added for the output of the tool for generating data for use in booting of the encrypted kernel, “_CERT” is added for the output of the tool for signing and “_INJ” is added for the output of the tool for injection.

Note*2: In the [Input] folder, sample Keys are arranged for evaluation in advance. For provisioning key (C1-Key.bin, C2-Key.bin, EncProvisioningKey), please put keys provided as other package by Renesas in [Input] folder.

Note*3: In the [ProvisioningKey] folder, please put the keys as Provisioning Key. This package includes sample keys(ProvisioningKey and EncProvisioningKey) for evaluation.

The names of folders and files in the tool are fixed. When using keys or data other than those generated by this tool as input data, rename them before putting them in place.

Details of file in Input folder is shown below.

Table 11-2 Key Files in Input folder(1/2)

Folder/File	Description	Type
ProvisioningKey	Provisioning Key	
C1-Key.bin	Temporary encryption key for keyring	AES-128
C2-Key.bin	MAC key for keyring	AES-128
ProvisioningKey	(Temporary encryption key of keyring || MAC key for keyring) If “ProvisioningKey” is put in this folder, C1-Key.bin and C2-Key.bin is created from “ProvisioningKey” within Provisioning Tool.	
EncProvisioningKey	Encrypted ProvisioningKey with Hidden Root Key	
UpdateKey	Key used to update keyring. User needs to rename and place the Secure Software Update key of the keyring that is already performed injection in this folder.	
SS-Up1-Key.bin	Temporary encryption key for keyring	AES-128
SS-Up2-Key.bin	MAC key for keyring	AES-128
KeyRing	Keys included in keyring	
EncryptedKernelBooting	for Encrypted Kernel Booting	
E1-Key.bin	Temporary encryption key for uImage	AES-128 || IV
E2-Key.bin	Temporary encryption key for DeviceTree	AES-128 || IV
SBS-Key.der	Temporary signature verification key for uImage/DeviceTree	RSA 204 PublicKey*
EncryptedCommunications	Reserved	
RMP-Key.der	Reserved	-
RMS-Key.der	Reserved	-
UpdatingOfKeyring	for Secure Software Update	
Up1-Key.bin	Temporary encryption key for keyring	AES-128
Up2-Key.bin	MAC key for keyring	AES-128
InjectionKeys	for Injection	
BCF1-Key.bin	Temporary encryption key for user key	AES-128
BCF2-Key.bin	MAC key for user key	AES-128

note *: der format

Table 11-3 Key Files in Input folder(2/2)

Folder/File		Description	Kind
Injection		Target Key of user key injection	
	AES128-Key.bin	User Key	AES-128
	AES256-Key.bin	User Key	AES-256
	HMAC-SHA1-Key.bin	User Key	HMAC-SHA1
	HMAC-SHA256-Key.bin	User Key	HMAC-SHA256
	RSA1024-Private-Key.der	User Key	RSA1024 PrivateKey-CRT-*
	RSA1024-Public-Key.der	User Key	RSA1024 PublicKey*
	RSA2048-Private-Key.der	User Key	RSA2048 PrivateKey-CRT-*
	RSA2048-Public-Key.der	User Key	RSA2048 PublicKey *
LinuxKernel		Kernel for Encrypted Kernel Booting	
	uImage	Linux Kernel (uImage)	-
	DeviceTree	Linux Kernel (DeviceTree)	-

note *: der format

11.4 How to Use

11.4.1 Starting and Ending a Session with the Tool

Double-clicking on ProvisioningTool.exe for execution produces the following tool window.

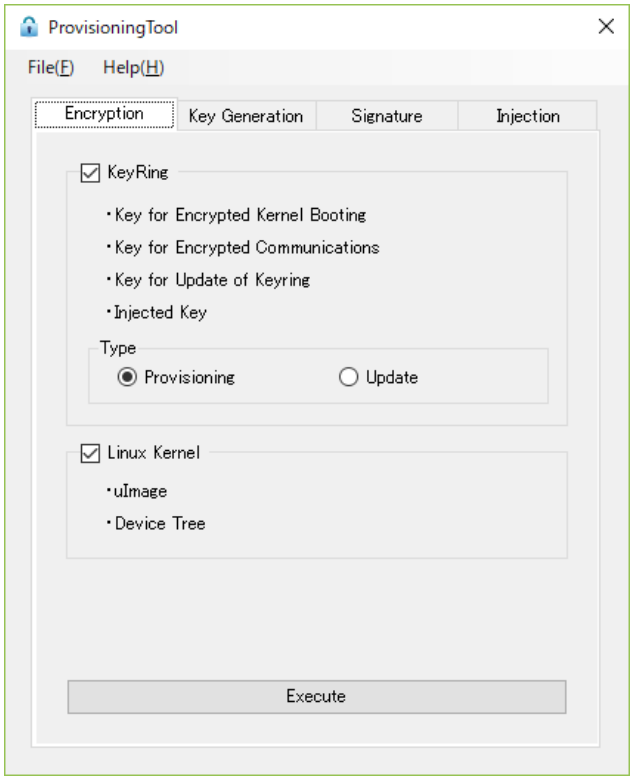


Figure 11-2 Initial Window

On completion of execution, click on “×” in the upper right corner of the tool window or select [Exit] from the [File] menu.

11.4.2 Generating Keys

Each key that can be used as a key in the Input folder creation is performed.

The steps involved in generating the keys required in booting of the encrypted kernel are listed below.

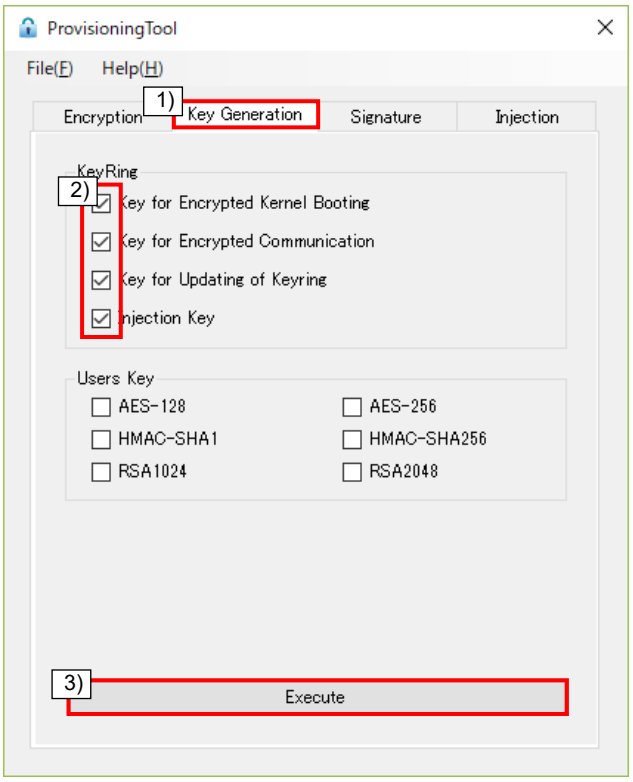


Figure 11-3 Tool Window for Generating Keys

1. Select the [Key Generation] tab.
2. Select the checkboxes for all keys to be generated.
   For booting of the encrypted kernel, select the checkbox [Keys for Encrypted Kernel Booting].
3. Click on the [Execute] button.

If you select the checkboxes [Keys for Encrypted Communication], [Keys for Updating of Keyring], [Injection Keys], then keys for encrypted communication, secure software update, and basic encryption will be created at the same time. If you selected checkboxes of [Users Key] section, keys of each cryptographic algorithm will be created at the same time. Please put key files created here in Input folder. (see section 11.3 Configuration of Files)

11.4.3 Generating the Keyring Data Required in Provisioning Process

Keyring creation from keys in KeyRing folder is performed and keyring is encrypted for use in provisioning process. keys in ProvisioningKey folder and KeyRing folder are used in this procedure. Provisioning Tool create C1-Key.bin and C2-Key.bin from ProvisioningKey file (ProvisioningKey = C1-Key.bin || C2-Key.bin). Each key in the KeyRing folder is binary concatenated according to format of keyring. Provisioning Tool calculates MAC of this keyring with C2-Key.bin and MAC is added keyring data. Then tool encrypts MAC added keyring with C1-Key.bin.

Procedure	Algorithm	Key	IV
MAC	CBC-MAC with AES-128	MAC key for keyring (C2-Key.bin)	0
Encryption	AES-128-CBC	Temporary encryption key for keyring (C1-Key.bin)	IV0*

note*: IV0=0x85c1673483d5d291f0d0713e3ea434a3

Public key created from SBS-Key.der is included in keyring, and RMS-Key.der is not included in keyring.

The steps involved in generating the keyring data required in provisioning are listed below.

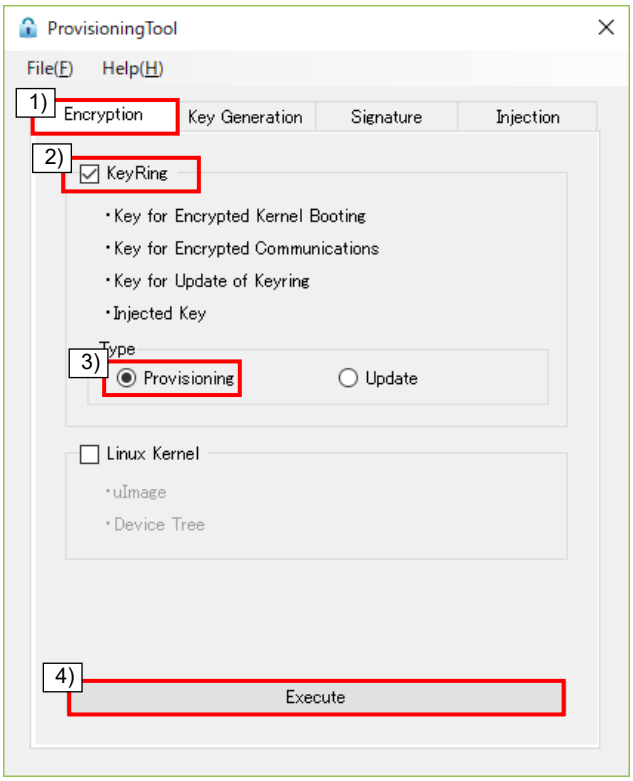


Figure 11-4 Tool Window for Generating Keyring Data

1. Select the [Encryption] tab.
2. Select the checkbox for [KeyRing].
3. Select the checkbox for [Provisioning] in the [Type] section.
4. Click on the [Execute] button.

11.4.4 Generating the Linux Kernel Required in Provisioning and Updating Process

Temporary Encryption of Linux kernel (uImage/DeviceTree) is performed for provisioning process and secure software updating. uImage and DeviceTree in LinuxKernel folder are padded with 0x00 so that the size of uImage and DeviceTree are 16*n bytes. Padded each data is signed with SBS-Key.der. Then signed uImage is encrypted with E1-Key.bin, and signed DeviceTree is encrypted with E2-Key.bin.

Procedure	Algorithm	Key
Signature	RSASSA-PKCS1-v1_5 SHA256	Temporary signature verification key for uImage/DeviceTree (SBS-Key.der)
Encryption	AES-128-CBC	Temporary encryption key for uImage (E1-Key.bin)* Temporary encryption key for DeviceTree (E2-Key.bin)*

note* : Each IV value is retrieved from last 16 bytes of the E1-Key.bin and E2-Key.bin. Refer to Table 11-2.

The steps involved in generating the Linux kernel data required in provisioning and updating are listed below.

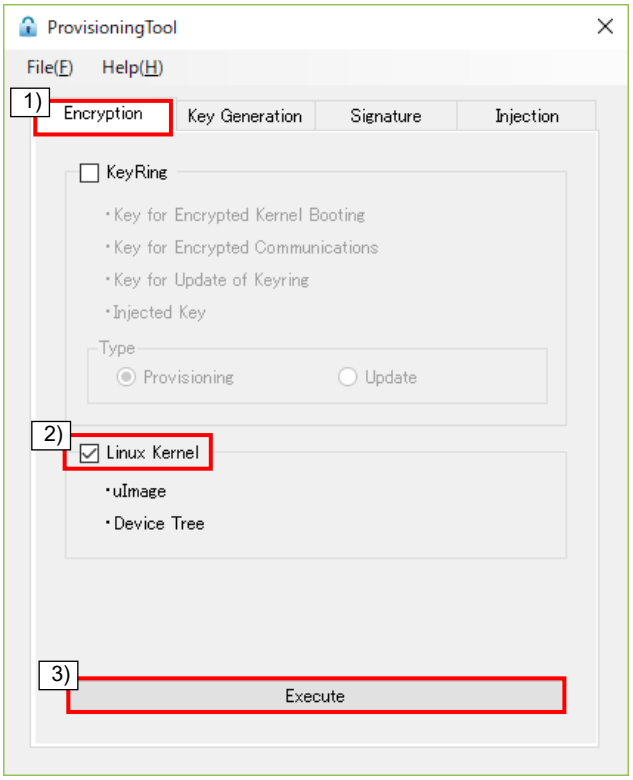


Figure 11-5 Tool Window for Generating the Linux Kernel Data

1. Select the [Encryption] tab.
2. Select the checkbox for [Linux Kernel].
3. Click on the [Execute] button.

11.4.5 Generating the Keyring Data Required in Updating

Keyring creation is performed from KeyRing folder for use in Secure Software Update. Keys in UpdateKey folder and KeyRing folder is used in this procedure. Each key in the KeyRing folder is binary concatenated according to format of keyring. Provisioning Tool calculates MAC of this keyring with SS-UP2-Key.bin and MAC is added keyring data. Then tool encrypts MAC added keyring with SS-UP2-Key.bin.

Procedure	Algorithm	Key	IV
MAC	CBC-MAC with AES-128	MAC key for keyring(SS-UP2-Key.bin)	0
Encryption	AES-128-CBC	Temporary encryption key for keyring (SS-UP1-Key.bin)	IV0*

note*: IV0=0x85c1673483d5d291f0d0713e3ea434a3

The steps involved in generating the keyring data required in updating are listed below.

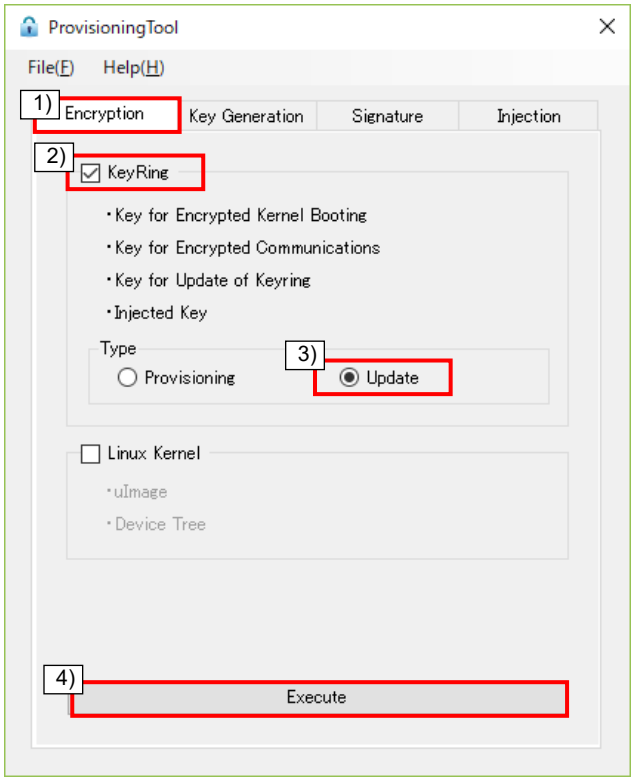


Figure 11-6 Tool Window for Generating the Keyring Data Required in Updating

1. Select the [Encryption] tab.
2. Select the checkbox for [Keyring]
3. Select the checkbox for [Update] in the [Type] section.
4. Click on the [Execute] button.

11.4.6 Key Data Injection

Temporary Encryption is performed for user key injection. Provisioning Tool calculates MAC for each key in Injection folder with BCF2-Key.bin. Each key data added MAC is encrypted with BCF1-Key.bin.

Procedure	Algorithm	Key	IV
MAC	CBC-MAC with AES-128	MAC key for user key (BCF2-Key.bin)	0
Encryption	AES-128-CBC	Temporary encryption key for user key (BCF1-Key.bin)	*

note*: Automatically created by Provisioning Tool.

The steps involved in injection process for using user key on RZ/G are listed below.

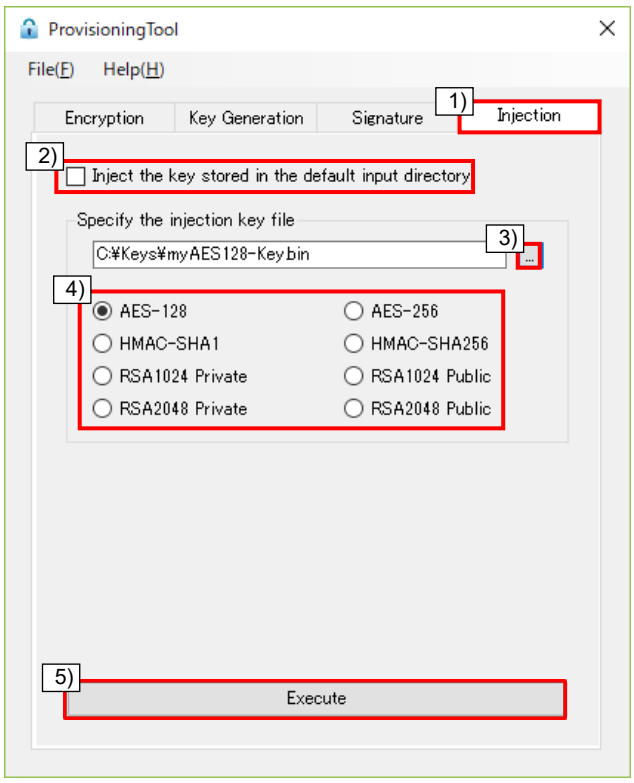


Figure 11-7 Tool Window for Key Data Injection

1. Select the [Injection] tab.
2. Uncheck the “Inject the key stored in the default input directory” check box.*note
3. Click on the button for open file dialog, and select user key file on dialog box.
4. Select the encryption method in the “Specify the injection key file” section.
5. Click on the [Execute] button.

note*: If you check the “Inject the key stored in the default input directory” and click on the [Execute] button, key data files in Input/injection folder is selected automatically as key data to be subjected to injection.

11.4.7 Display of an Error Message

If a required file does not exist, an error message of the type below appears.

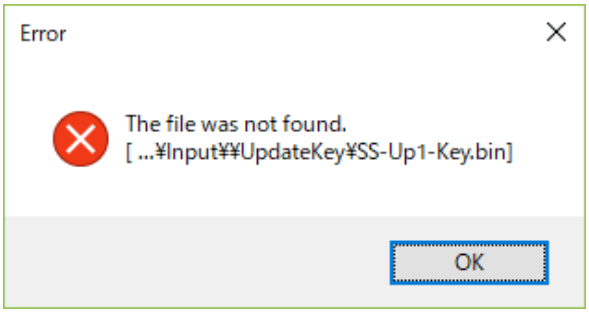


Figure 11-8 Error Message

Check the specified path for the given file, place the file there, and execute the tool again.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar.01, 2021	-	First release (Based on documents in Package V1.1.1)

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/.