Renesas RA Family

Device Lifecycle Management for Cortex-M33

Introduction

Device Lifecycle Management (DLM) is the management of the process by which a product goes from inception to development to production and then eventually end-of-life. The RA Family MCU debug capability and serial programming capability are defined by the device lifecycle states.

The RA Family MCUs with an Arm® Cortex®-M33 core leverage Arm® TrustZone® technology with authenticated DLM state transitions. This DLM system offers enhanced IP protection while maintaining the capability to regress the device lifecycle state.

This application note focuses on the use cases of the TrustZone-enabled RA Family MCU DLM System, walks through the DLM Key injection process, and demonstrates DLM state regression steps.

The RA6M4 MCU Group is used as an example. The general steps apply to all TrustZone-enabled RA Family MCUs.

Required Resources

The following resources are referenced throughout this application note:

Development Tools and Software

- Renesas Flash Programmer (RFP) v3.16
  https://www.renesas.com/us/en/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html
- Renesas Security Key Management Tool
  https://www.renesas.com/software-tool/security-key-management-tool
- Gpg4win (optional)
  http://www.gpg4win.org/

Hardware

- EK-RA6M4, Evaluation Kit for RA6M4 MCU Group (renesas.com/ra/ek-ra6m4)
- PC running Windows® 10 OS
- One USB device cable (type-A male to micro-B male)

Prerequisites and Intended Audience

This application note assumes you have some experience with the Renesas Flash Programmer (RFP). In addition, the application note assumes that you have some knowledge of RA Family MCU security features. See chapter 49, *Security Features*, in the *Renesas RA6M4 Group MCU User's Manual: Hardware* for background knowledge. User is also recommended to read the Security Key Management Tool (SKMT) User's Manual prior to proceeding with this application note.

The intended audience are product developers, product manufacturers, product support, or end users who are involved with any stage of the device lifecycle management of the RA Family MCUs.

Contents

1. Introduction to Device Lifecycle Management for RA Family MCU Groups

The RA Family DLM system provides an infrastructure that can play a key role in application development, production, product deployment, and failure analysis. Device Lifecycle Management is supported on RA Family MCUs with Arm® TrustZone®.

1.1 Device Lifecycle Management Overview

Device Lifecycle Management (DLM) identifies the current lifecycle phase of the device and controls access permissions to the debug interface and the serial programming interface. These RA Family MCUs offer cryptographic authentication to restore various levels of programming and debugging capabilities.

1.2 Device Lifecycle Management on RA Family MCUs

Figure 1 is a summary of all the possible states and state transitions for the entire lifecycle of the MCU, as described in Table 1.

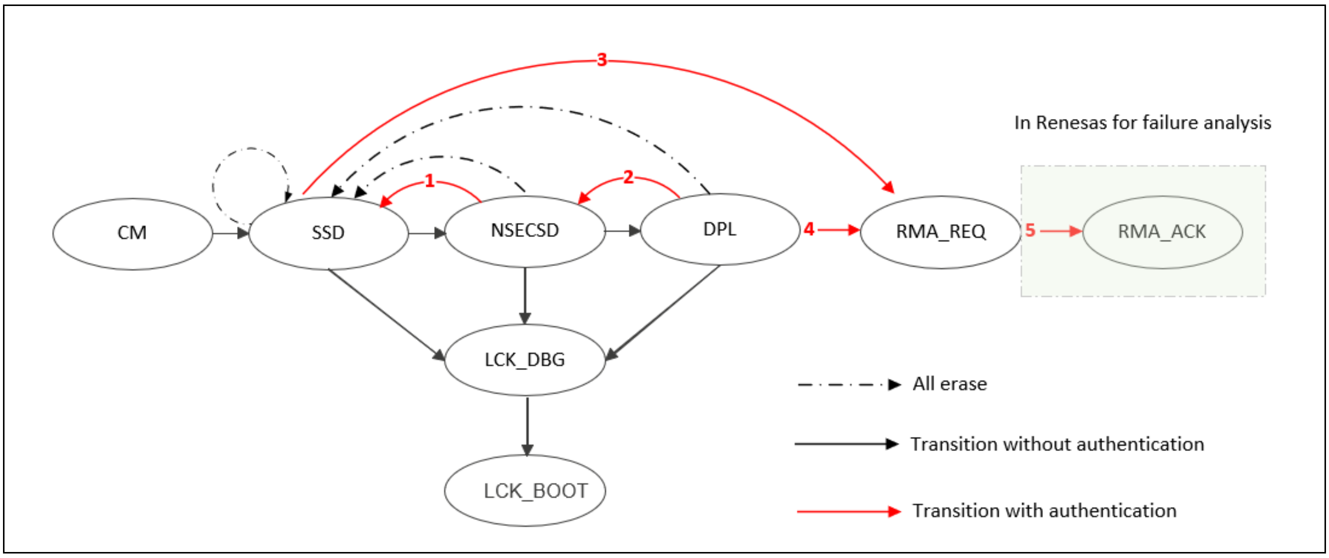

3
In Renesas for failure analysis
1
2
CM
SSD
NSECSD
DPL
4
RMA_REQ
5
RMA_ACK
LCK_DBG
LCK_BOOT
All erase
Transition without authentication
Transition with authentication

Figure 1. RA MCU Device Lifecycles

1.2.1 Definitions of the Device Lifecycle States

There are three debug levels for RA Family MCUs with the Cortex-M33 core and TrustZone support:

- DBG2: The debugger connection is allowed, with no restriction on access to memories and peripherals
- DBG1: The debugger connection is allowed, with access to only non-secure memory regions and peripherals
- DBG0: The debugger connection is not allowed

The serial programming interface can communicate with the factory bootloader of the device when the device is in boot mode. Tools such as the Renesas Flash Programmer (RFP) can communicate with the factory bootloader via the serial programming interface to change the device lifecycle states. Development tools such as the e^2 studio Interactive Development Environment provide partial support for the DLM state transitions that are typically used during product development.

Table 1 provides definitions of the various device lifecycle states, the corresponding debug level, and serial programming capability.

Table 1. TrustZone-enabled RA Family MCU Group Device Lifecycle States

Lifecycle state	Definition and State Features	Debug Level	Serial Programming
CM	• "**C**hip **M**anufacturing" • The customer may receive the device in this state. If so, the customer needs to move the state to SSD prior to application development.	DBG2	• Available • No access to code/data flash

Lifecycle state	Definition and State Features	Debug Level	Serial Programming
SSD	• "**S**ecure **S**oftware **D**evelopment" • The secure part of the application is being developed. The customer may receive the device in SSD state.	DBG2	• Available • Can program/erase/read all code/data flash
NSECSD	• "**N**on-**SEC**ure **S**oftware **D**evelopment" • The non-secure part of the application is being developed	DBG1	• Available • Can program/erase/read non-secure code/data flash
DPL	• "**D**e**PL**oyed" • The device is in-field	DBG0	• Available. • No access to code/data flash
LCK_DBG	• "**L**o**CK**ed **D**e**B**u**G**" • The debug interface is permanently disabled	DBG0	• Available • No access to code/data flash
LCK_BOOT	• "**L**o**CK**ed **BOOT** interface" • The debug interface and the serial programming interface are permanently disabled	DBG0	• Not available
RMA_REQ	• "**R**eturn **M**aterial **A**uthorization **REQ**uest" • Request for RMA • The customer must send the device to Renesas in this state	DBG0	• Available • No access to code/data flash
RMA_ACK	• "**R**eturn **M**aterial **A**uthorization **ACK**nowledged" • Failure analysis at Renesas	DBG2	• Available • No access to code/data flash

1.2.2 Summary of the Device Lifecycle Transitions

As we can see from Figure 1, there are three types of transitions.

Non-Authenticated Transitions

These transitions follow a typical device flow from development to production. Each transition reduces the level of access allowed to the MCU.

"All erase" Transitions

The factory bootloader's **Initialize** command will erase the entire flash memory and set the device lifecycle to SSD. The **Initialize** command is not available under the following conditions:

- Permanent Block Protection has been set on one or more flash memory blocks. Permanent block protection cannot be revoked.
- The **Initialize** command has been disabled. Refer to section 3.2 Figure 7 on how to disable the **Initialize** command using RFP. If the **Initialize** command is disabled, it cannot be re-enabled.

Authenticated Transitions

The lifecycle states can be regressed without erasing the contents of the MCU flash memory via authenticated transitions. These authenticated transitions are desirable for failure analysis of the protected code. Authentication is performed with the use of user-defined DLM keys.

1.2.3 DLM Keys

Authenticated transitions are possible through the use of DLM keys. These user-defined keys are injected during specific device lifecycle states to allow authenticated regression back to that state. Authentication is performed via a challenge-response mechanism, protecting against eavesdropping and replay attacks.

The primary keys that most applications will use are:

- SECDBG_KEY – The Secure Debug Key can be injected when the MCU is in the SSD state. It can be used when the device is in the NSECSD state to regress back to the SSD state without erasing flash memory.
- NONSECDBG_KEY – The Non-secure Debug Key can be injected when the MCU is in the NSECSD state. It can be used when the device is in the DPL state to regress back to the NSECSD state without erasing flash memory.

Note that authenticated regression from DPL back to SSD must be done in two steps.
One additional key is available, which can be used to send the MCU to Renesas for failure analysis through the RMA process:

- RMA_KEY – The RMA Key can be injected when the MCU is in the SSD state. It can be used when the device is in either the SSD state or the DPL state to prepare the MCU for return to Renesas.

1.3 Advantages of Device Lifecycle Management

Device Lifecycle Management provides the following advantages over simple ID Code protection:

- Authenticated re-enabling of the debug and the serial programming interfaces protects IP by providing protection against eavesdropping and replay attacks.
- A Root of Trust or other sensitive code can be programmed and stored securely whilst retaining the ability to erase all flash contents. This avoids unnecessary MCU scrappage.
- The separation of Secure and Non-secure development with code protection for both regions enables new business models for providing valuable IP.

2. Device Lifecycle Management Use Cases

This section discusses some of the use cases for Device Lifecycle Management with RA Family TrustZone-enabled MCUs.

2.1 Immutable Root of Trust Provisioning

A Root of Trust (RoT) consists of data and services that provide a unique identity to a product. As such, it is vital to protect the code and data that are used to implement the Root of Trust. Arm TrustZone provides an excellent mechanism for isolating the Root of Trust from the rest of the application.

An important point to note is that the Root of Trust is typically immutable. This is done by using the permanent block protect feature of the MCU. Since permanent block protection disables the **Initialize** command, some DLM capabilities will not be available in this use case.

2.1.1 Development Phase

For MCUs, the Root of Trust is often implemented by the same development team that implements the entire application (that is, the Combined Development Model). During development, the Root of Trust can be programmed into the secure region with the device in the SSD state. The DLM state should then be changed to the NSECSD state for application development, and finally to DPL for system test. However, since all parties are trusted, DLM key usage during development is probably not necessary. If a problem is found with either the Root of Trust or the application, the MCU can simply be initialized to erase all flash memory and return the DLM state to the SSD state.

Note: Do not permanently lock flash blocks at this stage to make the Root of Trust immutable. This will disable the **Initialize** command, and the MCU will not be able to be reprogrammed. For testing purposes, use the temporary block protect mechanism.

2.1.2 Production Phase

Production programming with a TrustZone-enabled application is often a two-step process. For the Root of Trust use case, the Root of Trust (in the secure region) will likely be provisioned separately from the application layer (in the non-secure region) to enable the injection of infrastructure keys.

The following shows the DLM options for a deployed MCU for the Root of Trust use case. For this use case, it is assumed that the RoT (Root of Trust) has been made immutable, rendering the **Initialize** command unavailable. Note that deploying the product in the DPL state with no DLM keys is not useful, since the device can no longer be completely erased

Table 2. RoT Use Case, Possible Production DLM States

DLM State	Initialize Command	DLM Keys	Capabilities after Programming
DPL	Unavailable	SECDBG_KEY NSECDBG_KEY	Authenticated DLM state regression, additional limited serial programming interface access (1).
LCK_DBG	Unavailable	None	Limited serial programming interface access (1).
LCK_BOOT	Unavailable	None	No access

Note: 1. MCU Unique ID, MCU type (for example, R7FA6M4AF3CFB), and factory boot firmware version available.

If the end-product firmware will not be debugged in case of a problem with the end product, the LCK_BOOT or LCK_DBG states are recommended. If the end-product firmware may be debugged, the DPL state with SECDBG_KEY and NSECDBG_KEY injected should be used.

2.2 IP Protection

The IP Protection use case consists of valuable IP provided in a pre-programmed MCU to another party, who then develops the end application. IP protection is critical, but it is also helpful to retain the ability to reprogram the IP to avoid MCU scrappage.

2.2.1 Development – Valuable IP

The valuable IP will be placed in the secure region, with the MCU in the SSD state. Upon delivery to the second party, the DLM state will be moved to NSECSD, but there are some options available, as described by the following table.

Table 3. IP Protection Use Case, Possible IP Delivery DLM States

DLM State	Initialize Command	DLM Keys	Capabilities after Programming
NSECSD	Available	None	Valuable IP is protected, but it can be erased by a second party, including accidental erasure. Cannot return to SSD state to debug Valuable IP with the application.
NSECSD	Available	SECDBG_KEY	Valuable IP is protected, but it can be erased by a second party, including accidental erasure. Can perform authenticated transition to SSD state to debug Valuable IP with the application.
NSECSD	Disabled	None	Valuable IP is protected, and it cannot be erased by the second party. Cannot return to SSD state to debug Valuable IP with the application.
NSECSD	Disabled	SECDBG_KEY	Valuable IP is protected, and it cannot be erased by the second party. Can perform authenticated transition to SSD state to debug Valuable IP with the application.

2.2.2 Development – Application

The second party will receive the device in the NSECSD state. They can then develop and debug the application code that uses the IP, but they cannot view the IP or any data that has been placed in the secure region.

To test the full application in a production configuration (i.e., the DPL state), it is recommended to inject a NONSECDBG_KEY DLM Key for the following reasons:

- If the pre-programmed MCU was delivered with the **Initialize** command disabled, there is no way to return from the DPL state to the NSECSD state without using an authenticated transition with the NONSECDBG_KEY.
- If the pre-programmed MCU was delivered with the **Initialize** command enabled, using the **Initialize** command will erase both secure and nonsecure regions, including the purchased IP.

2.2.3 Production – Final Product

Since the usage of the SECDBG_KEY and NSECDBG_KEY and the configuration of the **Initialize** command are all independent, there are eight variations on delivering the end product in the DPL state. Table 4 summarizes the options. Note that some of the options depend on the state in which the pre-programmed MCU is delivered.

Table 4. IP Protection Use Case, Possible Production DLM States

DLM State	Initialize Command	DLM Keys	Capabilities after Programming
DPL	Disabled	SECDBG_KEY NSECDBG_KEY	Authenticated DLM state regression to NSECSD and SSD states, additional limited serial programming interface access (1).
DPL	Disabled	NSECDBG_KEY	Authenticated DLM state regression to NSECSD state, additional limited serial programming interface access (1).
DPL	Disabled	SECDBG_KEY	Limited serial programming interface access (1).
DPL	Disabled	None	Limited serial programming interface access (1).
DPL	Enabled	SECDBG_KEY NSECDBG_KEY	Authenticated DLM state regression to NSECSD and SSD states, the device can be **Initialized**, additional limited serial programming interface access (1).
DPL	Enabled	NSECDBG_KEY	Authenticated DLM state regression to NSECSD state, the device can be **Initialized**, additional limited serial programming interface access (1).
DPL	Enabled	SECDBG_KEY	Device can be **Initialized**, additional limited serial programming interface access (1).
DPL	Enabled	None	Device can be **Initialized**, additional limited serial programming interface access (1).
LCK_DBG	Unavailable	None	Limited serial programming interface access (1).
LCK_BOOT	Unavailable	None	No access

Note: 1. MCU Unique ID, MCU type (for example, R7FA6M4AF3CFB), and factory boot firmware version available.

2.3 Non-TrustZone Usage

Arm TrustZone usage on an RA Family MCU is optional. The Flat Project Development Model supports the use case of creating an application without TrustZone awareness. (Note that TrustZone register configuration is still recommended and is required for ethernet applications.) However, DLM keys are still useful for the ability to provide authenticated re-enabling of the debug and serial programming interfaces.

2.3.1 Development Phase

It is not necessary to use DLM keys during development. In a Flat Project, all development is done in the SSD state. To test the application as it will be delivered in a production state, change the DLM state to DPL. To reprogram the device, issue the **Initialize** command.

Note: Do not permanently lock any flash blocks at this stage. This will disable the **Initialize** command, and the MCU will not be able to be reprogrammed. For testing purposes, use the temporary block protect mechanism.

2.3.2 Production Phase

If it is desired to enable authenticated debugging, the SECDBG_KEY and NSECDBG_KEY must both be programmed. For simplicity, the same value can be used for both keys. However, the regression will be a two-step process: DPL to NSECSD, then NSECSD to SSD.

If the end product may need to be reprogrammed but retention of the existing code flash is not required, the DPL state can be used without DLM keys.

Table 5 summarizes the options.

Table 5. Non-TrustZone Use Case, Possible Production DLM States

DLM State	Initialize Command	DLM Keys	Capabilities after Programming
DPL	Disabled or Unavailable	SECDBG_KEY NSECDBG_KEY	Authenticated DLM state regression, additional limited serial programming interface access (1).
DPL	Disabled or Unavailable	None	Limited serial programming interface access (1).
DPL	Enabled	SECDBG_KEY NSECDBG_KEY	Authenticated DLM state regression to NSECSD and SSD states, the device can be **Initialized**, additional limited serial programming interface access (1).
DPL	Enabled	None	Device can be **Initialized**, additional limited serial programming interface access (1).
LCK_DBG	Unavailable	None	Limited serial programming interface access (1).
LCK_BOOT	Unavailable	None	No access

Note: 1. MCU Unique ID, MCU type (e.g., R7FA6M4AF3CFB), and factory boot firmware version available.

2.4 Renesas RMA

To submit the MCU for failure analysis by Renesas through the RMA process, an RMA_KEY must be injected, and the device must be in either the SSD state or the DPL state prior to performing a transition to the RMA_REQ state. This transition will erase all code and data flash that has not been permanently locked, and the MCU can be delivered to Renesas for failure analysis.

This use case can be combined with all of the above use cases.

3. Non-Authenticated DLM State Transitions

Non-authenticated DLM state transitions do not require the use of DLM keys. These transitions reflect a typical lifecycle flow from development through to production and deployment, with decreasing levels of access with each transition.

3.1 Non-Authenticated DLM State Transition Using the Renesas Device Partition Manager

The Renesas Device Partition Manager is a utility integrated into the e^2 studio IDE for Device Lifecycle State management during development. The Renesas Device Partition Manage can perform the following functions:

- Query the current device lifecycle state
- Query the device IDAU region setup
- **Initialize** the device to the SSD state (unless any flash blocks have been permanently locked)
- Set up IDAU regions

Note: The user must power cycle the board prior to working with the **Renesas Device Partition Manager** after a debug session if J-Link is used as the connection interface.

In the e^2 studio IDE, open the **Renesas Device Partition Manager**.

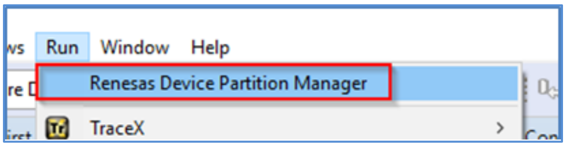
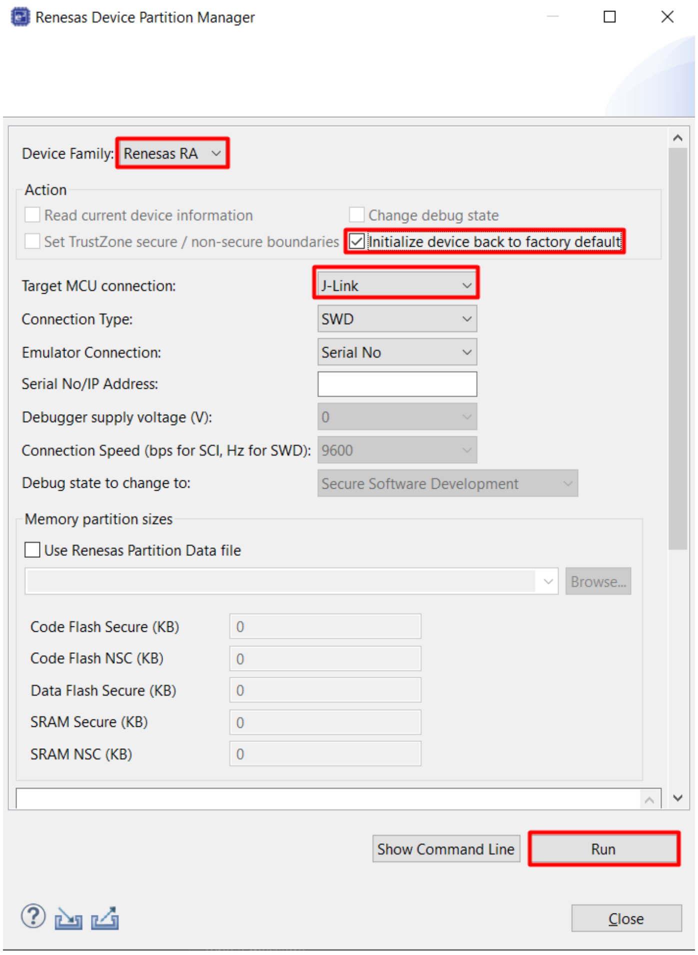


Figure 2. Open the Renesas Device Partition Manager

Figure 3 shows how to **Initialize** the device back to factory default. Choose the connection method and then click **Run**.



Renesas Device Partition Manager

Device Family: Renesas RA

Action

□ Read current device information
□ Change debug state
□ Set TrustZone secure / non-secure boundaries
☑ Initialize device back to factory default

Target MCU connection: J-Link

Connection Type: SWD

Emulator Connection: Serial No

Serial No/IP Address:

Debugger supply voltage (V): 0

Connection Speed (bps for SCI, Hz for SWD): 9600

Debug state to change to: Secure Software Development

Memory partition sizes

□ Use Renesas Partition Data file

Code Flash Secure (KB): 0

Code Flash NSC (KB): 0

Data Flash Secure (KB): 0

SRAM Secure (KB): 0

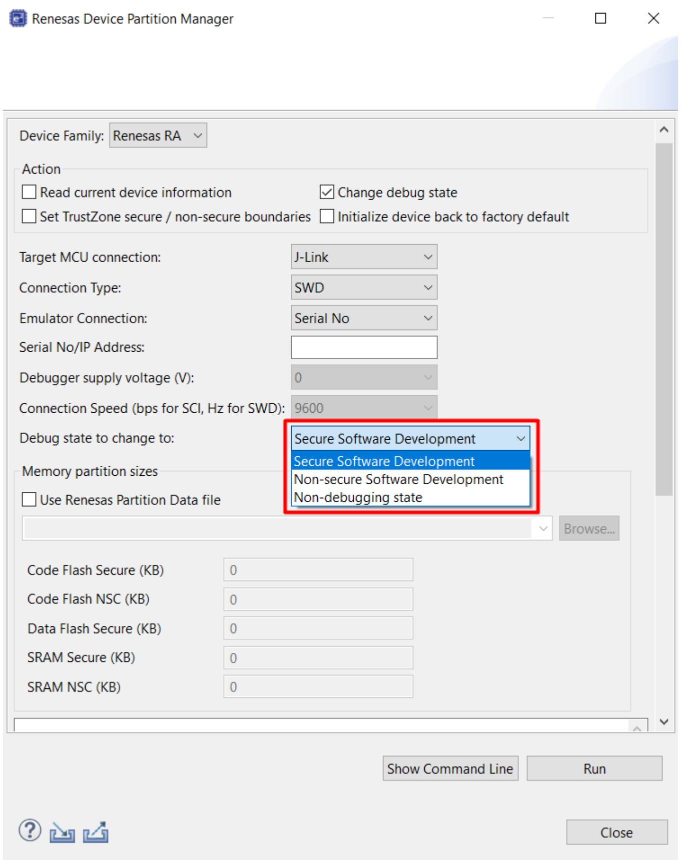
SRAM NSC (KB): 0

Figure 3. Initialize RA6M4 Using Renesas Device Partition Manager

During Product Development, users typically use the **Renesas Device Partition Manager** to perform Device Lifecycle Advancements (SSD to NSECSD, NSECSD to DPL) and to **Initialize** the device (erase all device memories and set the DLM state to SSD).

Limitations with the Renesas Device Partition Manager

- It does not support authenticated transitions.
- It supports transitions to limited device lifecycle states
- It requires the e^2 studio IDE to operate.



Renesas Device Partition Manager

Device Family: Renesas RA

Action

□ Read current device information ☑ Change debug state

□ Set TrustZone secure / non-secure boundaries □ Initialize device back to factory default

Target MCU connection:	J-Link
Connection Type:	SWD
Emulator Connection:	Serial No
Serial No/IP Address:	
Debugger supply voltage (V):	0
Connection Speed (bps for SCI, Hz for SWD):	9600
Debug state to change to:	Secure Software Development

Secure Software Development
Non-secure Software Development
Non-debugging state

Memory partition sizes

□ Use Renesas Partition Data file

Code Flash Secure (KB)	0
Code Flash NSC (KB)	0
Data Flash Secure (KB)	0
SRAM Secure (KB)	0
SRAM NSC (KB)	0

Figure 4. Advance Device Lifecycle States using Renesas Device Partition Manager

3.2 Non-Authenticated DLM State Transitions Using the Renesas Flash Programmer

The Renesas Flash Programmer provides end-to-end production flow support. RFP provides the following functionalities.

- Supports all unauthenticated device lifecycle state transitions.
- Supports DLM key injection.
- Supports authenticated device lifecycle state transitions.
- Supports disabling the **Initialize** command. This may be desired if the device is deployed in a DPL state and there is a requirement to avoid accidental flash content erasing. However, once the **Initialize** command is disabled, it cannot be re-enabled.

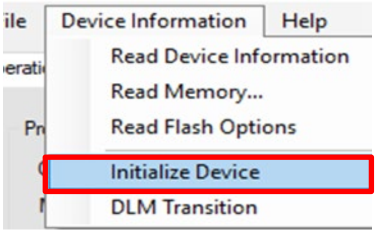
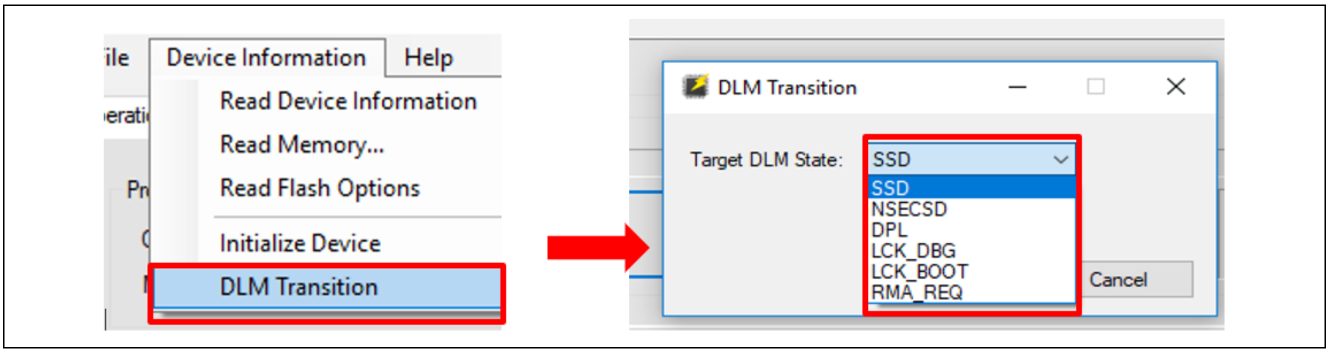


Figure 5. Using the Initialize Device Command

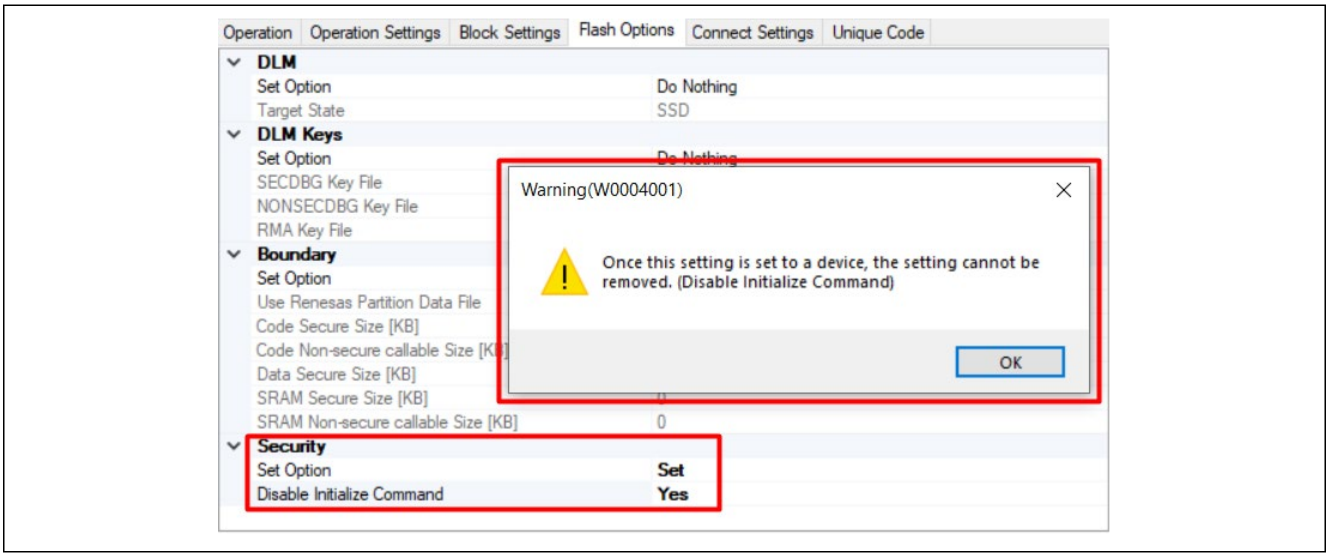
Unauthenticated transitions can be performed as shown in Figure 6.



Target DLM State: SSD

Figure 6. Available Lifecycle State Transitions Supported by RFP

The Initialize command can be disabled, as shown below. This operation cannot be reversed.



DLM	
Set Option	Do Nothing
Target State	SSD
DLM Keys	
Set Option	Do Nothing
SECDBG Key File	
NONSECDBG Key File	
RMA Key File	
Boundary	
Set Option	
Use Renesas Partition Data File	
Code Secure Size [KB]	
Code Non-secure callable Size [KB]	
Data Secure Size [KB]	
SRAM Secure Size [KB]	
SRAM Non-secure callable Size [KB]	0
Security	
Set Option	**Set**
Disable Initialize Command	**Yes**

Warning(W0004001)

Once this setting is set to a device, the setting cannot be removed. (Disable Initialize Command)

Figure 7. Disable the Initialize Command

4. DLM Key Injection

4.1 Tools Used in the DLM Key Injection Process

Three tools are used in the process of DLM key injection.

- Gpg4win - This tool is used to generate and use PGP keys, which are used to establish a secure communication channel between the developer and the Renesas Key Wrap server. Gpg4win is used here for demonstration, but any PGP management tool can be used.
- Renesas Security Key Management Tool (SKMT) - This tool is to generate the various key files needed for DLM key injection.
- Renesas Flash Programmer (RFP) - This tool is used to inject DLM keys.

4.2 DLM Key Injection Process Overview

There are three high-level steps required to inject DLM keys into the MCU.

1. Create an arbitrary 256-bit User Factory Programming Key (UFPK) and obtain a Wrapped UFPK (W-UFPK). The UFPK is used to encrypt the DLM key, to ensure that no plaintext keys are exposed

during the injection process. The UFPK must be wrapped by the Renesas Key Wrapping Service to obtain the W-UFPK.

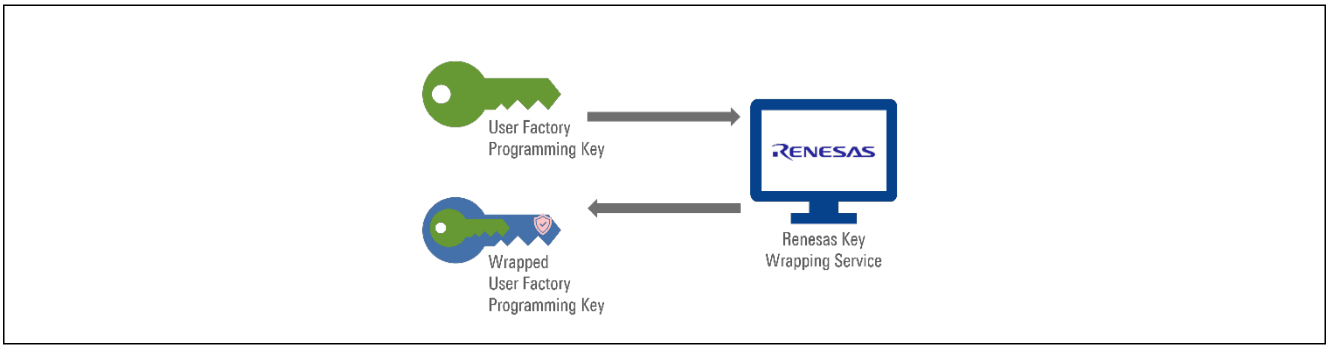

User Factory Programming Key
RENESAS
Renesas Key Wrapping Service
Wrapped User Factory Programming Key

Figure 8. Wrap the User Factory Programming Key (UFPK)

2. Wrap the DLM key using the UFPK.

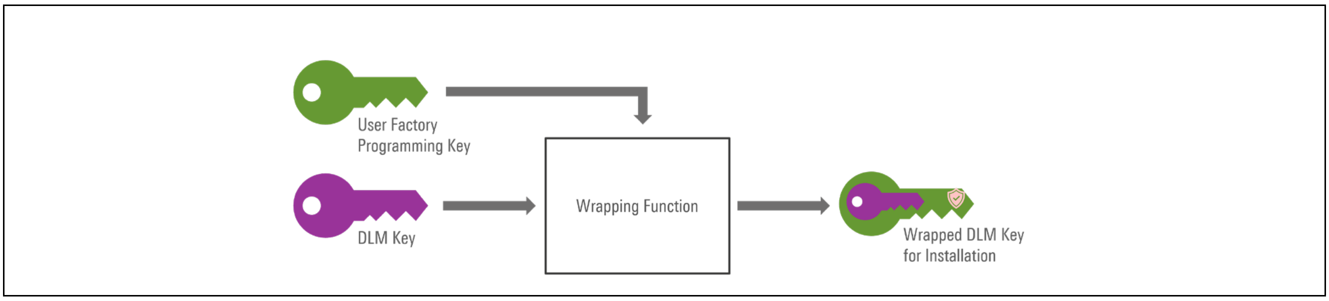

User Factory Programming Key
DLM Key
Wrapping Function
Wrapped DLM Key for Installation

Figure 9. Wrap the DLM Key Using UFPK

3. Inject the DLM key using the serial programming interface by providing the W-UFPK and the wrapped DLM key. Note that this is a conceptual representation; after the wrapped DLM key is generated by SKMT and injected by the RFP, it is not necessary to inject the W-UFPK through the RFP. The information in the W-UFPK is included in the generated wrapped DLM Key.

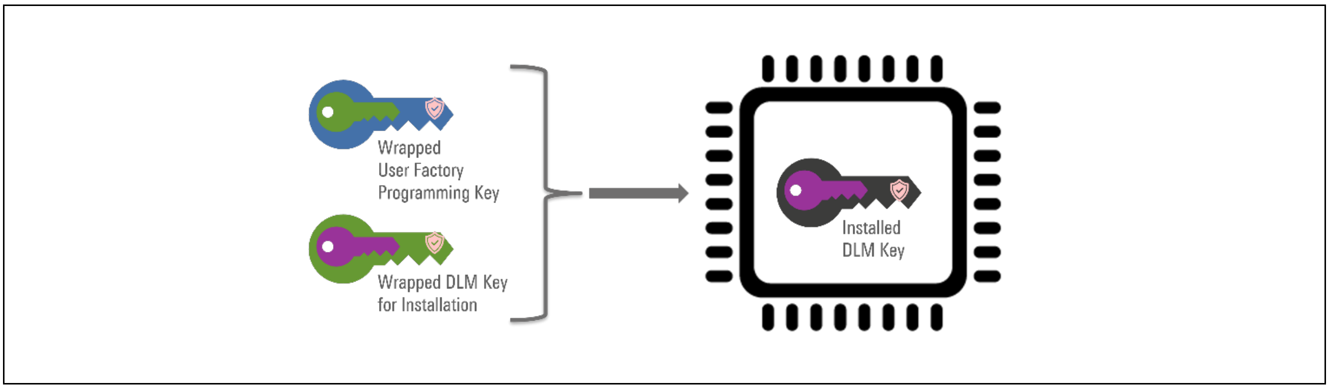

Wrapped User Factory Programming Key
Wrapped DLM Key for Installation
Installed DLM Key

Figure 10. Created Wrapped DLM Keys

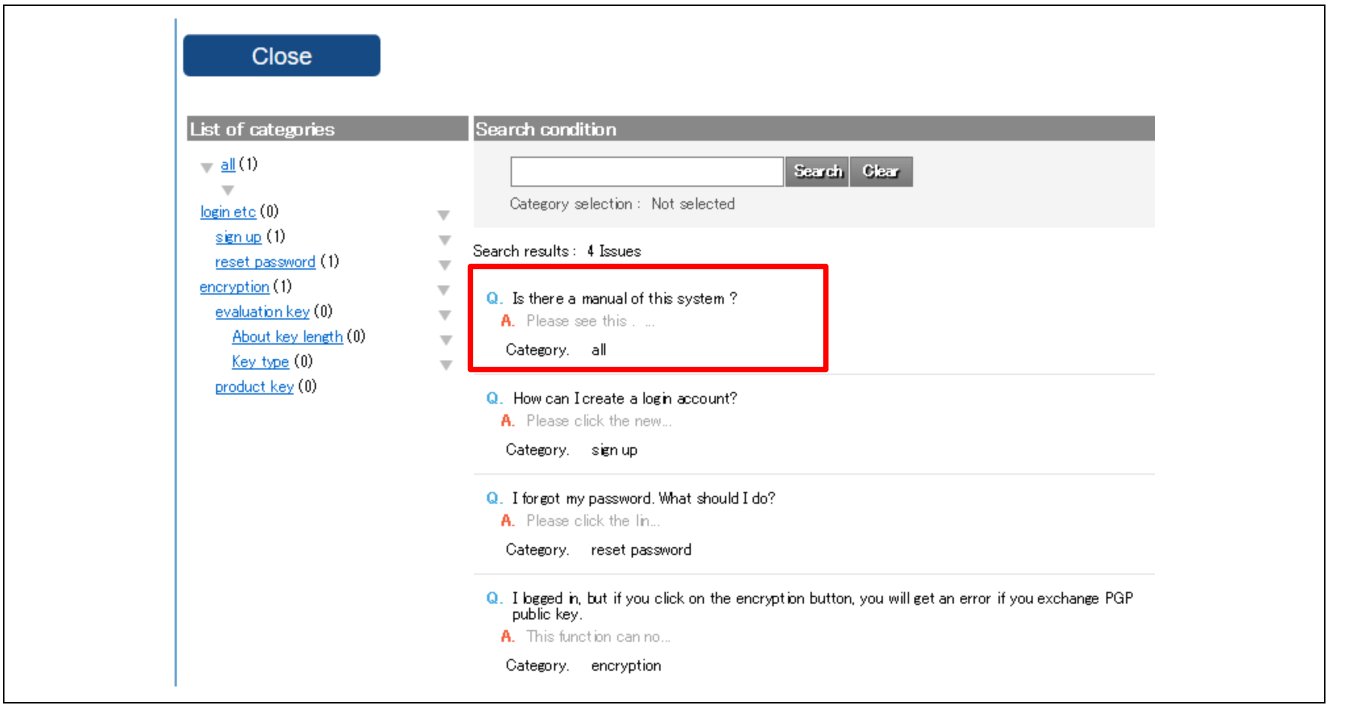
4.3 Establish PGP-Encrypted Communication with the Renesas DLM Server

Key material is exchanged with the Renesas DLM server using PGP encryption. This requires an exchange of public keys. This is a one-time process.

4.3.1 Overview of Device Lifecycle Management (DLM) Server

Using a web browser, enter the URL https://dlm.renesas.com/ to access the Renesas DLM server.

The *Key Wrap Service User's Manual* can be accessed by clicking on the FAQ link on the right of the screen. The link to the *DLM server User's Manual* is in the answer to the FAQ question, "Is there a manual of this system?" as shown in Figure 11.



Close

List of categories

- all (1)
- login etc (0)
  - sign up (1)
  - reset password (1)
- encryption (1)
  - evaluation key (0)
    - About key length (0)
    - Key type (0)
  - product key (0)

Search condition

Search Clear

Category selection : Not selected

Search results : 4 Issues

Q. Is there a manual of this system ?
A. Please see this . ..
Category. all

Q. How can I create a login account?
A. Please click the new...
Category. sign up

Q. I forgot my password. What should I do?
A. Please click the lin...
Category. reset password

Q. I logged in, but if you click on the encryption button, you will get an error if you exchange PGP public key.
A. This function can no...
Category. encryption

Figure 11. DLM Server FAQ and User's Manual

The information communicated between the user and the DLM server needs to be encrypted by OpenPGP. Figure 12 shows an overview of the operational flow for the Key Wrapping service with OpenPGP encryption as a security measure to protect the DLM Keys in transit.

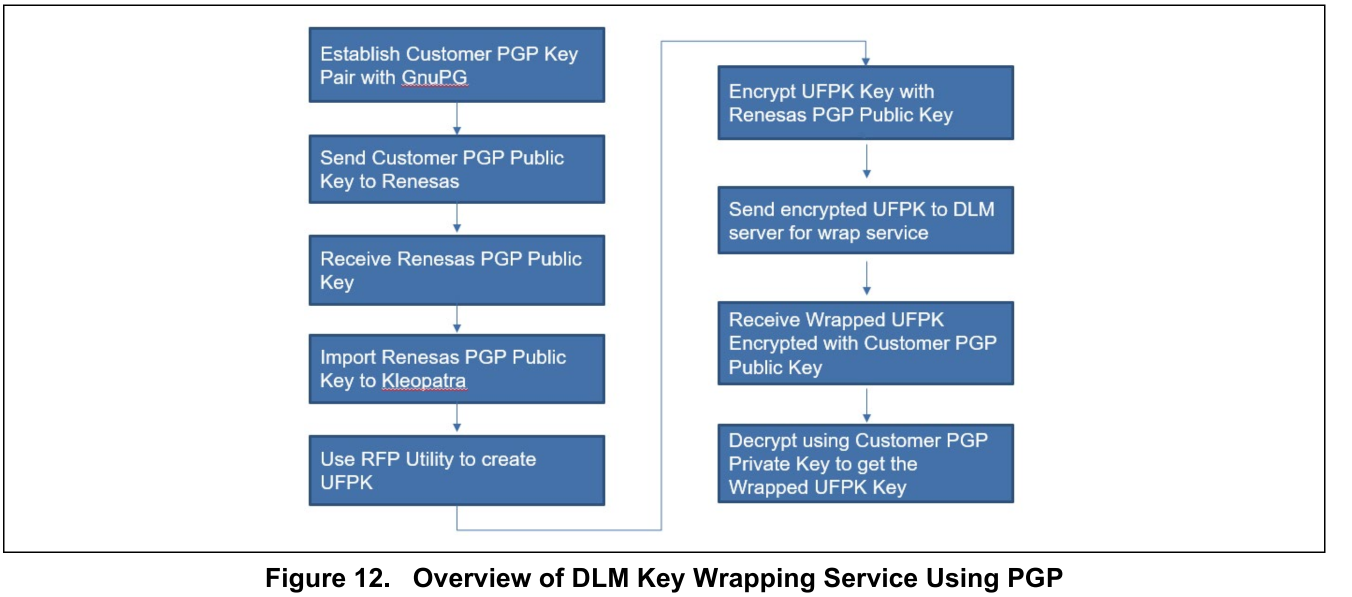

Establish Customer PGP Key Pair with GnuPG
Send Customer PGP Public Key to Renesas
Receive Renesas PGP Public Key
Import Renesas PGP Public Key to Kleopatra
Use RFP Utility to create UFPK
Encrypt UFPK Key with Renesas PGP Public Key
Send encrypted UFPK to DLM server for wrap service
Receive Wrapped UFPK Encrypted with Customer PGP Public Key
Decrypt using Customer PGP Private Key to get the Wrapped UFPK Key

Figure 12. Overview of DLM Key Wrapping Service Using PGP

4.3.2 Create a PGP Key Pair

If you already have a PGP key pair, you can skip this step. Otherwise, use the following steps to create an OpenPGP key pair.

This Application Note uses Gpg4win, the official GnuPG distribution for Windows®, for the PGP key generation, encryption, and decryption service. Note that the generated private key needs to be managed securely and protected from exposure.

Gpg4Win can be downloaded from this URL: http://www.gpg4win.org/

Kleopatra is the certificate manager and unified crypto GUI provided in Gpg4Win. The screenshots included in this application note are based on Gpg4win-4.3.1. There may be minor graphic interface updates with later versions; however, the functionality used in this application note should persist.

1. Launch Kleopatra. Click File->New OpenPGP Key Pair…

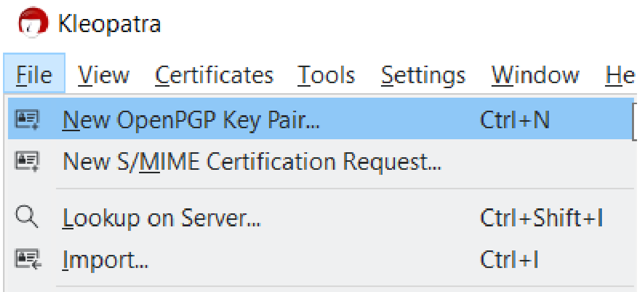
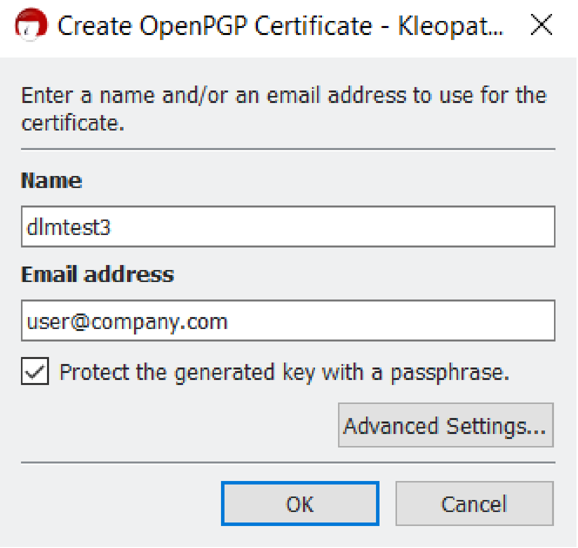

Kleopatra
File View Certificates Tools Settings Window He
New OpenPGP Key Pair... Ctrl+N
New S/MIME Certification Request...
Lookup on Server... Ctrl+Shift+I
Import... Ctrl+I

Figure 13. Generate PGP Key Pair

2. Enter your details as shown. Check Protect the generated key with a passphrase.


Create OpenPGP Certificate - Kleopat...

Enter a name and/or an email address to use for the certificate.

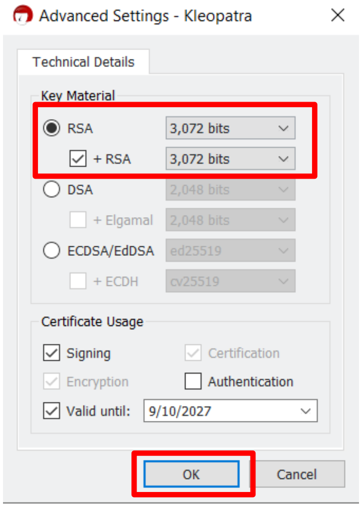
Name: dlmtest3

Email address: user@company.com

☑ Protect the generated key with a passphrase.

Figure 14. Provide Credentials to Create the PGP Key Pair

3. Click **Advanced Settings** and select **RSA** and **3,072 bits**. Click **OK** and then click **Create**.


Advanced Settings - Kleopatra

Technical Details

Key Material

● RSA 3,072 bits

☑ + RSA 3,072 bits

○ DSA 2,048 bits

□ + Elgamal 2,048 bits

○ ECDSA/EdDSA ed25519

□ + ECDH cv25519

Certificate Usage

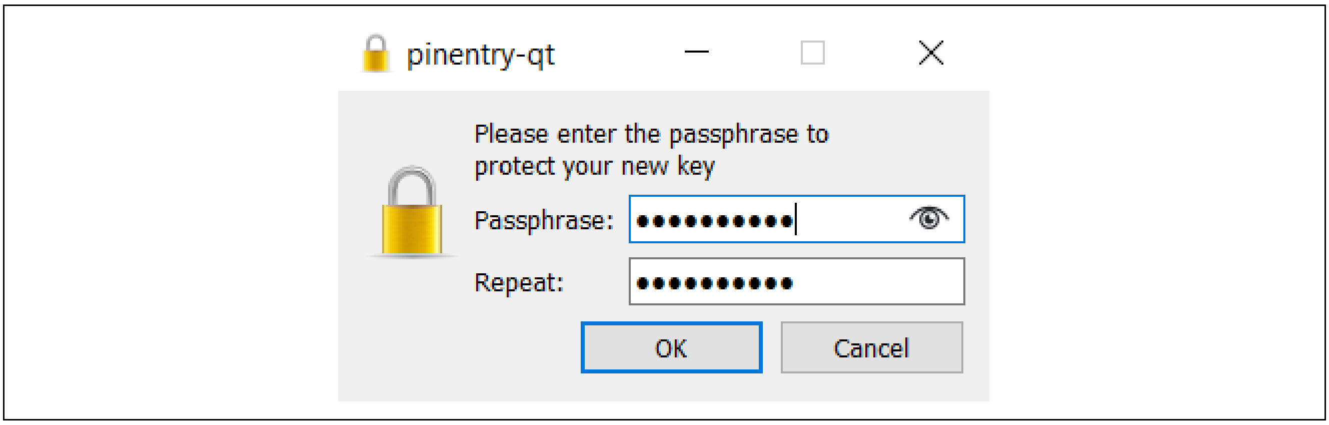
☑ Signing ☑ Certification

☑ Encryption □ Authentication

☑ Valid until: 9/10/2027

Figure 15. Advanced Settings for the PGP Key pair

4. Provide a passphrase to protect the private key. Be sure to save your passphrase.



pinentry-qt

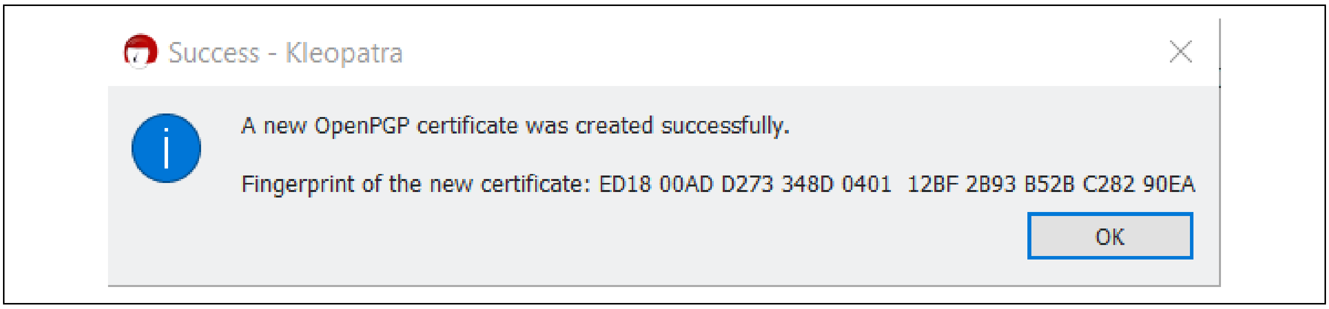
Please enter the passphrase to protect your new key

Passphrase: ●●●●●●●●●●

Repeat: ●●●●●●●●●●

Figure 16. Provide passphrases

5. Observe that the PGP key pair is created successfully.



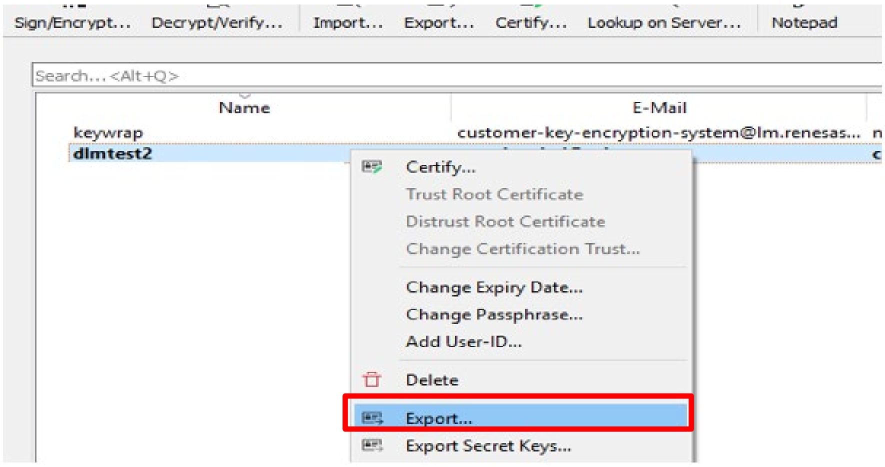
Success - Kleopatra

A new OpenPGP certificate was created successfully.

Fingerprint of the new certificate: ED18 00AD D273 348D 0401 12BF 2B93 B52B C282 90EA

Figure 17. Key Pair Successfully Created

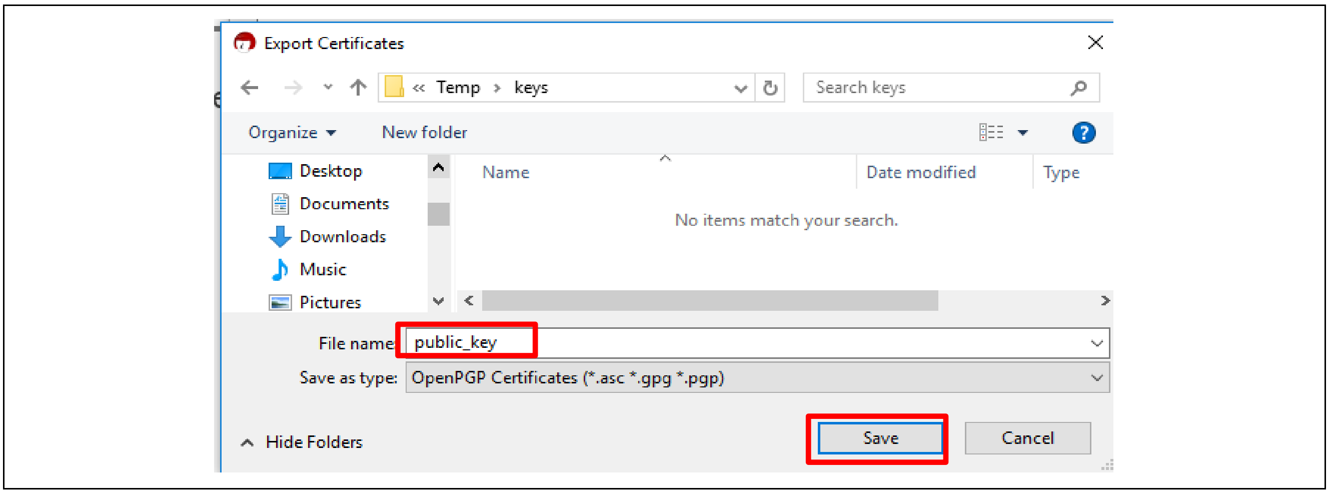
6. Export your public key by right clicking the name you just created and selecting **Export**, as shown below.



Name	E-Mail
keywrap	customer-key-encryption-system@lm.renesas...
dlmtest2	

Figure 18. Export the Public key

7. Save your public key to a file with `*.asc` extension, for example `public_key.asc`.



Export Certificates

Temp > keys

Search keys

File name: public_key

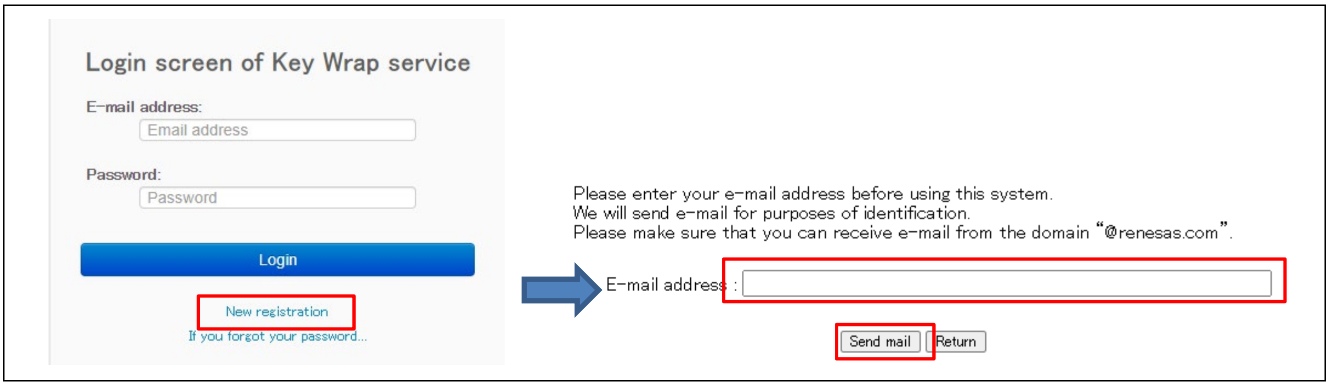
Save as type: OpenPGP Certificates (*.asc *.gpg *.pgp)

Figure 19. Save the Public Key

4.3.3 Registration with DLM Server

This section provides a brief walk-through of the DLM server registration steps. This is a one-time process for new users. You can also review the section “New registration” in the DLM user manual for further details on the new registration steps.

Open the URL https://dlm.renesas.com/ in a web browser and click **New registration**. Follow the prompt to provide an email address and click **Send mail**.

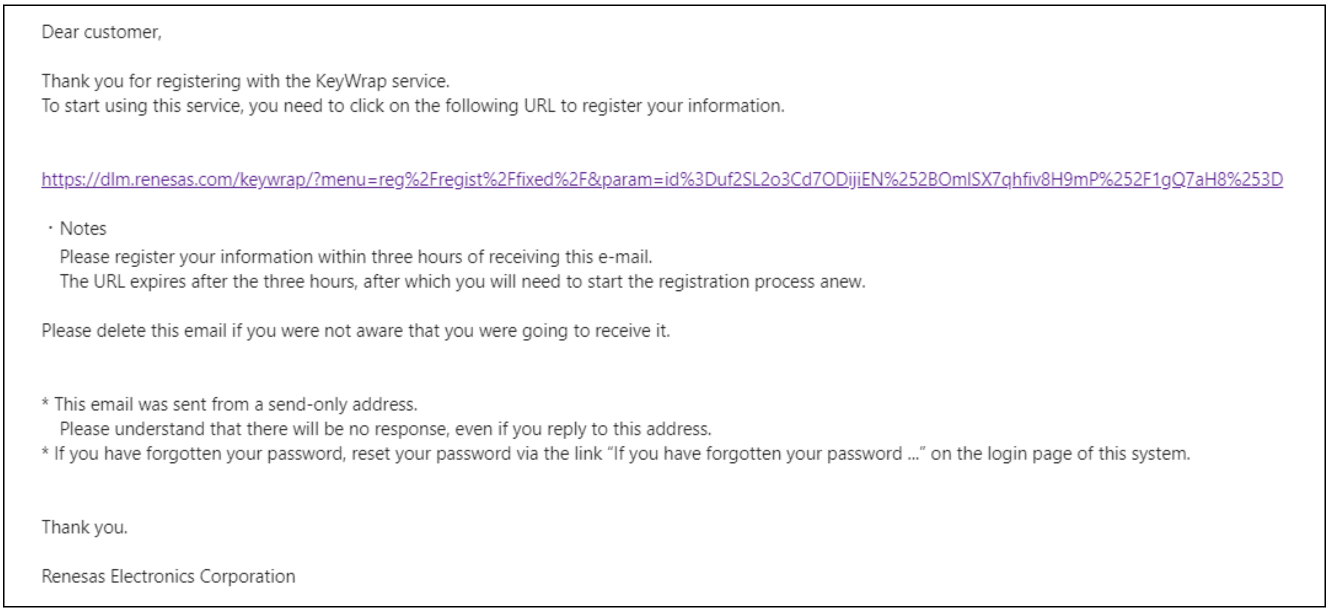


Login screen of Key Wrap service

Please enter your e-mail address before using this system.
We will send e-mail for purposes of identification.
Please make sure that you can receive e-mail from the domain “@renesas.com”.

Figure 20. Provide Email Address for New Registration

You will receive the registration link in an email, as shown in Figure 21.



Dear customer,

Thank you for registering with the KeyWrap service.
To start using this service, you need to click on the following URL to register your information.

https://dlm.renesas.com/keywrap/?menu=reg%2Fregist%2Ffixed%2F&param=id%3Duf2SL2o3Cd7ODijiEN%252BOmISX7qhfiv8H9mP%252F1gQ7aH8%253D

・Notes
Please register your information within three hours of receiving this e-mail.
The URL expires after the three hours, after which you will need to start the registration process anew.

Please delete this email if you were not aware that you were going to receive it.

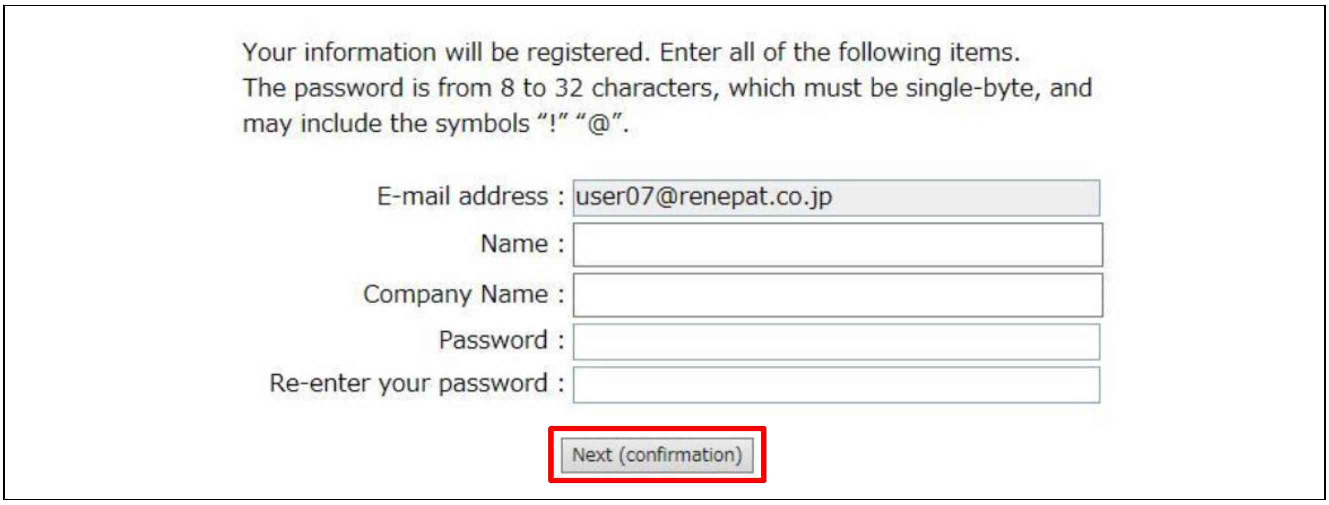
* This email was sent from a send-only address.
Please understand that there will be no response, even if you reply to this address.
* If you have forgotten your password, reset your password via the link "If you have forgotten your password ..." on the login page of this system.

Thank you.

Renesas Electronics Corporation

Figure 21. Email with Registration Link

Click on the URL in the confirmation email and provide the requested information. Click the **Next (confirmation)** button. After the confirmation screen is displayed, click on the **Registration** button to complete the user registration.



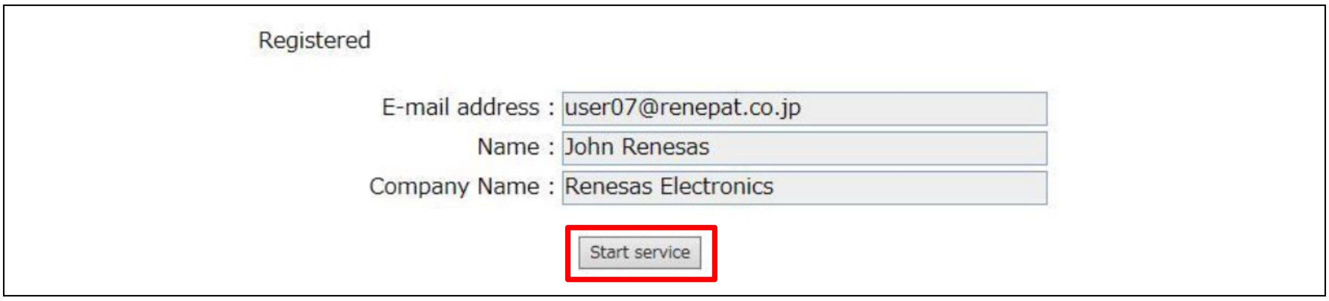
Your information will be registered. Enter all of the following items.
The password is from 8 to 32 characters, which must be single-byte, and may include the symbols "!" "@".

E-mail address : user07@renepat.co.jp
Name :
Company Name :
Password :
Re-enter your password :

Figure 22. Register Customer Information

4.3.4 PGP Public Key Exchange

Once you have successfully registered, the following screen will open. Click on the **Start service** button to start using the key encryption system.

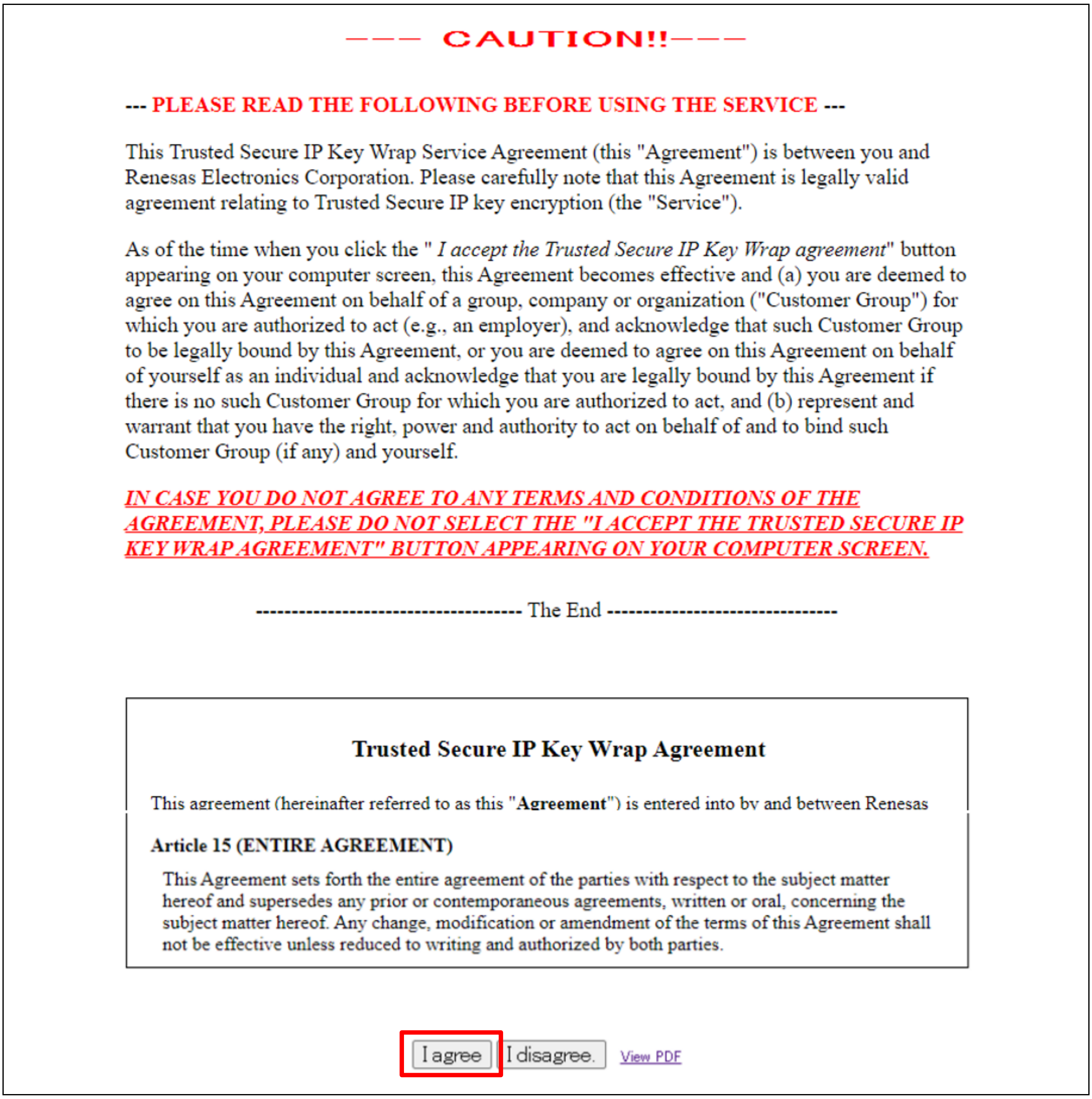


Registered

E-mail address : user07@renepat.co.jp
Name : John Renesas
Company Name : Renesas Electronics

Figure 23. Start Using the DLM Service

Accept the **Trusted Secure IP Key Wrap Agreement** as shown below by clicking **Agree**. Note that this Agreement will come up every time you log into the DLM server.



--- CAUTION!! ---

--- PLEASE READ THE FOLLOWING BEFORE USING THE SERVICE ---

This Trusted Secure IP Key Wrap Service Agreement (this "Agreement") is between you and Renesas Electronics Corporation. Please carefully note that this Agreement is legally valid agreement relating to Trusted Secure IP key encryption (the "Service").

As of the time when you click the " *I accept the Trusted Secure IP Key Wrap agreement*" button appearing on your computer screen, this Agreement becomes effective and (a) you are deemed to agree on this Agreement on behalf of a group, company or organization ("Customer Group") for which you are authorized to act (e.g., an employer), and acknowledge that such Customer Group to be legally bound by this Agreement, or you are deemed to agree on this Agreement on behalf of yourself as an individual and acknowledge that you are legally bound by this Agreement if there is no such Customer Group for which you are authorized to act, and (b) represent and warrant that you have the right, power and authority to act on behalf of and to bind such Customer Group (if any) and yourself.

IN CASE YOU DO NOT AGREE TO ANY TERMS AND CONDITIONS OF THE AGREEMENT, PLEASE DO NOT SELECT THE "I ACCEPT THE TRUSTED SECURE IP KEY WRAP AGREEMENT" BUTTON APPEARING ON YOUR COMPUTER SCREEN.

------------------------------------ The End -------------------------------

Trusted Secure IP Key Wrap Agreement

This agreement (hereinafter referred to as this "**Agreement**") is entered into by and between Renesas

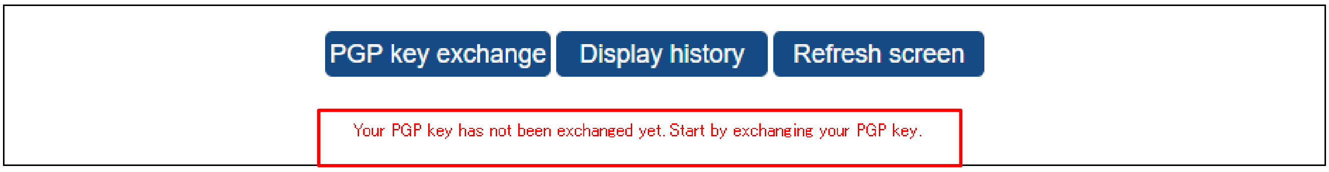
Article 15 (ENTIRE AGREEMENT)

This Agreement sets forth the entire agreement of the parties with respect to the subject matter hereof and supersedes any prior or contemporaneous agreements, written or oral, concerning the subject matter hereof. Any change, modification or amendment of the terms of this Agreement shall not be effective unless reduced to writing and authorized by both parties.

Figure 24. Agree with the Key Wrap Agreement

Communication between the user and the DLM server uses PGP to encrypt all transferred data. Public PGP keys must be exchanged to enable these transfers.

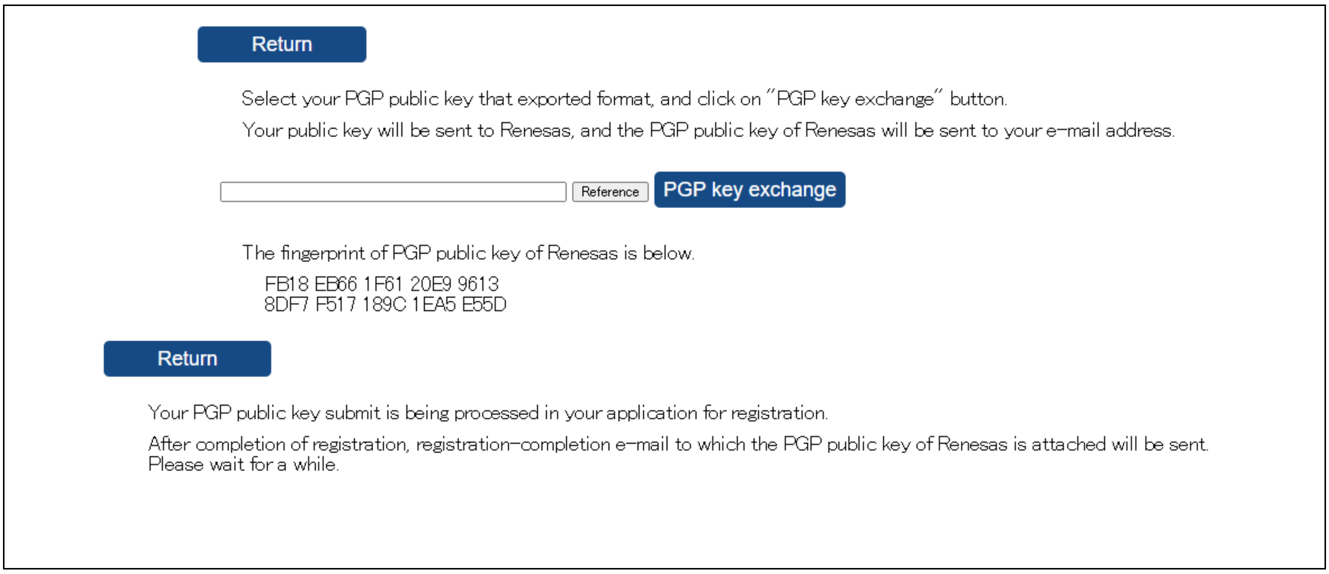
If you try to transfer data to the key wrap service without exchanging PGP keys, you will receive the error message shown below.



Your PGP key has not been exchanged yet. Start by exchanging your PGP key.

Figure 25. Request for PGP Key Exchange

To exchange PGP keys, click the **PGP key exchange** button. The dialog shown in Figure 26 will open. Click **Reference** and select the public key that you created and exported earlier (section 4.3.2, in Figure 19). Next, click **PGP key exchange** and wait to receive the Renesas PGP public key at the email address you provided during registration.



Select your PGP public key that exported format, and click on "PGP key exchange" button.
Your public key will be sent to Renesas, and the PGP public key of Renesas will be sent to your e-mail address.

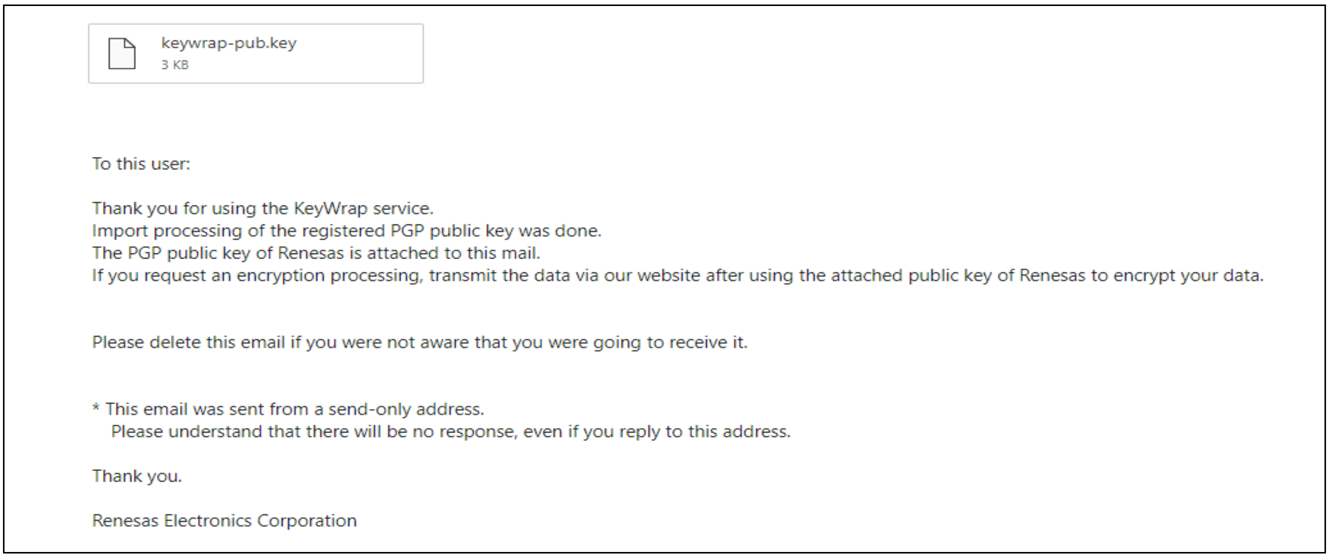
The fingerprint of PGP public key of Renesas is below.
FB18 EB66 1F61 20E9 9613
8DF7 F517 189C 1EA5 E55D

Your PGP public key submit is being processed in your application for registration.
After completion of registration, registration-completion e-mail to which the PGP public key of Renesas is attached will be sent.
Please wait for a while.

Figure 26. Provide PGP Public Key to DLM Server

You will receive an email with the content as shown in Figure 27, including the Renesas public key. Save the Renesas PGP public key received (`keywrap-pub.key`). Note that the fingerprint of this key is provided on the PGP key exchange dialog, as shown in Figure 26.

PGP keys can be exchanged any number of times. If keys are exchanged multiple times, all previous PGP keys are discarded, and the key wrap service will use the latest PGP public key that has been successfully exchanged to encrypt the transferred data.



To this user:

Thank you for using the KeyWrap service.
Import processing of the registered PGP public key was done.
The PGP public key of Renesas is attached to this mail.
If you request an encryption processing, transmit the data via our website after using the attached public key of Renesas to encrypt your data.

Please delete this email if you were not aware that you were going to receive it.

* This email was sent from a send-only address.
Please understand that there will be no response, even if you reply to this address.

Thank you.

Renesas Electronics Corporation

Figure 27. Receive the Renesas PGP Public Key

4.3.5 Import Renesas PGP Public Key into Kleopatra

Go back to the Kleopatra application and import the Renesas PGP Public key into Kleopatra as shown in Figure 28.

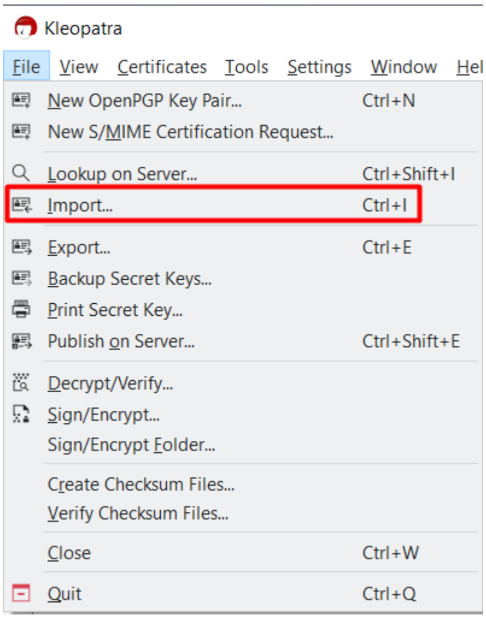
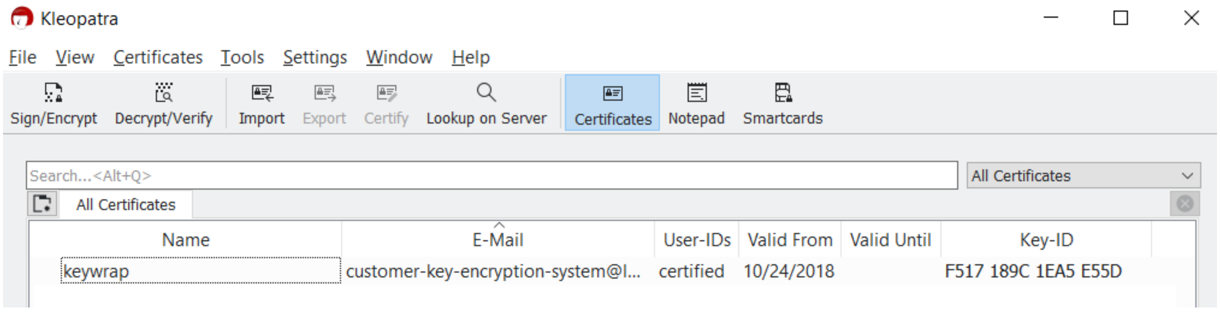


Figure 28. Import Renesas Public Key into Kleopatra

Click **File->Import**, and select the `keywrap-pub.key` file, received from the key wrap service.

The following item will appear in the **Imported Certificates** in Kleopatra, and the Renesas public key is ready to be used.



Name	E-Mail	User-IDs	Valid From	Valid Until	Key-ID
keywrap	customer-key-encryption-system@l...	certified	10/24/2018		F517 189C 1EA5 E55D

Figure 29. Renesas PGP Public Key

4.4 Create the UFPK and W-UFPK

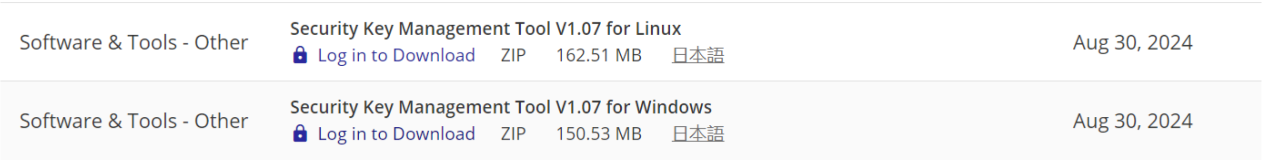
This section walks the user through the process of creating a UFPK and obtaining the W-UFPK.

4.4.1 Renesas Security Key Management Tool

The Renesas Security Key Management Tool (SKMT) performs several functions during the DLM key injection process. The SKMT is available from the following link:

https://www.renesas.com/software-tool/security-key-management-tool

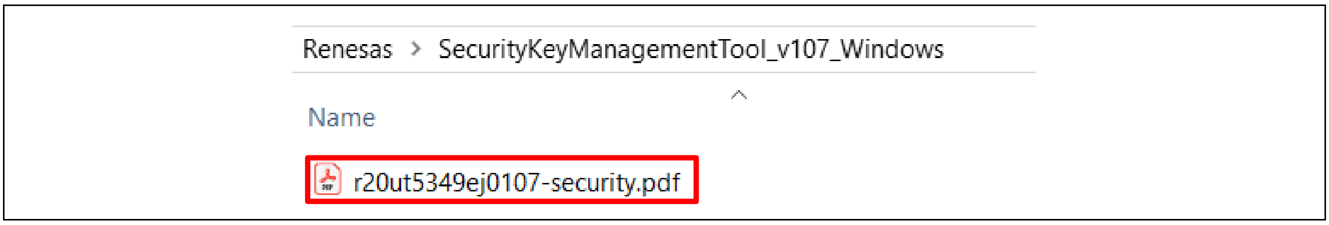
Locate the **Downloads** area, download the latest Security Key Management Tool installer, and install it as required for your operating environment. This tool supports Windows and Linux. The screenshots and instructions in this document use the Windows environment.



Software & Tools - Other	Security Key Management Tool V1.07 for Linux Log in to Download ZIP 162.51 MB 日本語	Aug 30, 2024
Software & Tools - Other	Security Key Management Tool V1.07 for Windows Log in to Download ZIP 150.53 MB 日本語	Aug 30, 2024

Figure 30. Download the Security Key Management Tool for Windows or Linux

The User's Manual of this tool is located in the extract folder. We recommend that you read through the User's Manual before proceeding to the following section.



Renesas > SecurityKeyManagementTool_v107_Windows

Name

r20ut5349ej0107-security.pdf

Figure 31. User's Manual for the Security Key Management Tool

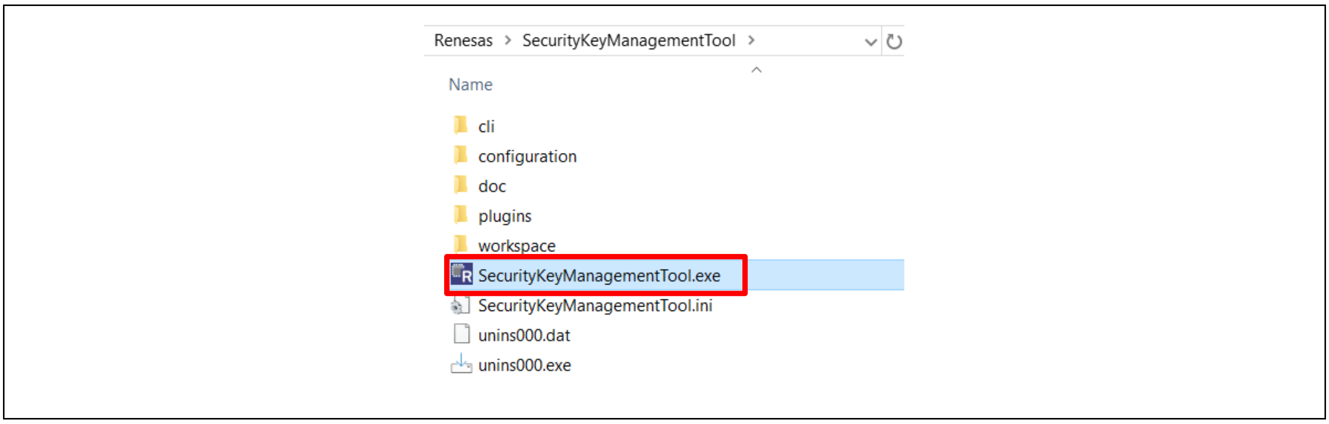
The SKMT provides two interfaces to users: a Command Line Interface (CLI) and a Graphical User Interface (GUI). The GUI interface is primarily intended for development usage. The CLI interface is primarily intended for production support or development efforts involving multiple keys due to its ability to create key file-generation scripts. This application note will explain how to use both interfaces to perform DLM key injection.

4.4.2 Create a UFPK Key File

The Security Key Management Tool (SKMT) can be used in either Graphic User Interface (GUI) or Command Line Interface (CLI) mode to create a UFPK key file.

4.4.2.1 Generate UFPK Using the GUI

Launch the **Security Key Management Tool** executable.



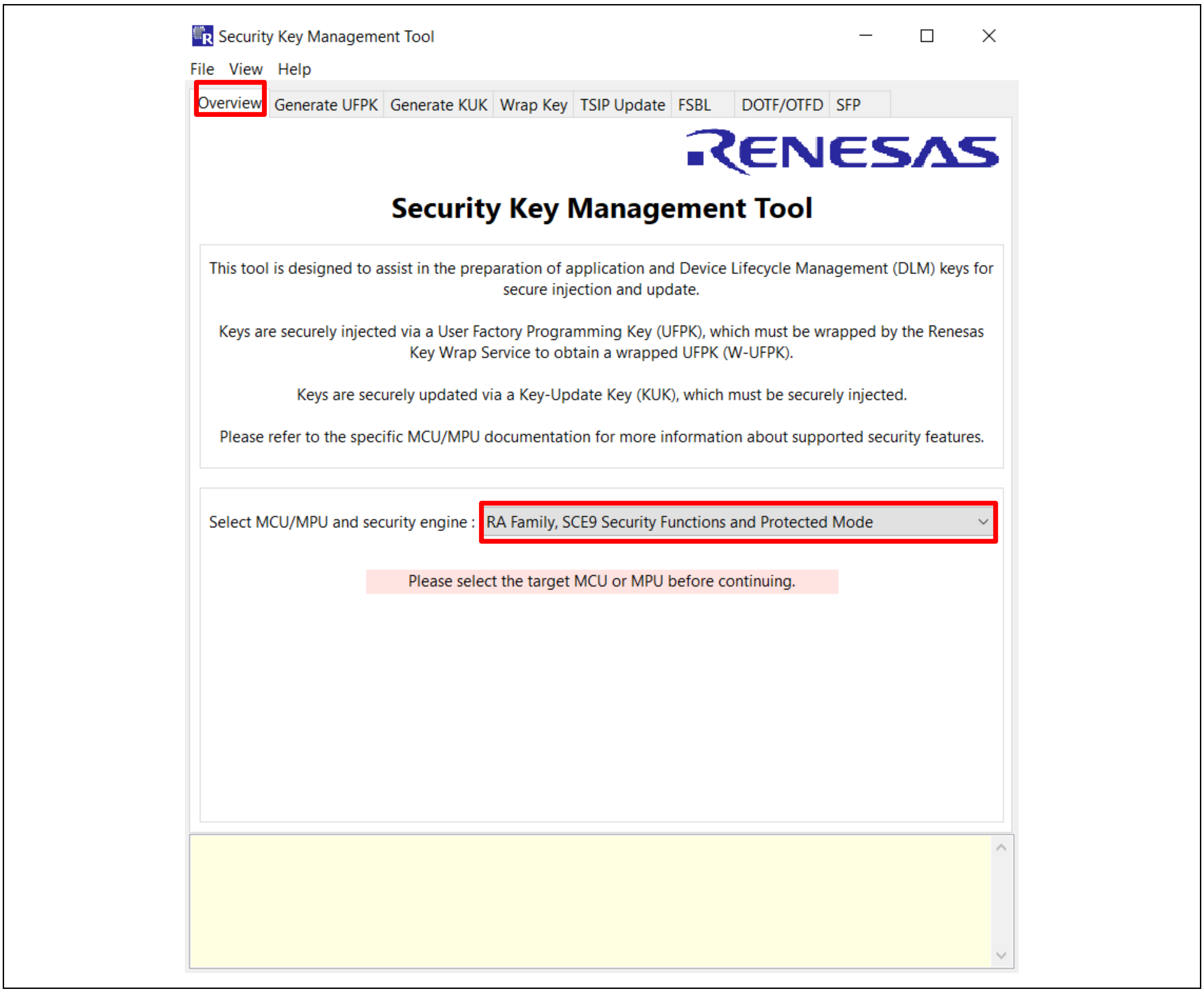
Renesas > SecurityKeyManagementTool >

Name

- cli
- configuration
- doc
- plugins
- workspace
- SecurityKeyManagementTool.exe
- SecurityKeyManagementTool.ini
- unins000.dat
- unins000.exe

Figure 32. Launch SKMT GUI Interface

In the **Overview** window, select the microcontroller/microprocessor and crypto engine as **RA Family, SCE9 Security Functions and Protected Mode**.


Security Key Management Tool
File View Help
Overview Generate UFPK Generate KUK Wrap Key TSIP Update FSBL DOTF/OTFD SFP
RENESAS

Security Key Management Tool

This tool is designed to assist in the preparation of application and Device Lifecycle Management (DLM) keys for secure injection and update.

Keys are securely injected via a User Factory Programming Key (UFPK), which must be wrapped by the Renesas Key Wrap Service to obtain a wrapped UFPK (W-UFPK).

Keys are securely updated via a Key-Update Key (KUK), which must be securely injected.

Please refer to the specific MCU/MPU documentation for more information about supported security features.

Select MCU/MPU and security engine : RA Family, SCE9 Security Functions and Protected Mode

Please select the target MCU or MPU before continuing.

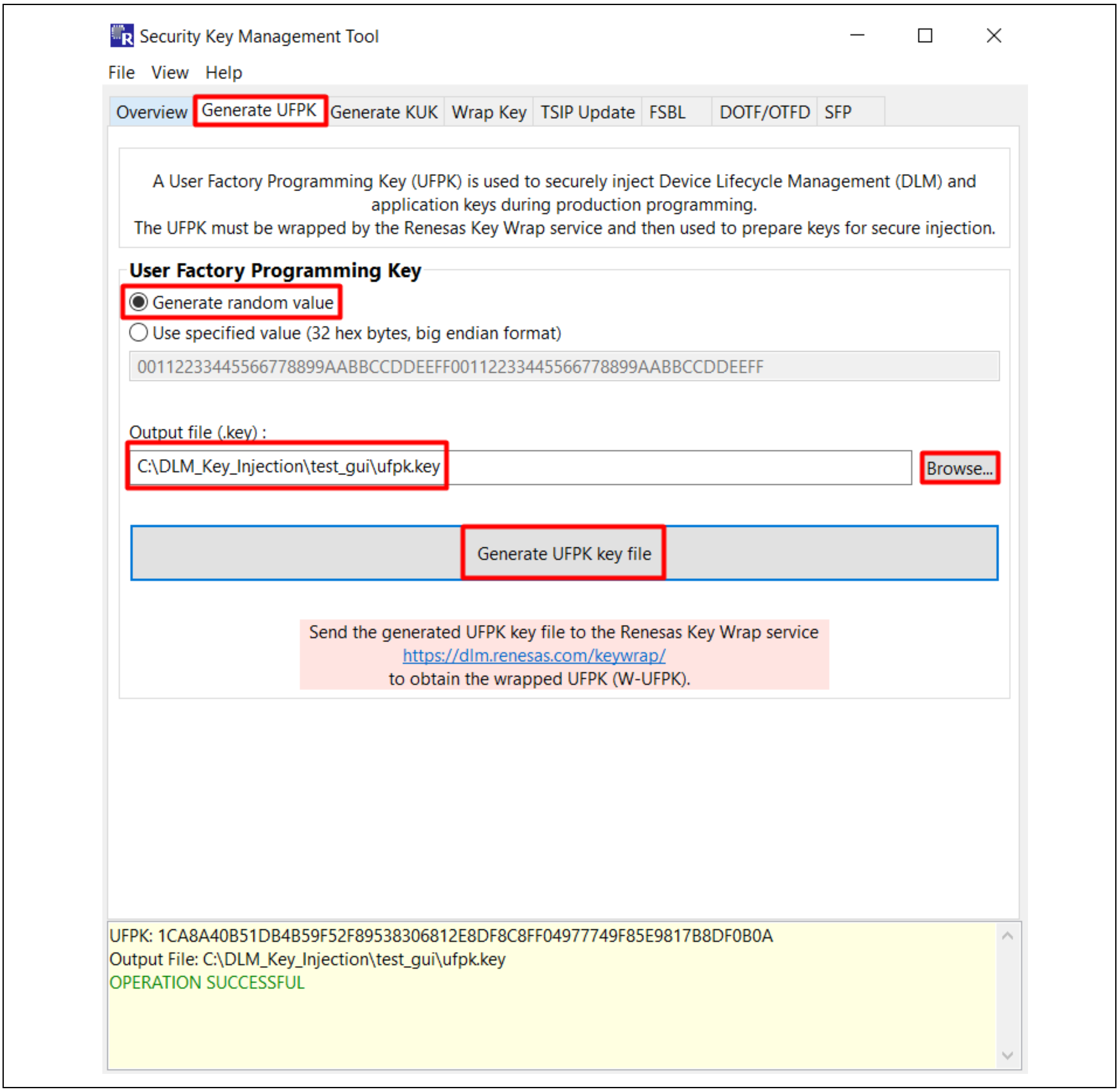
Figure 33. Select RA Family, SCE9 Protected Mode

Next, click on the **Generate UFPK** tab.

As shown in Figure 34 and Figure 35, you can specify a desired value for the UFPK, or the tool can create a random value. Figure 34 shows the tool generating a random value by selecting **Generate random value**. Figure 35 shows specifying a specific value by selecting **User specified value** and entering the 32-byte key value in big-endian format.

Click the **Browse** button to enter a file name for the generated key file. Here we have chosen `ufpk.key`.

Finally, click the button **Generate UFPK key file** to generate the key file `ufpk.key`.



A User Factory Programming Key (UFPK) is used to securely inject Device Lifecycle Management (DLM) and application keys during production programming.
The UFPK must be wrapped by the Renesas Key Wrap service and then used to prepare keys for secure injection.

User Factory Programming Key

● Generate random value

○ Use specified value (32 hex bytes, big endian format)

00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF

Output file (.key) : C:\DLM_Key_Injection\test_gui\ufpk.key

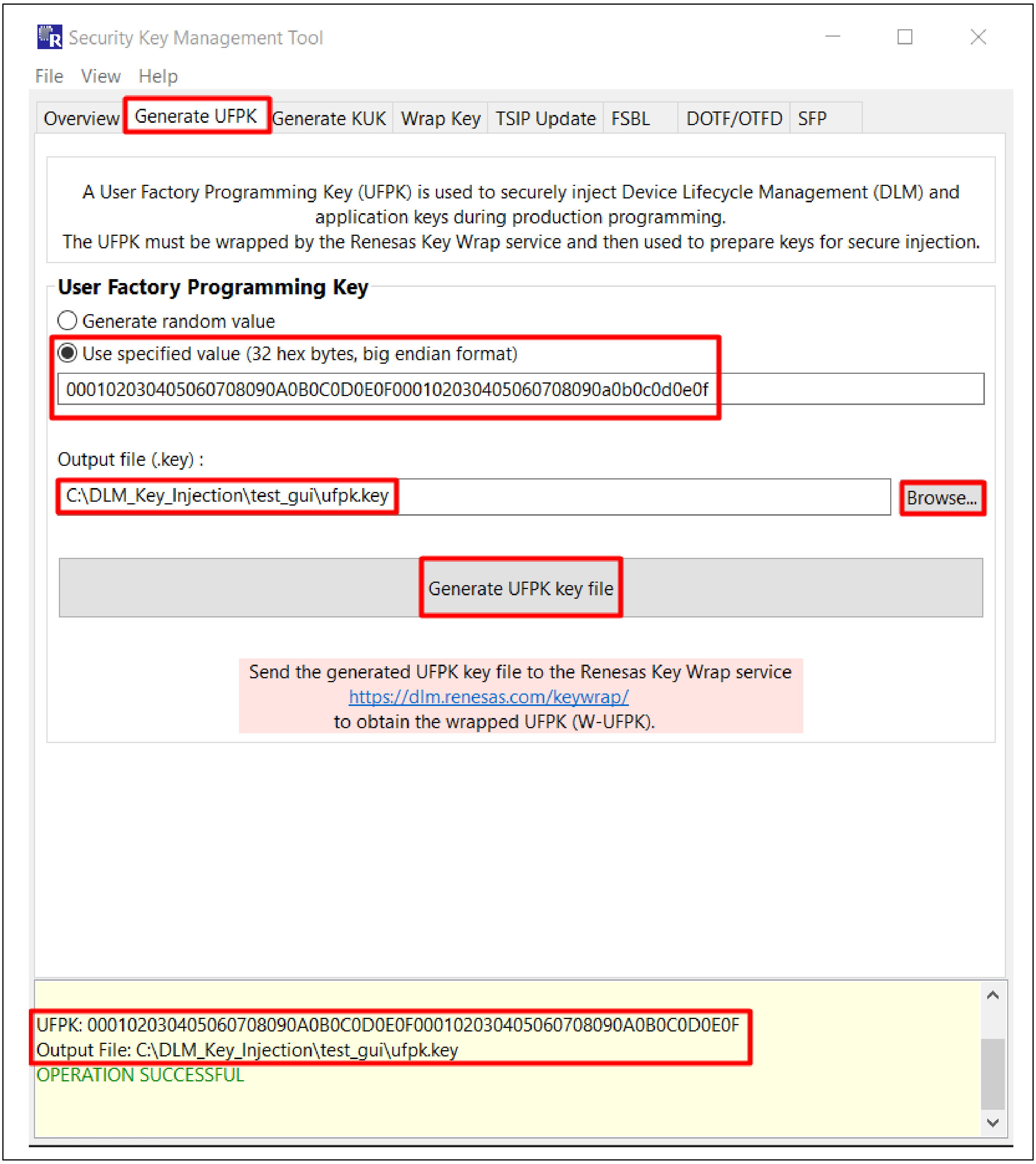
Send the generated UFPK key file to the Renesas Key Wrap service
https://dlm.renesas.com/keywrap/
to obtain the wrapped UFPK (W-UFPK).

```
UFPK: 1CA8A40B51DB4B59F52F89538306812E8DF8C8FF04977749F85E9817B8DF0B0A
Output File: C:\DLM_Key_Injection\test_gui\ufpk.key
OPERATION SUCCESSFUL
```

Figure 34. Generate a Random UFPK Using the GUI

In this example, the following 32-byte key is used:

```
000102030405060708090A0B0C0D0E0F000102030405060708090a0b0c0d0e0f
```



A User Factory Programming Key (UFPK) is used to securely inject Device Lifecycle Management (DLM) and application keys during production programming.
The UFPK must be wrapped by the Renesas Key Wrap service and then used to prepare keys for secure injection.

User Factory Programming Key

○ Generate random value

● Use specified value (32 hex bytes, big endian format)

000102030405060708090A0B0C0D0E0F000102030405060708090a0b0c0d0e0f

Output file (.key) : C:\DLM_Key_Injection\test_gui\ufpk.key

Send the generated UFPK key file to the Renesas Key Wrap service
https://dlm.renesas.com/keywrap/
to obtain the wrapped UFPK (W-UFPK).

```
UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F
Output File: C:\DLM_Key_Injection\test_gui\ufpk.key
OPERATION SUCCESSFUL
```

Figure 35. Create a Fixed UFPK Using the GUI

4.4.2.2 Generate UFPK Using the CLI

Open a Command Prompt window and navigate to the folder where `skmt.exe` resides, typically under `\Renesas\Security Key Management Tool\CLI\`.

Use the following command to generate a random UFPK and place it in a key file (`ufpk.key`). If desired, a complete file name with a path may be specified. Refer to the Security Key Management Tool User’s Manual for more details about the `/genufpk` command.

```
skmt.exe /genufpk /output "C:\DLM_Key_Injection\keys\ufpk.key"
```

This command will generate a random 256-bit UFPK, as shown below.

```
UFPK: E8AB23E99C9AD42823DA4215549A41496720F7243680A4715F4B944ACC94B691
Output File: C:\DLM_Key_ Injection\test\ufpk.key
```

Figure 36. Generate a Random UFPK Using the CLI

It is also possible to specify a specific UFPK, as shown by the following command:

```
skmt.exe /genufpk /ufpk
"000102030405060708090A0B0C0D0E0F000102030405060708090a0b0c0d0e0f" /output
"C:\DLM_Key_Injection\test\ufpk.key"
```

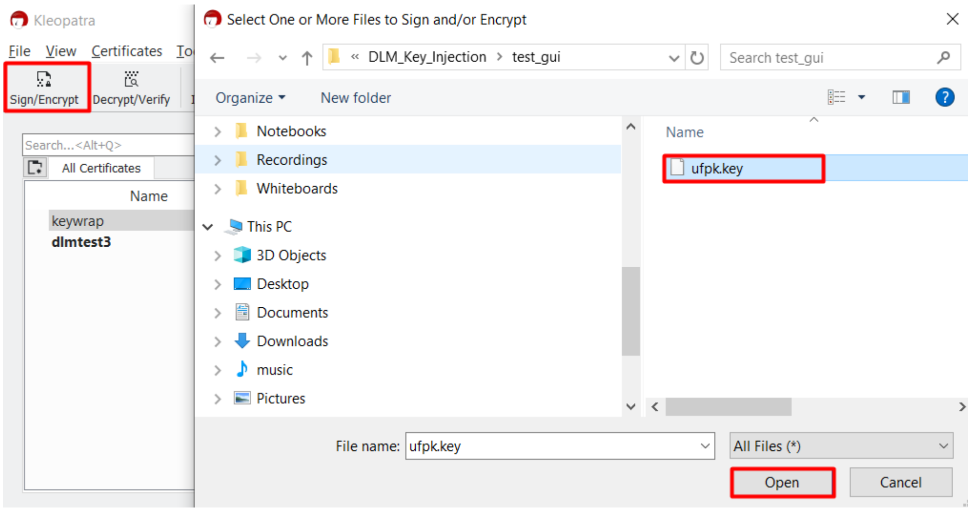
```
UFPK: 000102030405060708090A0A0C0D0E0F000102030405060708090a0b0c0d0e0f
Output File: C:\DLM_Key_Injection\test\ufpk.key
```

Figure 37. Create a Fixed UFPK Using the CLI

Note: The `ufpk.key` generated from any one of the previous executions can be used for the purpose of exercising the example project.

4.4.3 Encrypt UFPK with Renesas PGP Public Key

Select **Sign/Encrypt** from Kleopatra and select to encrypt the `ufpk.key` key file with the Renesas PGP public key.

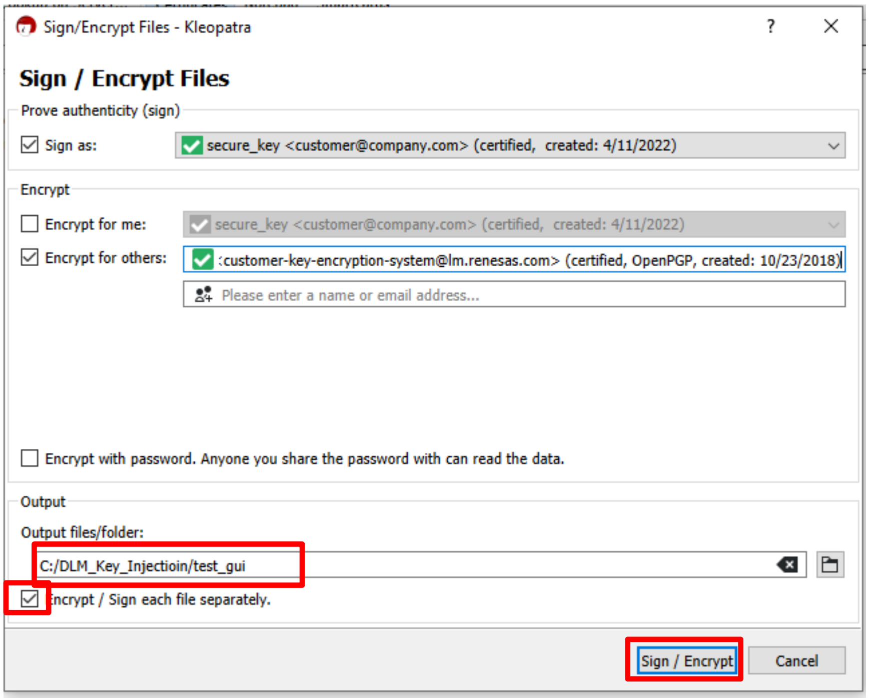


File name: ufpk.key

All Files (*)

Figure 38. Select the UFPK Key to be Encrypted

Choose **Encrypt for others** and select the Renesas PGP Public key**.** Click **Sign/Encrypt**.



Sign/Encrypt Files - Kleopatra

Sign / Encrypt Files

Prove authenticity (sign)

☑ Sign as: secure_key <customer@company.com> (certified, created: 4/11/2022)

Encrypt

□ Encrypt for me: secure_key <customer@company.com> (certified, created: 4/11/2022)

☑ Encrypt for others: :customer-key-encryption-system@lm.renesas.com> (certified, OpenPGP, created: 10/23/2018)

Please enter a name or email address...

□ Encrypt with password. Anyone you share the password with can read the data.

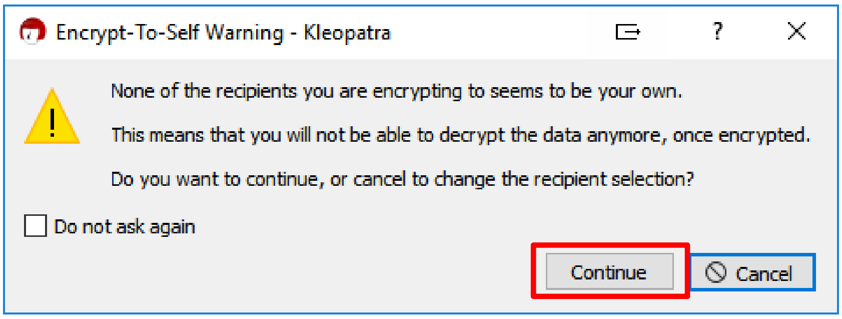
Output

Output files/folder: C:/DLM_Key_Injectioin/test_gui

☑ Encrypt / Sign each file separately.

Figure 39. Use Renesas Public Key to Encrypt UFPK

You may also want to select **Encrypt for me** so you have an encrypted version of the UFPK key file to archive. If you do not, you will get an Encrypt-To-Self Warning that you cannot decrypt the data. Press **Continue**.



Encrypt-To-Self Warning - Kleopatra

None of the recipients you are encrypting to seems to be your own.

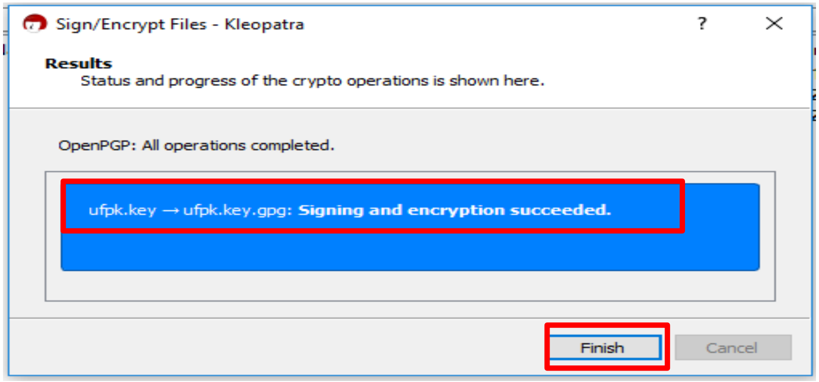
This means that you will not be able to decrypt the data anymore, once encrypted.

Do you want to continue, or cancel to change the recipient selection?

□ Do not ask again

Figure 40. Confirm Encryption Option

The UFPK encrypted with the Renesas public key will be generated in the selected folder, and .gpg will be added to the extension. In this example, `ufpk.key.gpg` is generated. Click **Finish**.



Sign/Encrypt Files - Kleopatra

Results

Status and progress of the crypto operations is shown here.

OpenPGP: All operations completed.

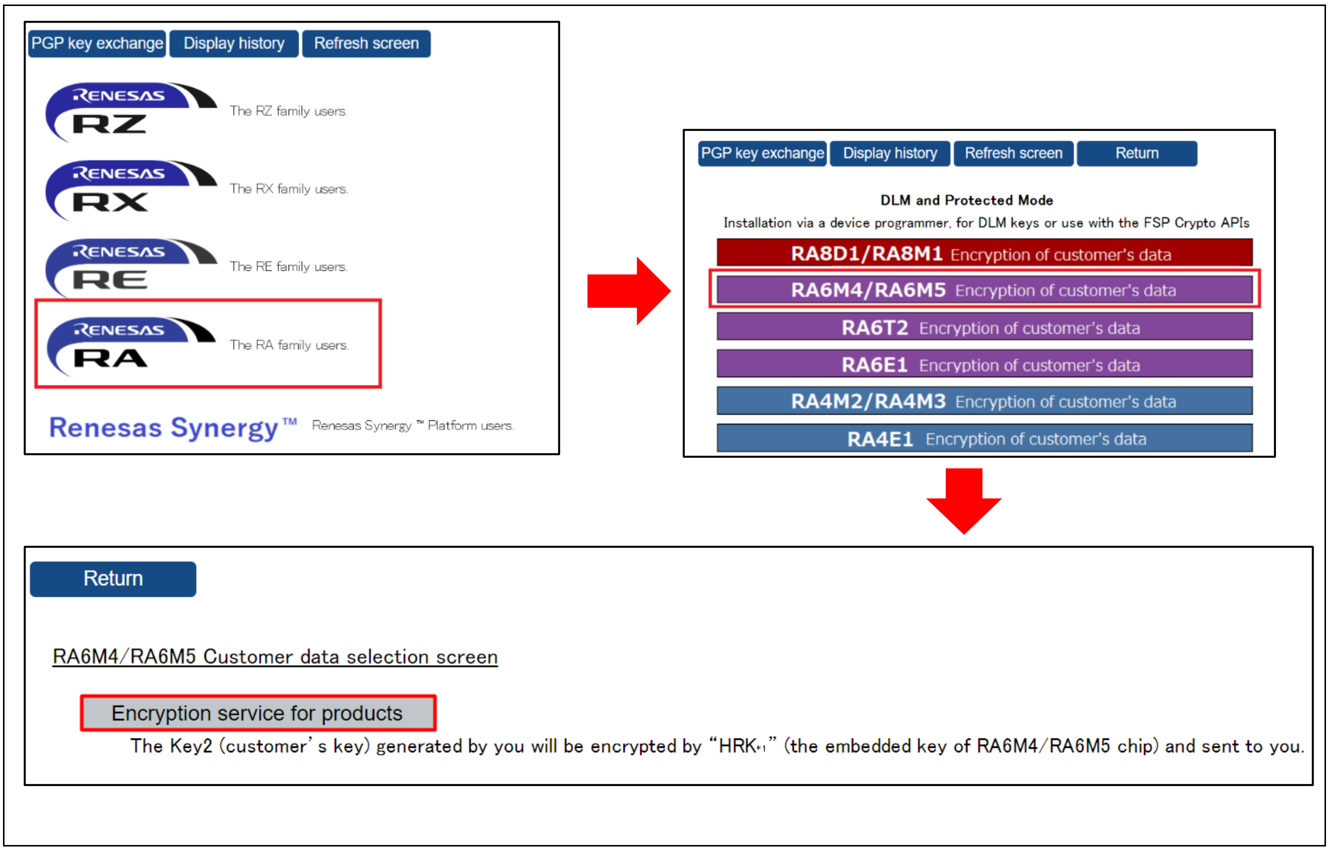
ufpk.key → ufpk.key.gpg: **Signing and encryption succeeded.**

Figure 41. Encrypt the UFPK Key with Renesas Public Key

4.4.4 Send UFPK to Renesas DLM Server

Now, we can send the encrypted UFPK to the Renesas DLM Server, where it will be decrypted by the Renesas private key and used to generate the Wrapped UFPK (W-UFPK).

From the DLM Server user interface, select the RA Family series and choose **RA6M4/RA6M5 Encryption of customer's data** -> **Encryption service for products,** as shown below.


PGP key exchange
Display history
Refresh screen
RENESAS RZ
The RZ family users.
RENESAS RX
The RX family users.
RENESAS RE
The RE family users.
RENESAS RA
The RA family users.
Renesas Synergy™
Renesas Synergy ™ Platform users.
PGP key exchange
Display history
Refresh screen
Return
RA8D1/RA8M1 Encryption of customer's data
RA6M4/RA6M5 Encryption of customer's data
RA6T2 Encryption of customer's data
RA6E1 Encryption of customer's data
RA4M2/RA4M3 Encryption of customer's data
RA4E1 Encryption of customer's data
Return
Encryption service for products

DLM and Protected Mode

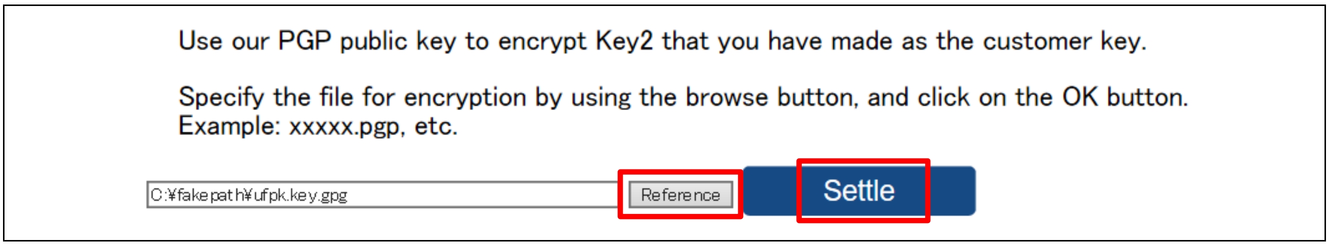
Installation via a device programmer, for DLM keys or use with the FSP Crypto APIs

RA6M4/RA6M5 Customer data selection screen

The Key2 (customer' s key) generated by you will be encrypted by "HRK*1" (the embedded key of RA6M4/RA6M5 chip) and sent to you.

Figure 42. Select RA Device

Next, click **Reference,** and select the `.gpg` file generated from Figure 41.



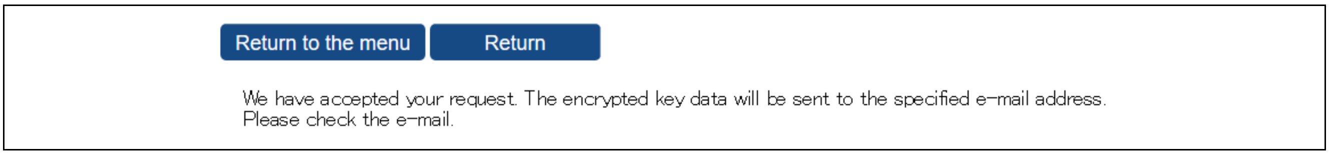
Use our PGP public key to encrypt Key2 that you have made as the customer key.

Specify the file for encryption by using the browse button, and click on the OK button.
Example: xxxxx.pgp, etc.

C:¥fakepath¥ufpk.key.gpg

Figure 43. Send Encrypted UFPK to DLM Server

Click **Settle.** The DLM server will display a message that the file has been received, and the W-UFPK will be sent to the registered email address.

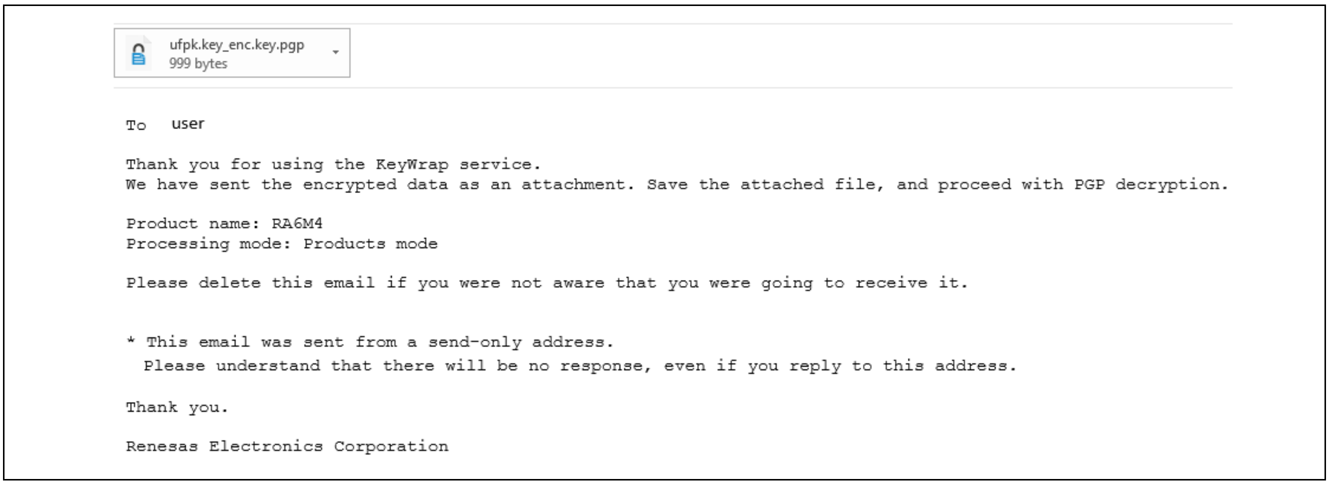


We have accepted your request. The encrypted key data will be sent to the specified e-mail address.
Please check the e-mail.

Figure 44. Message from DLM Sever

4.4.5 Receive the Encrypted W-UFPK Key

The W-UFPK, encrypted with your PGP public key, should arrive in your email within a few minutes.



ufpk.key_enc.key.pgp
999 bytes

```
To   user

Thank you for using the KeyWrap service.
We have sent the encrypted data as an attachment. Save the attached file, and proceed with PGP decryption.

Product name: RA6M4
Processing mode: Products mode

Please delete this email if you were not aware that you were going to receive it.


* This email was sent from a send-only address.
  Please understand that there will be no response, even if you reply to this address.

Thank you.

Renesas Electronics Corporation
```

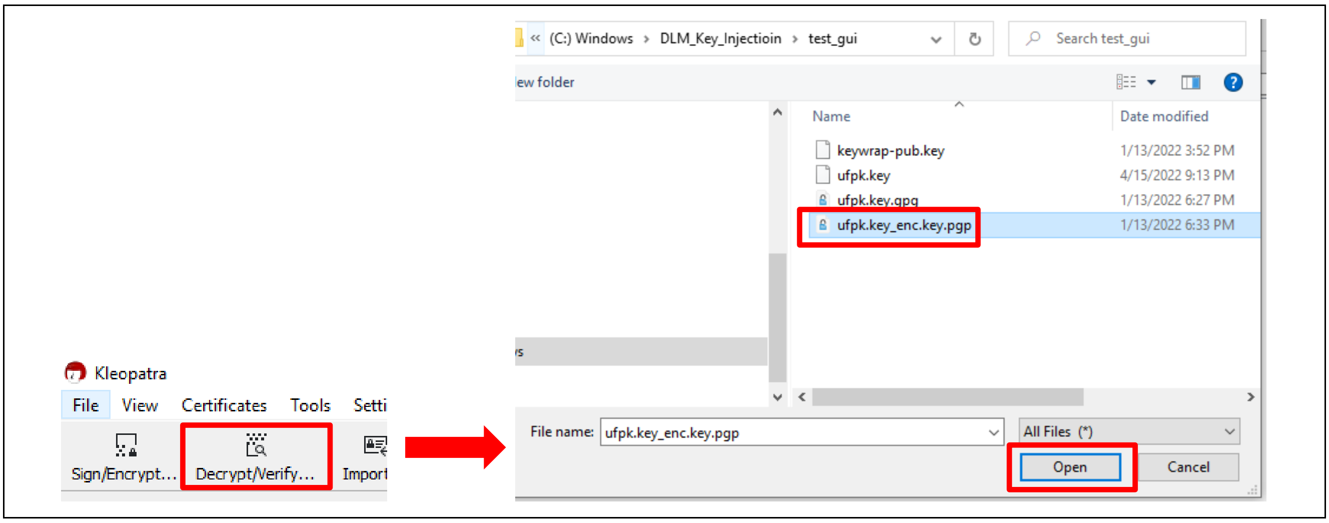
Figure 45. Receive the Wrapped DLM Key

Save this file for use in next step.

4.4.6 Decrypt the Encrypted W-UFPK Key

The W-UFPK received from the email is encrypted with your public key. You must decrypt the W-UFPK key file using your private key.

In Kleopatra, click **Decrypt/Verify** and select the encrypted W-UFPK key file.

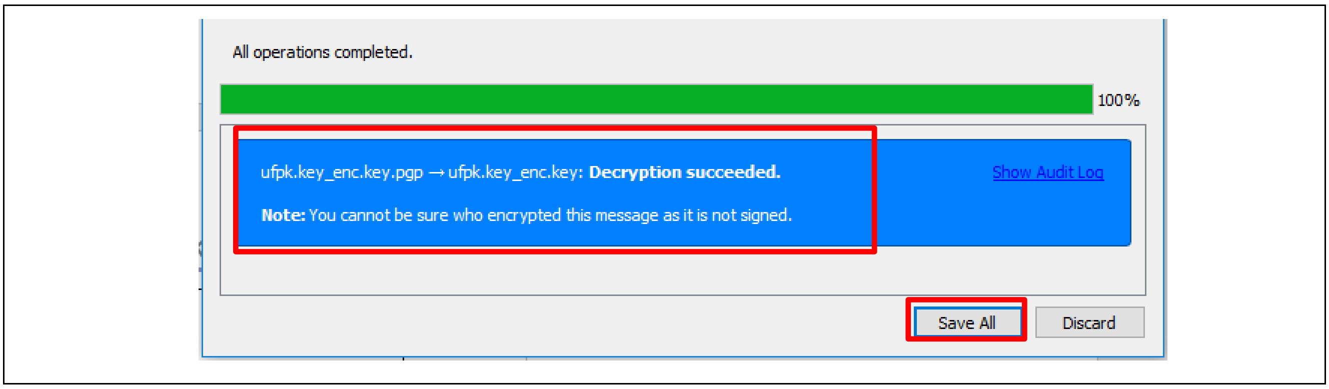


Name	Date modified
keywrap-pub.key	1/13/2022 3:52 PM
ufpk.key	4/15/2022 9:13 PM
ufpk.key.gpg	1/13/2022 6:27 PM
ufpk.key_enc.key.pgp	1/13/2022 6:33 PM

File name: ufpk.key_enc.key.pgp
All Files (*)

Figure 46. Encrypted W-UFPK Key File

If prompted, provide your PGP key passphrase. The decrypted file will be stored at the specified location.



All operations completed.

100%

ufpk.key_enc.key.pgp → ufpk.key_enc.key: **Decryption succeeded.**

Note: You cannot be sure who encrypted this message as it is not signed.

Figure 47. Decrypt with PGP Private Key

4.5 Generate the DLM Key Injection File

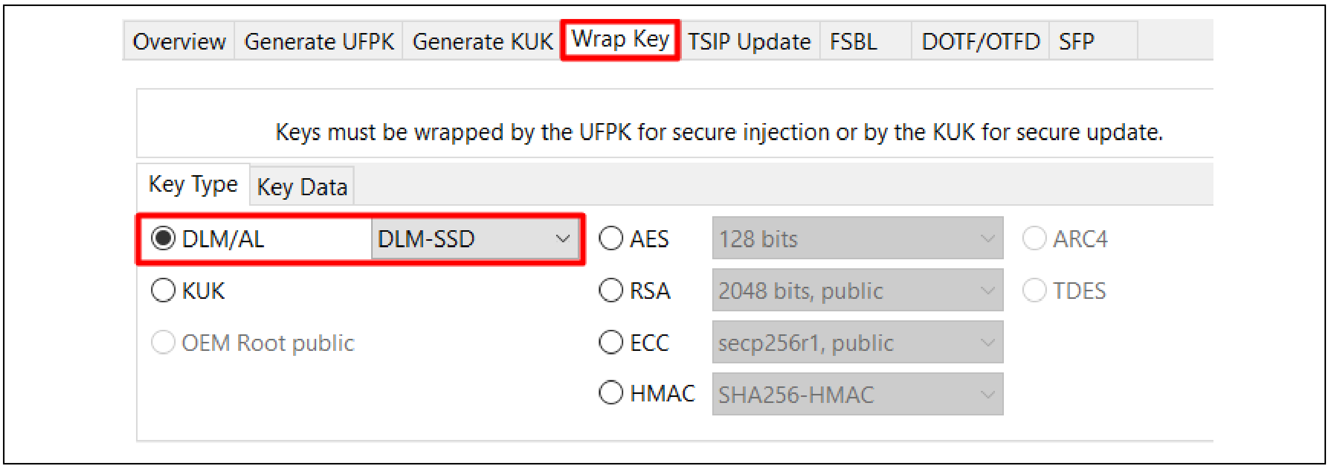
To inject a DLM key, the key needs to be wrapped by the UFPK, and both the wrapped DLM Key and the W-UFPK need to be sent to the MCU via the serial programming interface. The SKMT can create an RFP key injection file, which includes the wrapped DLM key and W-UFPK.

4.5.1 Wrap DLM Keys Using SKMT GUI

This section shows how to create key injection files for the SECDBG_KEY and NONSECDBG_KEY using the GUI interface.

4.5.1.1 Wrap a SECDBG_KEY

Launch the SKMT GUI and select the **Wrap Key** tab. Then open the **Key Type** tab and select **DLM**. First, we will inject the SECDBG_KEY, so select **DLM-SSD**.

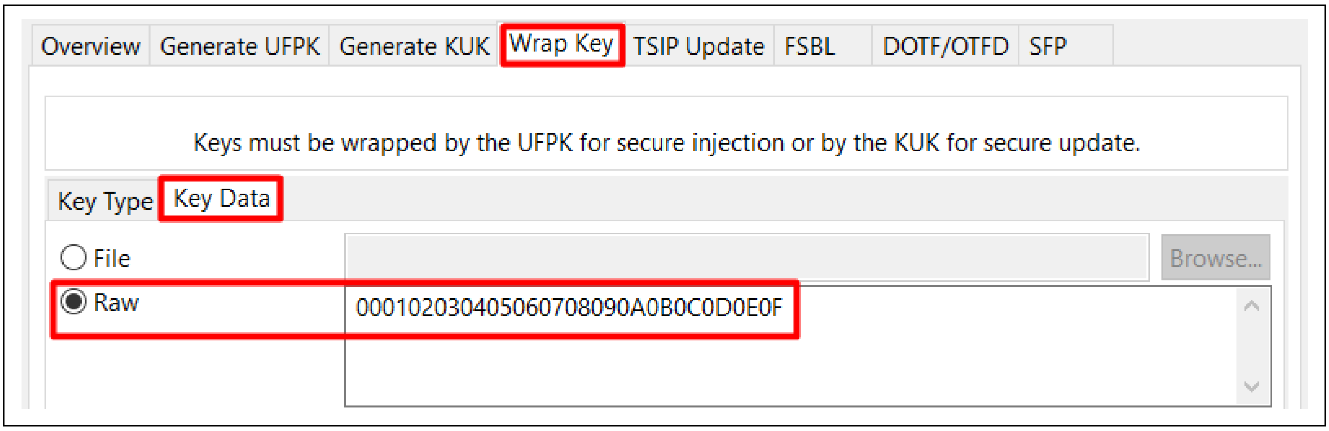


Keys must be wrapped by the UFPK for secure injection or by the KUK for secure update.

● DLM/AL DLM-SSD
○ KUK
○ OEM Root public
○ AES 128 bits
○ RSA 2048 bits, public
○ ECC secp256r1, public
○ HMAC SHA256-HMAC
○ ARC4
○ TDES

Figure 48. Choose DLM-SSD as the Key Type

Select the **Key Data** tab. For this example, we will specify the key as raw data by selecting **Raw** and entering the key value (`000102030405060708090A0B0C0D0E0F`), but we could also specify a binary key file containing the key value.



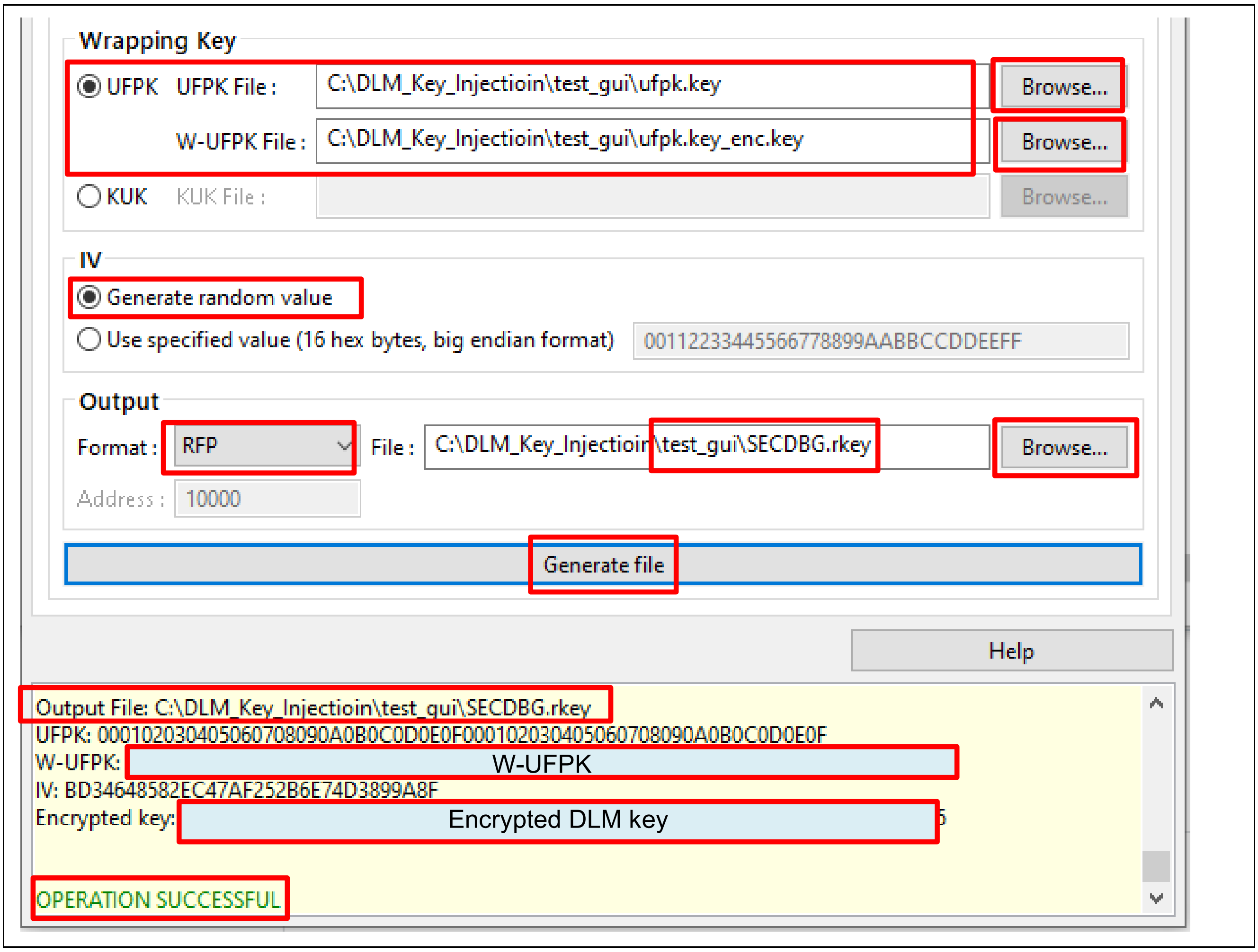
Keys must be wrapped by the UFPK for secure injection or by the KUK for secure update.

○ File
● Raw 000102030405060708090A0B0C0D0E0F

Figure 49. Enter the SECDBG_KEY Data

In the **Wrapping Key** section, click the corresponding **Browse** buttons to select the **UFPK** and **W-UFPK** key pair created in section 4.4.2 and 4.4.6. Choose **Generate random value** for the IV. In the **Output** section, select **RFP** and click the **Browse** button to enter the output file name.

Now click the **Generate File** button. The Renesas key file (`SECDBG.rkey`) will be generated.



Wrapping Key

● UFPK UFPK File : C:\DLM_Key_Injectioin\test_gui\ufpk.key

W-UFPK File : C:\DLM_Key_Injectioin\test_gui\ufpk.key_enc.key

○ KUK KUK File :

IV

● Generate random value

○ Use specified value (16 hex bytes, big endian format) 00112233445566778899AABBCCDDEEFF

Output

Format : RFP File : C:\DLM_Key_Injectioin\test_gui\SECDBG.rkey

Address : 10000

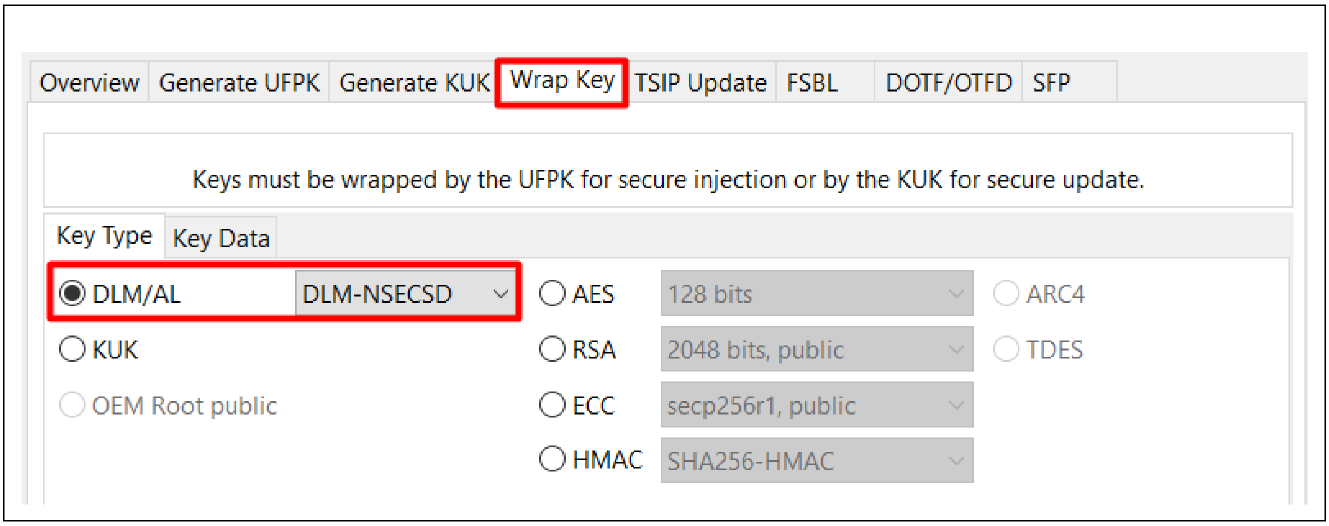
```
Output File: C:\DLM_Key_Injectioin\test_gui\SECDBG.rkey
UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F
W-UFPK: W-UFPK
IV: BD34648582EC47AF252B6E74D3899A8F
Encrypted key: Encrypted DLM key

OPERATION SUCCESSFUL
```

Figure 50. Generate the RFP Injection File for the SECDBG_KEY

The plaintext DLM key and UFPK are NOT contained in the `*.rkey` file, enabling confidential transfer of the key injection file.

4.5.1.2 Wrap a NONSECDBG_KEY

Select **DLM-NSECSD** in the **Key Type** tab.



Keys must be wrapped by the UFPK for secure injection or by the KUK for secure update.

● DLM/AL DLM-NSECSD

○ KUK

○ OEM Root public

○ AES 128 bits

○ RSA 2048 bits, public

○ ECC secp256r1, public

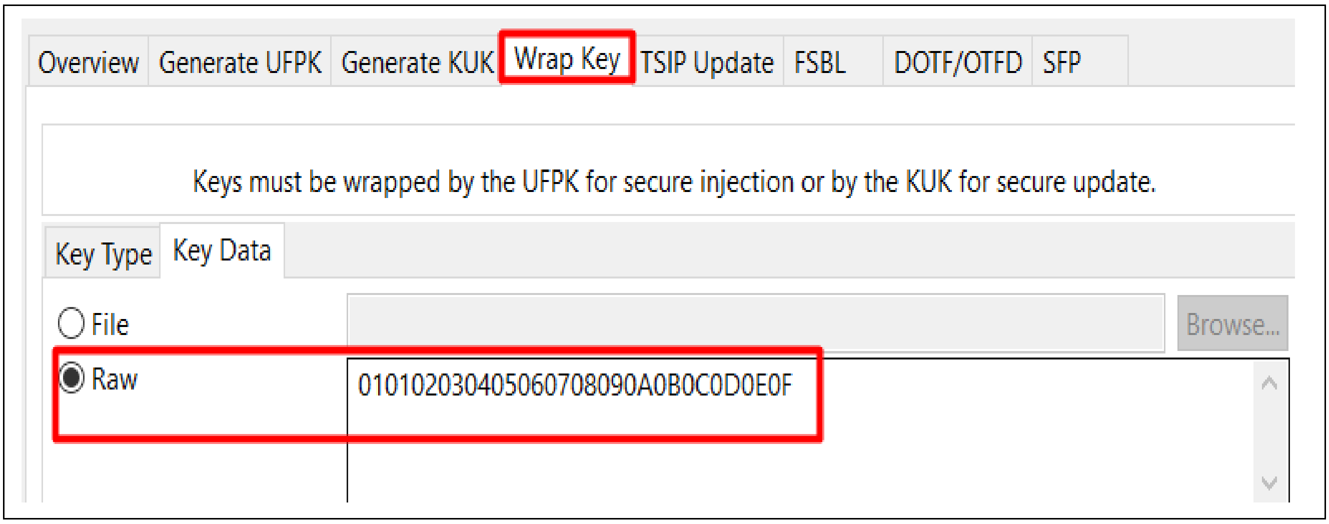
○ HMAC SHA256-HMAC

○ ARC4

○ TDES

Figure 51. Choose DLM-NSECSD as the Key Type

Enter the data for the key value on the **Key Data** tab. Again, for this example, we will specify raw data for the key (`0101020304050607080900A0B0C0D0E0F`).



Keys must be wrapped by the UFPK for secure injection or by the KUK for secure update.

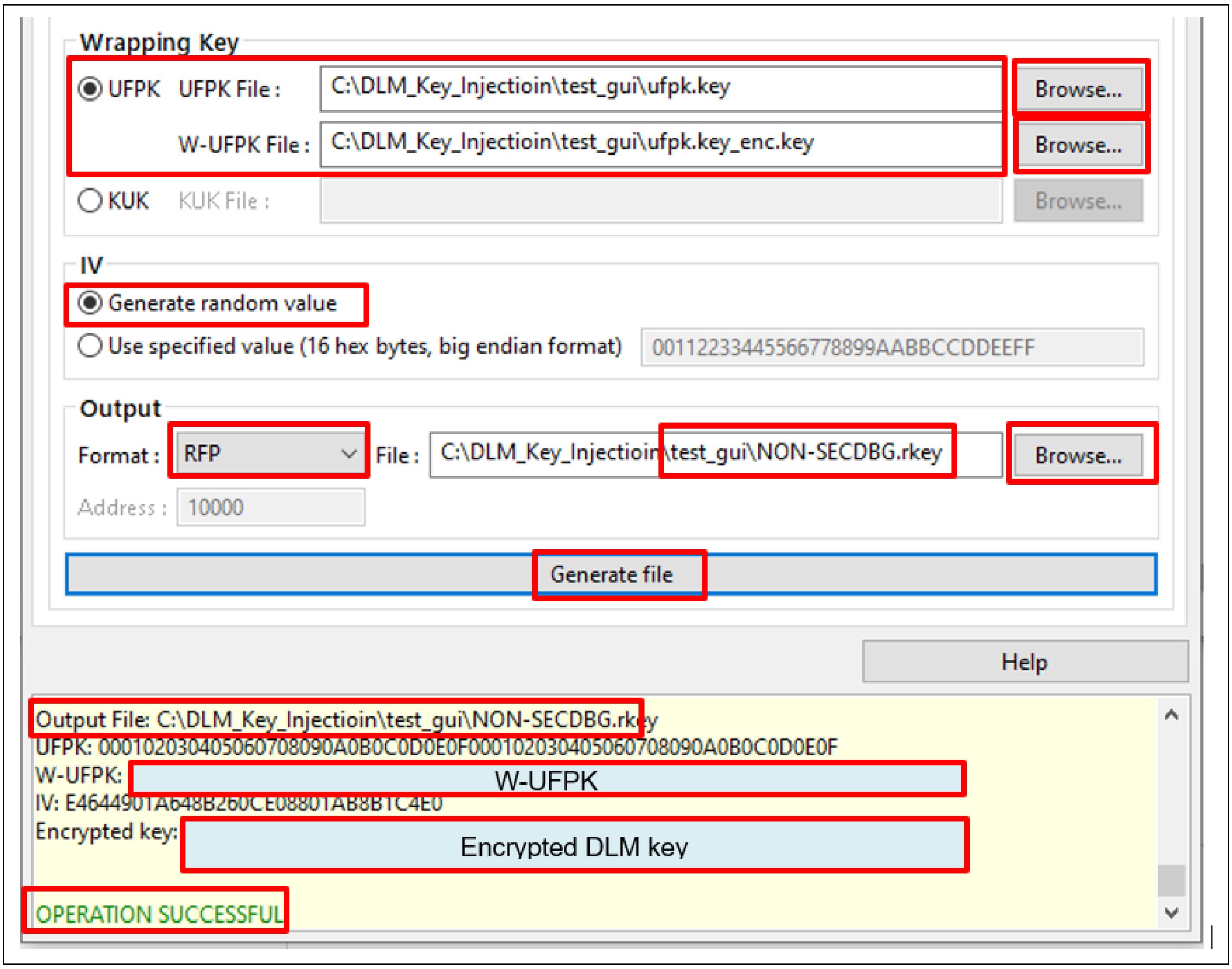
○ File

● Raw 010102030405060708090A0B0C0D0E0F

Figure 52. Enter the NONSECDBG_KEY Data

In the **Wrapping Key** section, click the corresponding **Browse** buttons to select the **UFPK** and **W-UFPK** key pair created in section 4.4.2 and 4.4.6. Choose **Generate random value** for the IV. In the **Output** section, select **RFP**, and click the **Browse** button to enter the output file name.

Now click the **Generate File** button. The Renesas key file (`NON-SECDBG.rkey)` will be generated.



Wrapping Key

● UFPK UFPK File : C:\DLM_Key_Injectioin\test_gui\ufpk.key

W-UFPK File : C:\DLM_Key_Injectioin\test_gui\ufpk.key_enc.key

○ KUK KUK File :

IV

● Generate random value

○ Use specified value (16 hex bytes, big endian format) 00112233445566778899AABBCCDDEEFF

Output

Format : RFP File : C:\DLM_Key_Injectioin\test_gui\NON-SECDBG.rkey

Address : 10000

```
Output File: C:\DLM_Key_Injectioin\test_gui\NON-SECDBG.rkey
UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F
W-UFPK: W-UFPK
IV: E4644901A648B260CE08801AB8B1C4E0
Encrypted key: Encrypted DLM key

OPERATION SUCCESSFUL
```

Figure 53. Generate the RFP Injection File for the NONSECDBG_KEY

4.5.2 Wrap DLM Keys Using SKMT CLI

This section shows how to create key injection files for the SECDBG and NON_SECBUG keys using the CLI interface.

4.5.2.1 Wrap a SECDBG_KEY

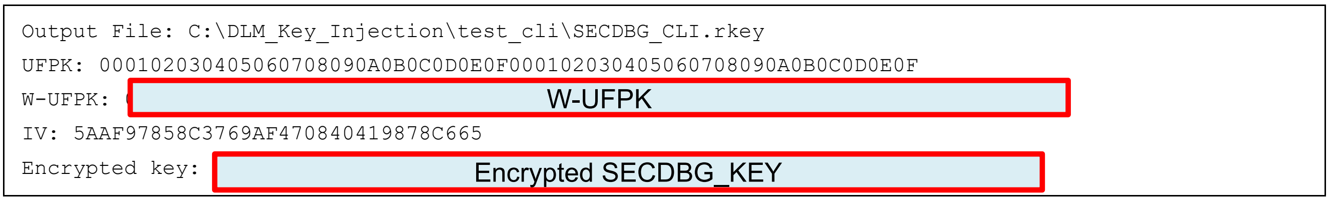
In a Command Prompt window, use the following command to create the SECDBG_KEY key injection file (`SECDBG_CLI.rkey`). Refer to the Security Key Management Tool user manual for more details about the `/genkey` command.

```
skmt.exe /genkey /ufpk file="C:\DLM_Key_Injection\test\ufpk.key" /wufpk
file="C:\DLM_Key_Injection\test\ufpk.key_enc.key" /mcu "RA-SCE9" /keytype
"DLM-SSD" /key "000102030405060708090A0B0C0D0E0F " /filetype "rfp" /output
"C:\DLM_Key_Injection\test_cli\SECDBG_CLI.rkey"
```

In this example:

- The UFPK key file created earlier is specified.
- The decrypted W-UFPK file received from the Renesas key wrap service is specified.
- The MCU selection for the RA6M4 is `RA-SCE9`, as per the options specified in the Security Key Management Tool User's Manual.
- The selected key type is `DLM-SSD`, as per the options specified in the Security Key Management Tool User's Manual.
- The DLM key value is `000102030405060708090A0B0C0D0E0F`. When this operation is performed for production hardware, this value should be replaced by a unique value and managed in a secure environment.
- Secure key injection on the RA6M4 is performed over the serial programming interface, so we must select `RFP` for the output file type.
- We are using a randomly generated IV. The IV is updated in each encryption instance, so if you execute this command multiple times, you will generate a different key injection file each time.
- The Renesas Key File that can be used by RFP to securely inject the DLM key will be generated as `SECDBG_CLI.rkey`.

The command will generate the Renesas key file as shown below.



```
Output File: C:\DLM_Key_Injection\test_cli\SECDBG_CLI.rkey
UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F
W-UFPK:
IV: 5AAF97858C3769AF470840419878C665
Encrypted key:
```

Figure 54. Create the SECDBG_KEY Injection File

The generated key file `SECDBG_CLI.rkey` can be used with RFP to inject the wrapped DLM SECDBG_KEY. This file contains the wrapped DLM key along with the W-UFPK. The plaintext SECDBG_KEY and UFPK are NOT contained in the `*.rkey` file, enabling confidential transfer of the key injection file.

4.5.2.2 Wrap a NONSECDBG_KEY

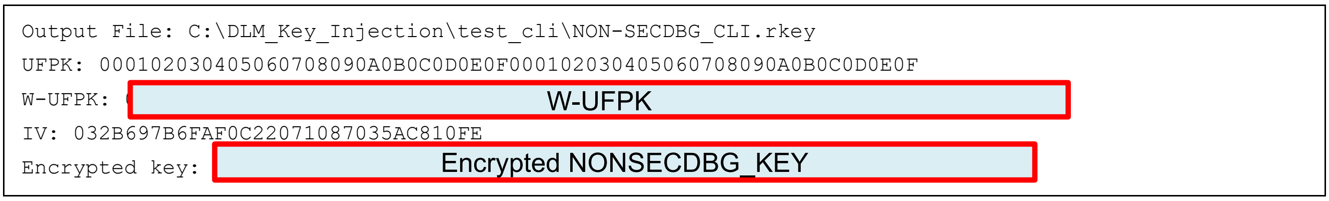
Similarly, use the following command to generate the NONSECDBG_KEY key injection file.

```
skmt.exe /genkey /ufpk file="C:\DLM_Key_Injection\test\ufpk.key" /wufpk
file="C:\DLM_Key_Injection\test\ufpk.key_enc.key" /mcu "RA-SCE9" /keytype
"DLM-NSECSD" /key "010102030405060708090A0B0C0D0E0F " /filetype "rfp" /output
"C:\DLM_Key_Injection\test_cli\NON-SECDBG_CLI.rkey"
```

This example is identical to the previous example, with the following differences:

- A different value is used for the DLM key (`0101020304050607080 90A0B0C0D0E0F`).
- The selected key type is `DLM-NSECSD`, as per the options specified in the *Security Key Management Tool User's Manual*.
- The Renesas Key File that can be used by RFP to securely inject the DLM key will be generated as `NON-SECDBG_CLI.rkey`.

The command will generate the Renesas key file, as shown below.



```
Output File: C:\DLM_Key_Injection\test_cli\NON-SECDBG_CLI.rkey
UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F
W-UFPK: W-UFPK
IV: 032B697B6FAF0C22071087035AC810FE
Encrypted key: Encrypted NONSECDBG_KEY
```

Figure 55. Create the Non-SECDBG DLM Key Injection File

The generated key file `NON-SECDBG_CLI.rkey` can be used with RFP to inject the wrapped DLM NONSECDBG_KEY. This file contains the wrapped DLM key along with the W-UFPK. The plaintext NONSECDBG_KEY and UFPK are NOT contained in the `*.rkey` file, enabling confidential transfer of the key injection file.

4.5.2.3 Wrap an RMA_KEY

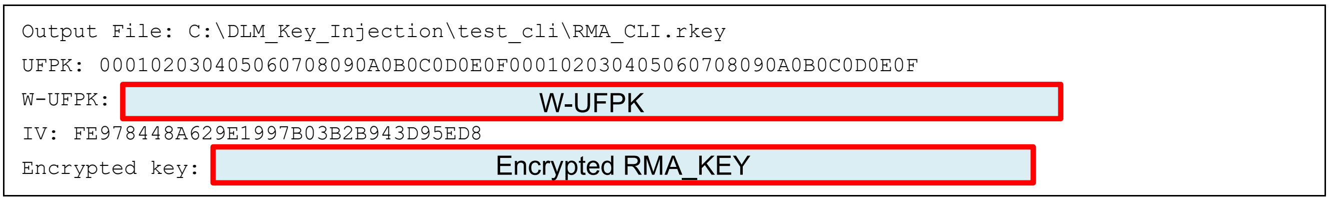
Similarly, the user can generate an RMA_KEY key injection file using the following command line input:

```
skmt.exe /genkey /ufpk file="C:\DLM_Key_Injection\test\ufpk.key" /wufpk
file="C:\DLM_Key_Injection\test\ufpk.key_enc.key" /mcu "RA-SCE9" /keytype
"DLM-RMA-REQ" /key "020102030405060708090A0B0C0D0E0F" /filetype "rfp" /output
"C:\DLM_Key_Injection\test_cli\RMA_CLI.rkey"
```

This example is identical to the previous example, with the following differences:

- A different value is used for the DLM key (`020102030405060708090A0B0C0D0E0F`).
- The selected key type is `DLM-RMA-REQ`, as per the options specified in the *Security Key Management Tool User's Manual*.
- The Renesas Key File that can be used by RFP to securely inject the DLM key will be generated as `RMA_CLI.rkey`.

The command will generate the Renesas key file, as shown below.



```
Output File: C:\DLM_Key_Injection\test_cli\RMA_CLI.rkey
UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F
W-UFPK: W-UFPK
IV: FE978448A629E1997B03B2B943D95ED8
Encrypted key: Encrypted RMA_KEY
```

Figure 56. Create the RMA DLM Key Injection File

The generated key file `RMA_CLI.rkey` can be used with RFP to inject the wrapped RMA_KEY. This file contains the wrapped DLM key along with the W-UFPK. The plaintext RMA_KEY and UFPK are NOT contained in the `*.rkey` file, enabling confidential transfer of the key injection file.

4.6 Inject the DLM Keys

This section demonstrates how to perform DLM Key injection. For instructions on how to create an RFP project and establish connections to the target board, see the *RFP User's Manual*. This section provides key configuration settings for each of the state transitions.

4.6.1 Inject a SECDBG_KEY

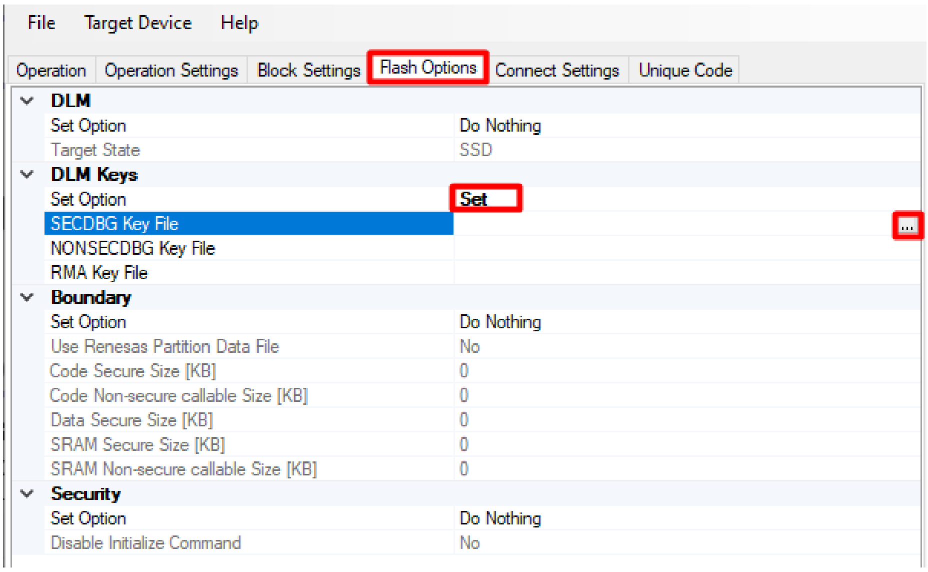
A Secure Debug Key (SECDBG_KEY) and a Return Material Authorization Key (RMA_KEY) can be injected when the MCU is in an SSD state.

Note: DLM key injection is not mandatory. It is also not mandatory to inject all possible DLM keys if only specific DLM capabilities are required. Be sure to examine the DLM states and authenticated transitions to determine which, if any, DLM keys should be injected.

This example uses the SECDBG_KEY Renesas key file generated in the previous section as an example. The RMA_KEY can be injected simultaneously using the same general steps. \Since the transition to RMA_REQ using the RMA_KEY cannot be regressed, it is not demonstrated here.

Unzip `ra6m4_dlm_key_inject_rfp.zip` to reveal `ra6m4_dlm_key_inject.rpj`. Launch RFP and open the RFP project `ra6m4_dlm_key_inject.rpj` and review the settings explained in this section.

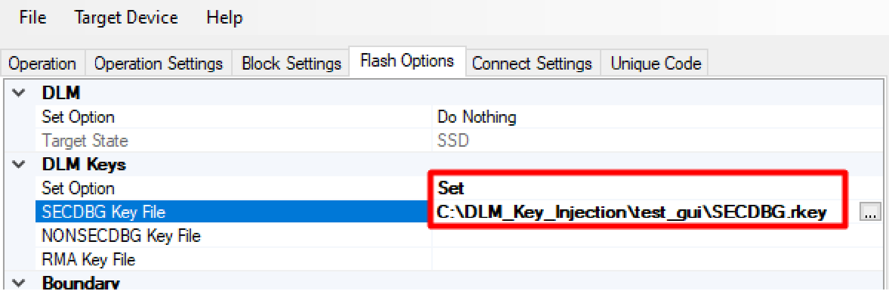
On the **Flash Options** tab, under **DLM Keys**, the option **Set Option** is set to **Set**. Click on **Encrypted SECDBG Key,** and then click the "**…**" to the right to select the corresponding SECDBG_KEY key file.



DLM	
Set Option	Do Nothing
Target State	SSD
DLM Keys	
Set Option	Set
SECDBG Key File	
NONSECDBG Key File	
RMA Key File	
Boundary	
Set Option	Do Nothing
Use Renesas Partition Data File	No
Code Secure Size [KB]	0
Code Non-secure callable Size [KB]	0
Data Secure Size [KB]	0
SRAM Secure Size [KB]	0
SRAM Non-secure callable Size [KB]	0
Security	
Set Option	Do Nothing
Disable Initialize Command	No

Figure 57. Select the SECDBG_KEY to Inject

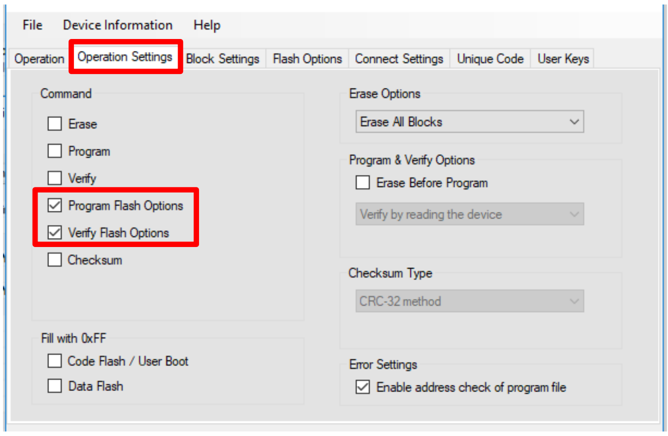
Select the `SECDBG.rkey` generated in section 4.5.



DLM	
Set Option	Do Nothing
Target State	SSD
DLM Keys	
Set Option	Set
SECDBG Key File	C:\DLM_Key_Injection\test_gui\SECDBG.rkey
NONSECDBG Key File	
RMA Key File	
Boundary	

Figure 58. Select the SECDBG_KEY to Inject

Under the **Operation Settings** tab, **Program Flash Options** and **Verify Flash Options** are selected.



Command
□ Erase
□ Program
□ Verify
☑ Program Flash Options
☑ Verify Flash Options
□ Checksum

Fill with 0xFF
□ Code Flash / User Boot
□ Data Flash

Erase Options: Erase All Blocks

Program & Verify Options
□ Erase Before Program
Verify by reading the device

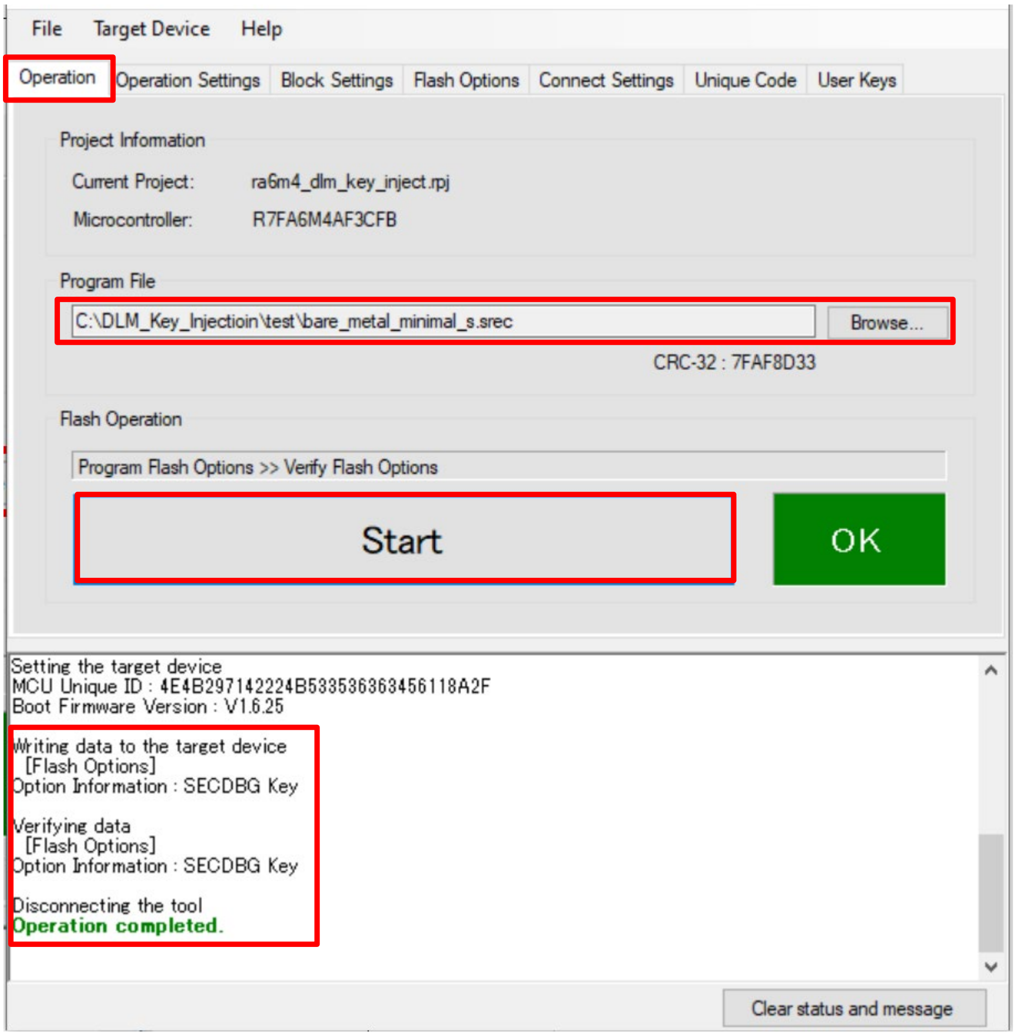
Checksum Type: CRC-32 method

Error Settings
☑ Enable address check of program file

Figure 59. Select Program and Verify Flash Option Setting

DLM keys and program code can be programmed at the same time or separately. Unzip `test.zip` to reveal `bare_metal_minimal_s.srec` and `bare_metal_minimal_ns.srec`. These binaries are included with this application note for demonstration, but **Program File** can also be left blank and **Code Flash** unchecked in the **Block Settings** tab.

Navigate to the **Operation** tab, select the secure application binary (if desired), and click **Start** to program the data.



Project Information
Current Project: ra6m4_dlm_key_inject.rpj
Microcontroller: R7FA6M4AF3CFB

Program File
C:\DLM_Key_Injectioin\test\bare_metal_minimal_s.srec
CRC-32 : 7FAF8D33

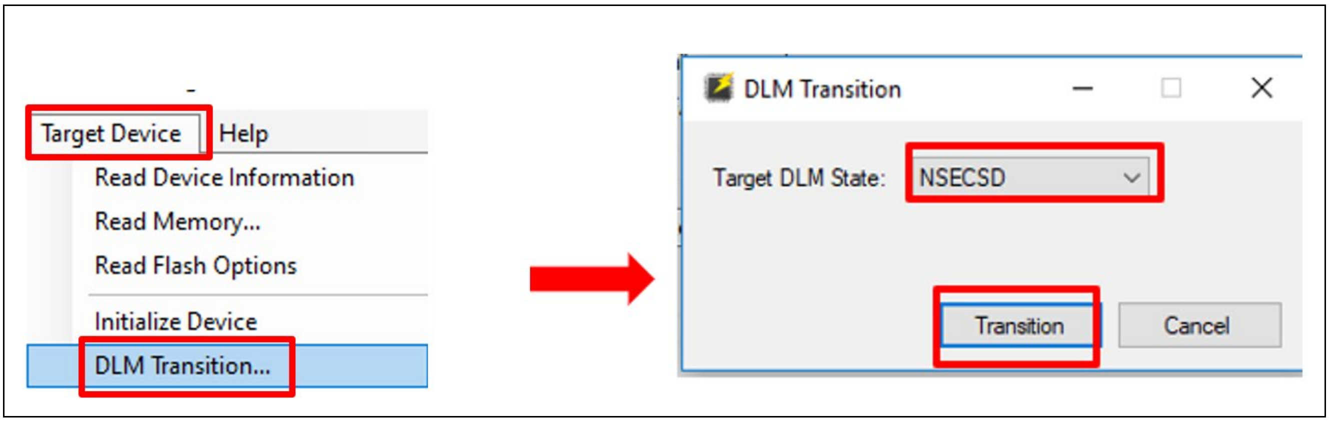
Flash Operation
Program Flash Options >> Verify Flash Options

```
Setting the target device
MCU Unique ID : 4E4B297142224B533536363456118A2F
Boot Firmware Version : V1.6.25

Writing data to the target device
  [Flash Options]
Option Information : SECDBG Key

Verifying data
  [Flash Options]
Option Information : SECDBG Key

Disconnecting the tool
Operation completed.
```

Figure 60. Inject the SECDBG_KEY

4.6.2 Inject a NONSECDBG_KEY

The NONSECDBG_KEY DLM key must be injected when the MCU is in the NSECSD DLM state. Transition the MCU to this state as shown below.

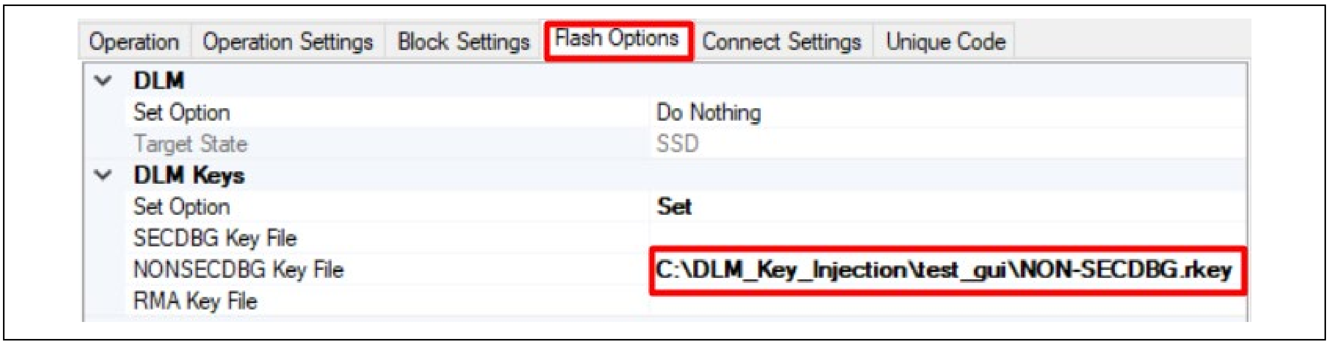


Target DLM State: NSECSD

Figure 61. Transition the MCU Device Lifecycle State to NSECSD

Next, follow similar steps as in section 4.6.1 to inject the NONSECDBG_KEY DLM Key. In this example, we can use the NONSECDBG_KEY (`NONSECDBG.rkey`) generated from Figure 53 to illustrate the operation.

Add the NONSECDBG_KEY key file entry as shown below. Note that any SECDBG and RMA key file entries must be deleted.

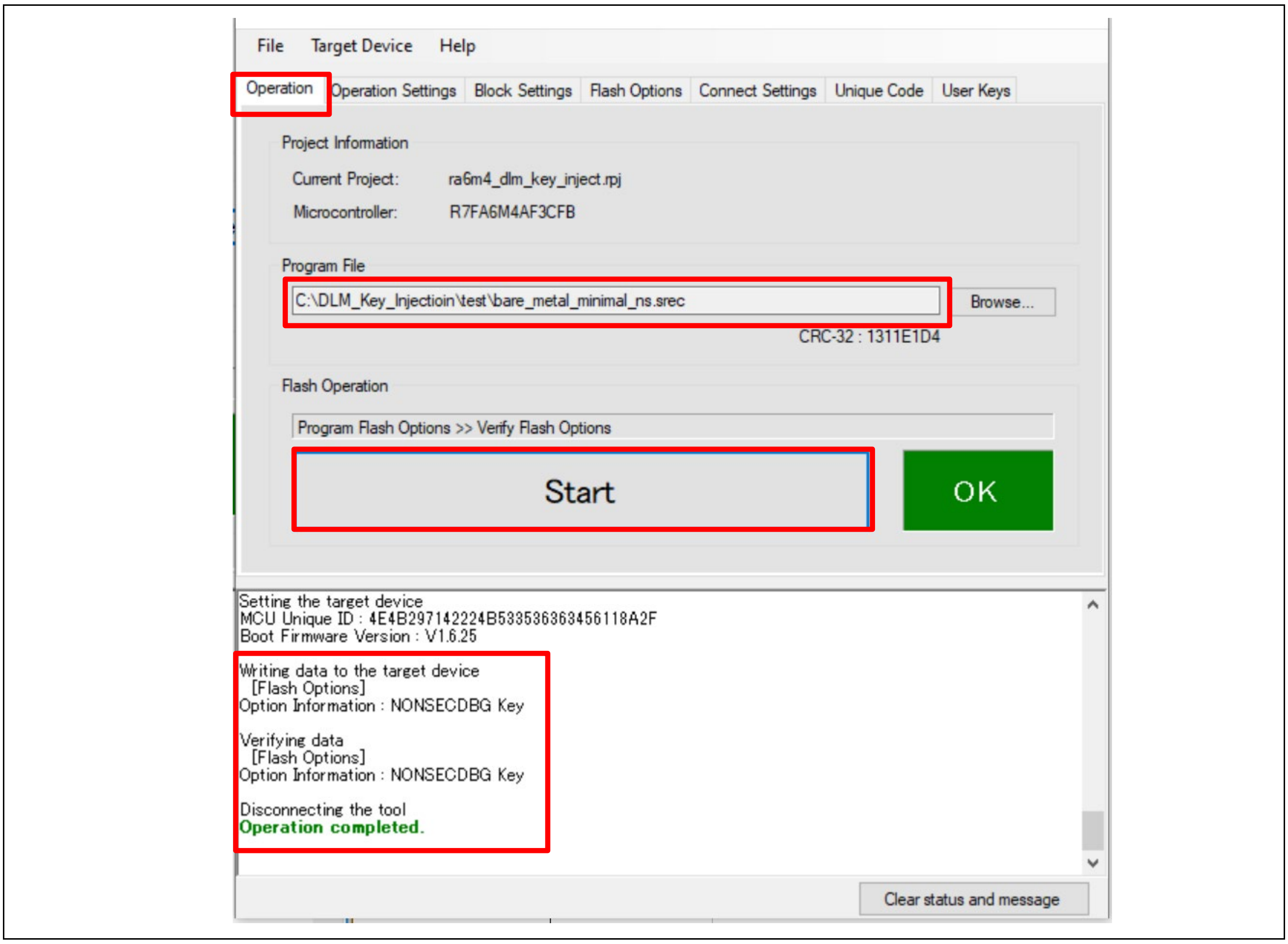


DLM	
Set Option	Do Nothing
Target State	SSD
DLM Keys	
Set Option	**Set**
SECDBG Key File	
NONSECDBG Key File	**C:\DLM_Key_Injection\test_gui\NON-SECDBG.rkey**
RMA Key File	

Figure 62. Select the NONSECDBG_KEY to Inject

DLM keys and program code can be programmed at the same time or separately. Unzip `test.zip` to reveal `bare_metal_minimal_s.srec` and `bare_metal_minimal_ns.srec`. These binaries are included with this application note for demonstration, but **Program File** can also be left blank and **Code Flash** unchecked in the **Block Settings** tab.

Navigate to the **Operation** tab, select the non-secure application binary (if desired), and click **Start** to program the data.



Current Project: ra6m4_dlm_key_inject.rpj

Microcontroller: R7FA6M4AF3CFB

Program File: C:\DLM_Key_Injectioin\test\bare_metal_minimal_ns.srec

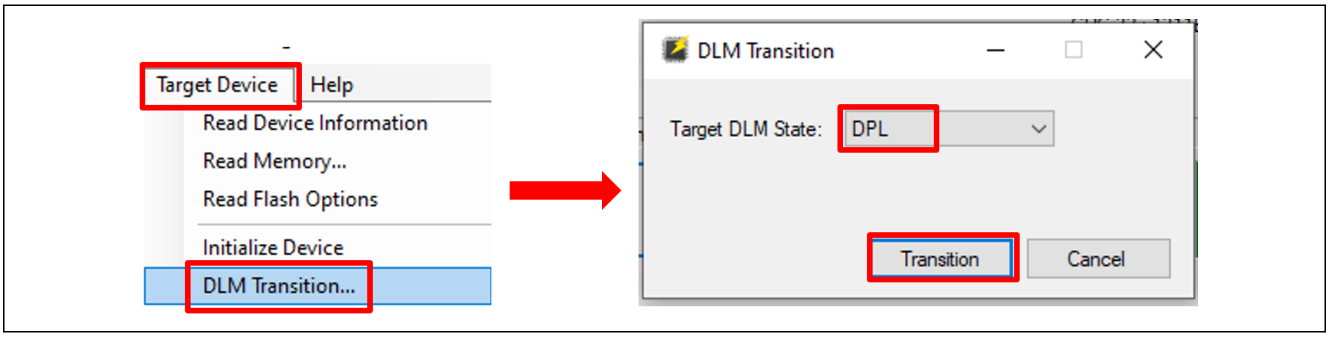
CRC-32 : 1311E1D4

Program Flash Options >> Verify Flash Options

```
Setting the target device
MCU Unique ID : 4E4B297142224B533536363456118A2F
Boot Firmware Version : V1.6.25

Writing data to the target device
  [Flash Options]
Option Information : NONSECDBG Key

Verifying data
  [Flash Options]
Option Information : NONSECDBG Key

Disconnecting the tool
Operation completed.
```

Figure 63. Inject the NONSECDBG_KEY Using RFP

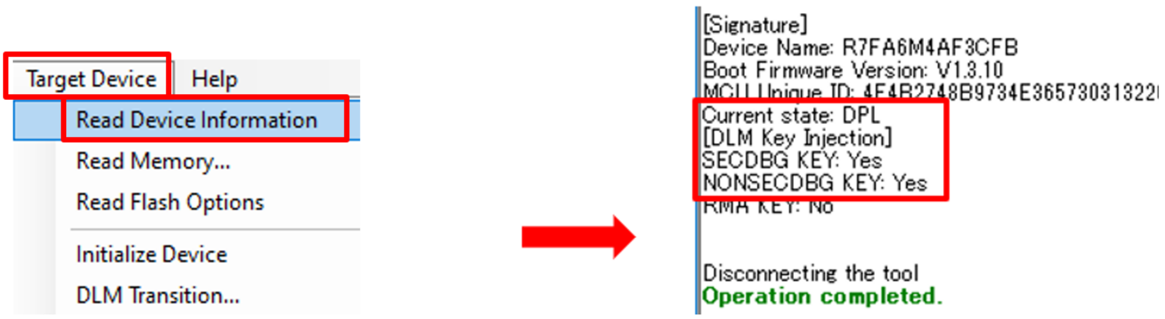
To test the authenticated DLM State regression as shown in section 5, Authenticated DLM State Transitions, we can advance the DLM state to DPL.



Target DLM State: DPL

Figure 64. Transition the MCU Device Lifecycle State to DPL

The MCU Device Lifecycle State can be confirmed by reading out the Device Information as shown in the following graphic.



```
[Signature]
Device Name: R7FA6M4AF3CFB
Boot Firmware Version: V1.3.10
MCU Unique ID: 4F4B2748B9734E36573031322
Current state: DPL
[DLM Key Injection]
SECDBG KEY: Yes
NONSECDBG KEY: Yes
RMA KEY: No


Disconnecting the tool
Operation completed.
```

Figure 65. Confirm the DPL Device Lifecycle State

5. Authenticated DLM State Transitions

This section provides operational steps for authenticated DLM state transitions using RFP. The assumption is that SECDBG_KEY and NONSECDBG_KEY are already injected using the previous examples.

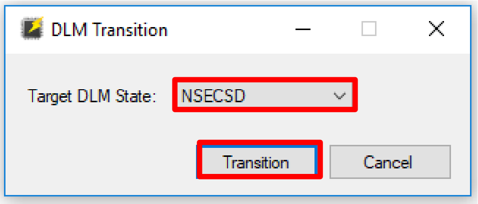
Note: For practice purposes, you can inject an RMA_KEY. However, unless you are returning the MCU to Renesas for failure analysis, DO NOT transition to the RMA_REQ state using the RMA_KEY. Once transitioned to RMA_REQ state, the **Initialize** command is disabled, and the MCU will be locked out of debugging and serial reprogramming capabilities.

5.1 Authenticated Transition from Deployed State to Non-secure Debug State

As explained in section 1.2, if the DLM state is DPL and a NONSECDBG_KEY has been injected, it is possible to regress the MCU device lifecycle state from DPL to NSECSD and retain flash memory contents.

Use the following steps to regress the Device Lifecycle State to NSECSD.

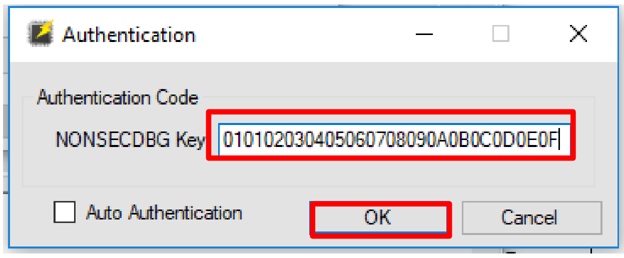
1. Select to transition to **NSECSD**.



Target DLM State: NSECSD

Figure 66. Select Transition to NSECSD

2. If the device is in the DPL state and a NONSECDBG_KEY has been injected on the MCU, the following prompt will pop up. Follow the prompt to provide the NONSECDBG_KEY and then click **OK**. In the example shown here, this is `010102030405060708090A0B0C0D0E0F`.

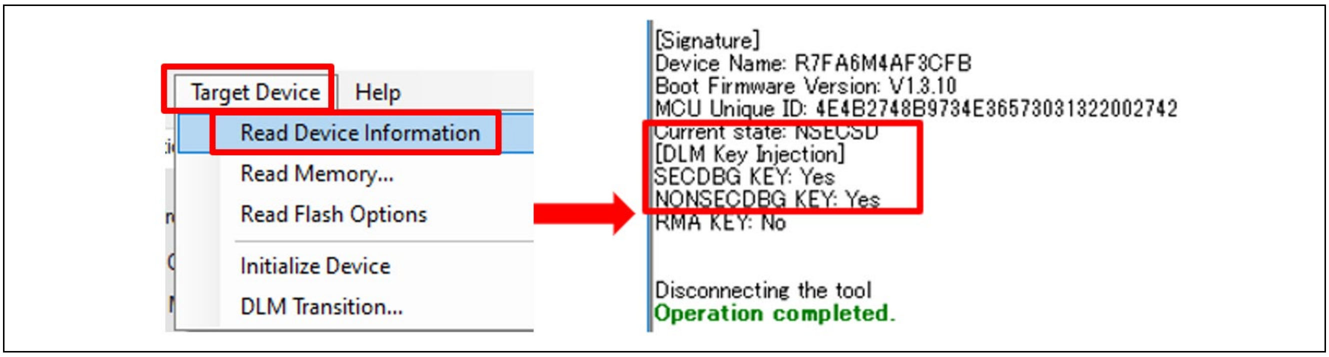


NONSECDBG Key 0101020304050607080 90A0B0C0D0E0F

□ Auto Authentication

Figure 67. Provide the Authentication Key for NONSECDBG

3. You can now confirm that the Device Lifecyle State is regressed back to NSECSD state.



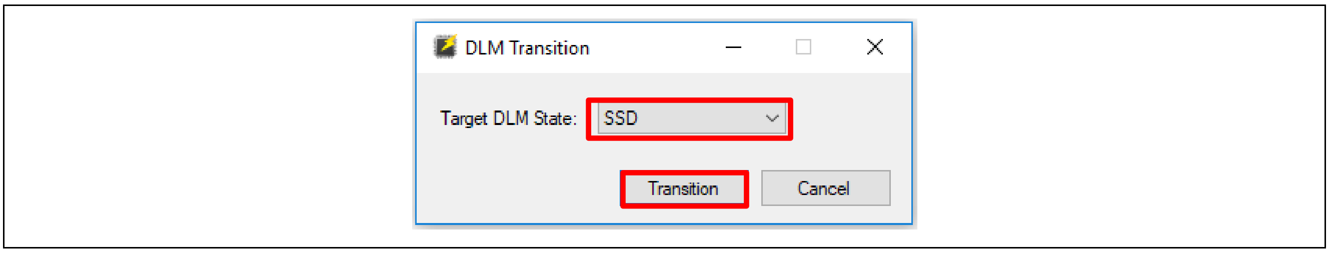
```
[Signature]
Device Name: R7FA6M4AF3CFB
Boot Firmware Version: V1.3.10
MCU Unique ID: 4E4B2748B9734E36573031322002742
Current state: NSECSD
[DLM Key Injection]
SECDBG KEY: Yes
NONSECDBG KEY: Yes
RMA KEY: No

Disconnecting the tool
Operation completed.
```

Figure 68. Confirm the MCU is in NSECSD and SECDBG_KEY is Injected

5.2 Authenticated Transition from Non-secure Debug State to Secure Debug State

When the device is in the NSECSD DLM state, and a SECDBG_KEY has been injected, the following steps will regress the MCU to the SSD DLM state without erasing flash memory:

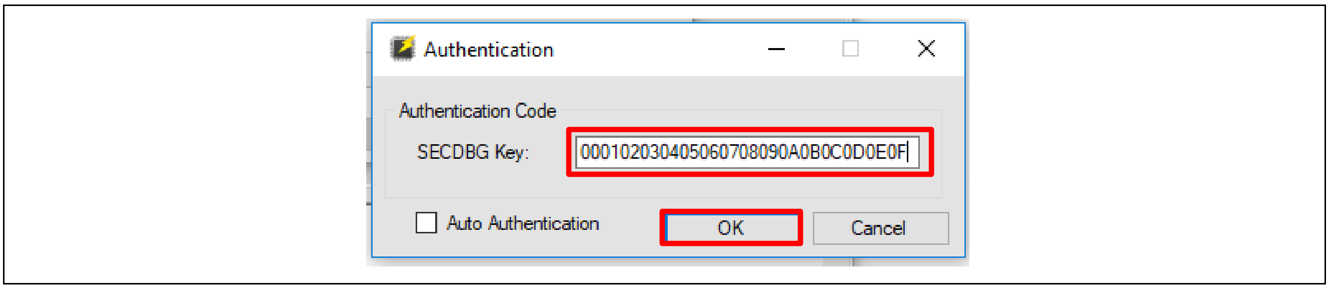
1. Select to transition to **SSD**.



Target DLM State: SSD

Figure 69. Select Transition to SSD

2. If the Device is in NSECSD state and a SECDBG_KEY has been injected on the MCU, the following prompt will pop up. Follow the prompt to provide the SECDBG_KEY and then click **OK**. In the example shown here, this is `000102030405060708090A0B0C0D0E0F`.

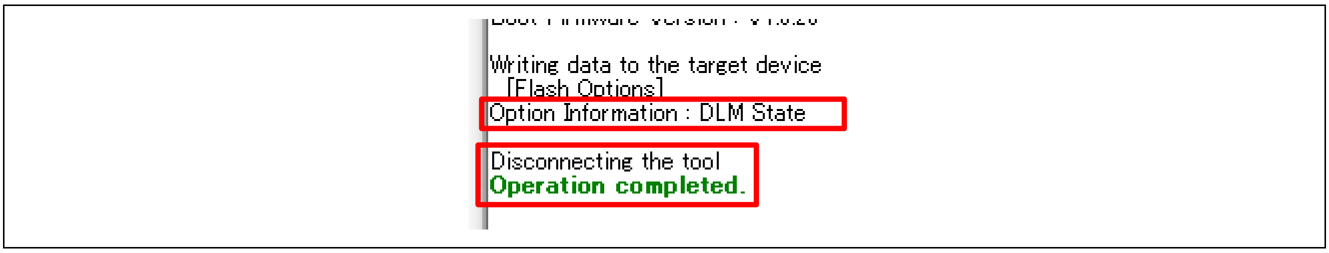


Authentication Code

SECDBG Key: 000102030405060708090A0B0C0D0E0F

Figure 70. Provide the Authentication Key for SECDBG

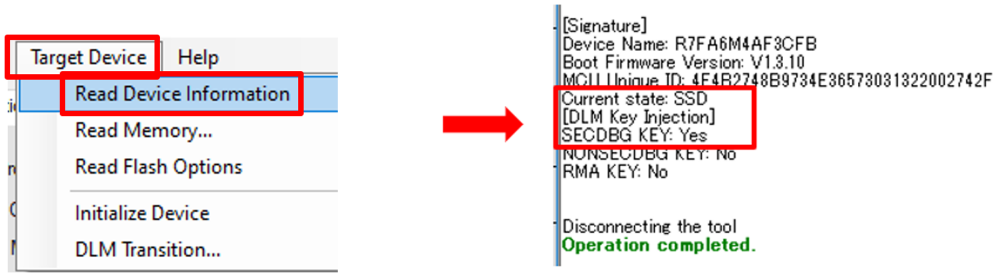
3. If the DLM state regression is successful, the following output will be printed.



```
Boot Firmware Version : V1.3.10

Writing data to the target device
  [Flash Options]
Option Information : DLM State

Disconnecting the tool
Operation completed.
```

Figure 71. RFP Output with DLM State Regression

4. You can now confirm that the Device Lifecycle State has transitioned back to SSD state.



```
[Signature]
Device Name: R7FA6M4AF3CFB
Boot Firmware Version: V1.3.10
MCU Unique ID: 4F4B2748B9734E3657303132002742F
Current state: SSD
[DLM Key Injection]
SECDBG KEY: Yes
NONSECDBG KEY: No
RMA KEY: No


Disconnecting the tool
Operation completed.
```

Figure 72. Confirm Device Lifecycle has Transitioned Back to SSD

6. References

- Renesas RA Family Secure Key Injection and Update Application Project (R11AN0496)
- Renesas RA6M4 Group User’s Manual: Hardware
- Flexible Software Package (FSP) User’s Manual

7. Appendix

7.1 Glossary

Term	Meaning
SCE9	Secure Crypto Engine 9 is a hardware unit that resides on Renesas Arm® Cortex®-M33 MCU
Device Certificate	Certificate uniquely identifying an individual device. It is digitally signed, asserting that the certificate comes from a known source and has not been modified and that the device is trusted.
Root of Trust	Roots of trust are highly reliable hardware, firmware, and software components that perform specific, critical security functions. (https://csrc.nist.gov/projects/hardware-roots-of-trust)
SCE	Secure Crypto Engine – A module in the MCU that provides for efficient, low-power cryptographic acceleration, TRNG (True Random Number Generation), and creation and isolation of cryptographic keys.
PKI	Public Key Infrastructure – A set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates, which are typically used to manage secure identity via public key cryptography.
Key Pair	Asymmetric keys are generated in pairs – a public and private key. The private key is held in secret by only one party and can be used to assert that party’s identity. The public key is freely distributed and is uniquely associated with the private key.
Secure Code	A function or group of functions that resides in a secure region of internal flash, as defined and enforced by the MPUs. These secure functions can access both secure data and non-secure data regions.
Non-Secure code	A function or group of functions that resides in a non-secure region of internal flash. These non-secure codes cannot access the secure region. They can only access the non-secure region.
HUK	Hardware Unique Key. This is a key stored inside the RA Family MCU that is unique to every individual MCU.
Challenge String	Randomly generated string at the host application. This string is used by the host application to validate the ownership of the private key by the target.
Unique ID	An identification value, unique to each individual RA Family MCU is stored inside the MCU.
Challenge Response String	The response to the challenge string. The Challenge Response String is the signature of the challenge data as created by signing the Challenge String with the receiver's private key.

Website and Support

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

EK-RA6M4 Resources	renesas.com/ra/ek-ra6m4
RA Product Information	renesas.com/ra
RA Product Support Forum	renesas.com/ra/forum
RA Flexible Software Package	renesas.com/FSP
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct.01.20	—	First release document
1.10	Dec.09.20	—	Updated importing Renesas Public Key to Kleopatra procedure
2.00	June.23.22	—	Updated to use Security Key Management Tool (SKMT)
2.10	Sep.10.24	—	Update with environment FSP version 5.5.0

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/.