User's Manual

RENESAS

Renesas Synergy™ Security Evaluation

User's Manual: Software

RENESAS Synergy™

Renesas Electronics
www.renesas.com

Rev.1.00 April 2017

Contents

1.Introduction

1.1 Glossary

Term	Meaning
AES	Advanced Encryption System - symmetric encryption algorithm [2]
Authentication	Unambiguous identification of a device
Cryptography	The practice and study of techniques for secure communication in the presence of third parties
Cryptographic Strength	Measure of the resistance of a cryptographic algorithm to attack.
Device	A single Renesas Synergy series silicon chip.
Device Certificate	Certificate identifying an individual Synergy device
Device Seed	Device-specific random number used to derive device-specific keys
DSA	Digital Signature Algorithm
Cryptographic Hash	A hash function which is considered practically impossible to invert, that is, to recreate the input data from its hash value alone
JTAG	A communications standard widely used for integrated circuit debug ports
Module	A block of code which executes on a device and which can be installed and version managed independently
MPU	Memory Protection Unit – protects flash and RAM from interference
Root of Trust	A component with a system whose integrity must be assumed
RSA	Rivest, Shamir, Adelman asymmetric encryption algorithm
RTOS	Real Time Operating System – the operating system which may run in supervisor mode on the device
Security Kernel	The module containing all the security code on the device
Secure Boot	The mechanism which checks that all code running on the device is correct and has not been compromised
SHA-1	160-bit cryptographic hash algorithm [3]
SHA-256	256-bit cryptographic hash algorithm [3]
Supervisor Mode	Privileged mode in which the RTOS normally operates
User Mode	The totally unprivileged mode in which applications should run on the device

1.2 References

1. NIST SP 800-57 Recommendation for Key Management Part 1
2. AES: FIPS-197
3. SHA-256: FIPS 180-4
4. NIST Recommended Curves for Government use
5. RFC 7525 - Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)

2.Synergy Security Evaluation System

2.1 Purpose

The goal of the evaluation system is to allow you to evaluate the Synergy Security Architecture and in particular the mechanisms that enable Secure Manufacturing and Secure Update of Devices and Device Secure Boot. An example application provides a walk-through of application building, secure manufacturing, and the ability to generate application updates and secure installation. In particular, the Evaluation System will:

- Provide a Renesas Synergy PK- S5D9 development board with Secure Kernel pre-installed and appropriate serial cabling to allow secure installation of applications
- Provide an e^2 studio build environment for building applications suitable for install on the development board
- Provide a Secure Mastering tool, which will convert an application binary into the format required for Secure Manufacturing
- Provide a Secure Manufacturing tool which will allow the installation of Secured applications onto the development board
- Provide a version of the Secure Mastering tool which allows secured updates to be generated from an application binary, with control over update versioning that is permitted
- Provide an example of an update server which will permit the update of an application onto the development board, according to the rules defined using the mastering tool
- Provide documentation for a walk-though of the development, mastering, manufacturing, and update process and enable you to build your own application and step through the processes defined.
- (Provide a Secure Kernel Install tool in order to enable Renesas to program Synergy PK-S5D9 development boards with a Secure Kernel)

Note: The evaluation system is not intended to be used for production devices

2.2 Evaluation Kit

The Evaluation Kit consists of the following items:

- Renesas PK-S5D9 Development Kit including standard cabling
- 2 x USB to RS232 Serial TTL Converter Adapters
- Jump Wires
- 2 x USB Smart Card Readers
- Smart card pre-programmed with Manufacturing Key handling application
- Documentation – Evaluation User Guide

2.3 Use cases

The Synergy Security Evaluation supports the following use cases.

2.3.1 Secure Boot

From reset, the device executes secure boot code which checks the authenticity and integrity of application code before the application code is executed. In the event that the checking fails, then the application code will not be executed. It is possible to define a recovery mechanism to allow correctly signed application code to be re-installed on the device.

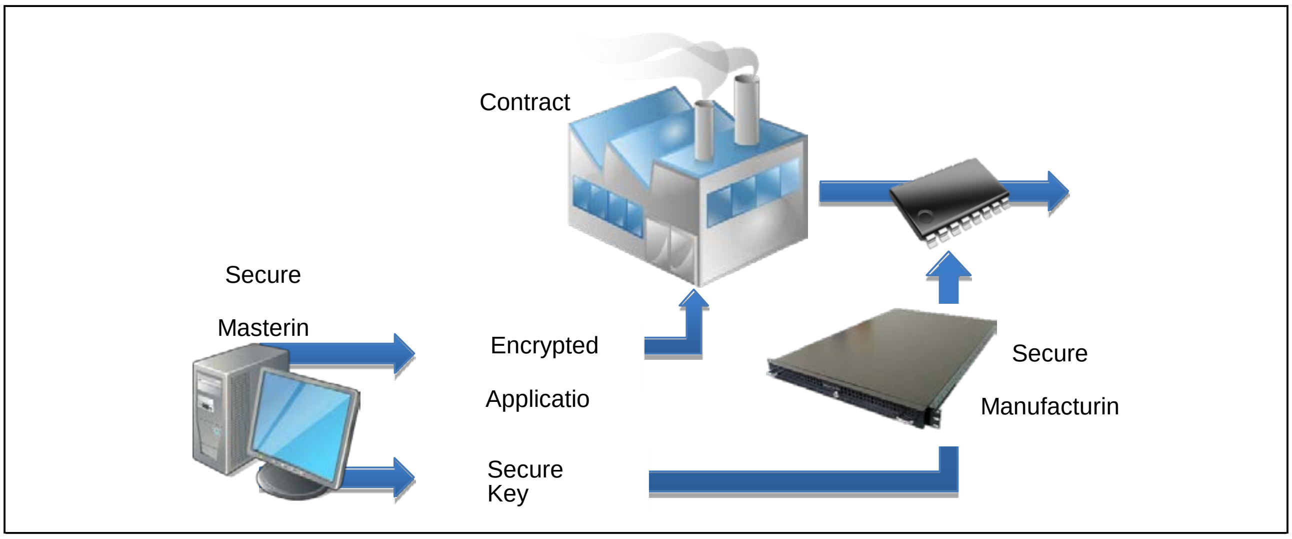


Figure 1 Use Cases

2.3.2 Secure Mastering and Manufacturing

The Secure Manufacturing walkthrough shows the process for generating and programming a secured application onto the evaluation system. The evaluation system is pre-installed with a Security Kernel that includes a Secure Boot process. This will only allow application code that has correct Integrity and Authenticity checks applied to it to be installed and run on the device.

The process can be used to protect the application code from exposure to 3rd parties such as the Contract Manufacturer as the Application binary is encrypted using an AES key and signed by the Secure Mastering tool to validate its authenticity and integrity. The Key material used to manage the application binary is held securely together with a user defined count of the number of devices to be programmed. The Key material is held on a smart card for this evaluation, but may be sent securely to a HSM (Hardware Security Module) at the CM (Contract Manufacturer).

During the Manufacturing process, the Encrypted binary is downloaded to the device, together with its Key material which is uniquely re-encrypted for the specific device being programmed

2.3.3 Secure Update

The Secure Update walk-through shows the process of generating an installing an application update onto a development board which has been through the Secure Manufacturing process. The Update is also a 2 stage process. First the Application Update binary is passed through a mastering process to package and encrypt the application binary. Second, the encrypted binary is sent to the Synergy board together with Key material which allows the binary to be updated. The update mechanism has flexibility in the methods of rollback or rollback protection that it allows.

2.4 System Components

2.4.1 Design Flow

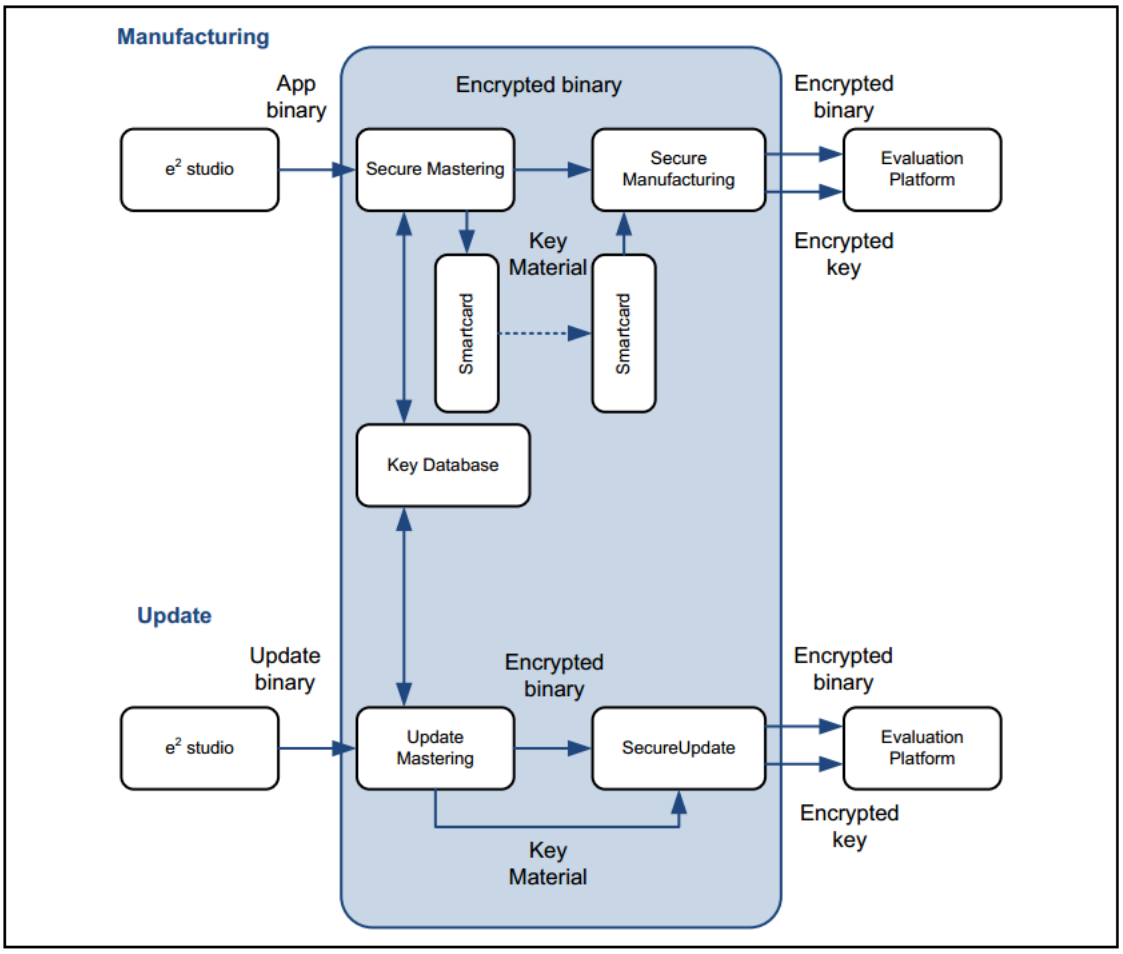


Figure 2 Design Flow

2.4.2 Application Development

Application development occurs in the standard way using e^2 studio and the Synergy SSP. (There is a slightly modified address map to cope with the Security Kernel which is installed at the base of the address map).

2.4.3 Evaluation System

The Evaluation system consists of a Renesas Synergy PK-S5D9 standard development board which has had a Security Kernel pre-installed onto the device. With the Security Kernel loaded, the Secure Manufacturing process must be followed to install application binaries onto the device.

2.4.4 Security Kernel

The Security Kernel contains the secure boot code, a unique device certificate and unique Device Seed used to generate device specific keys. In addition, the Receiver1stTime module is used to load the application binary onto the device through the SCI -2 during the secure manufacturing process.

2.4.5 Secure Mastering

The Secure Mastering program is a PC application which takes the Application binary (in srecord format) from the output of the e^2 studio development process and encrypts and signs the binary in preparation for the Secure Manufacturing process. In a real development environment, this step will be managed in a secure environment. The keys used to sign the binary are stored in a key database.

The keys used to encrypt and sign the binary are stored on a smart card for use during the Secure Manufacturing process.

2.4.6 Secure Manufacturing

The Secure Manufacturing program is a PC application with emulates the production programming environment. The encrypted application binary can be installed onto evaluation systems connected to the PC via a USB-to-Serial convertor. Only devices with a valid device certificate will be programmed. The key material for the encrypted binary is stored on the smart card and only released (in a re-encrypted form) to the valid device.

2.4.7 Update Mastering

Update mastering is a PC application that takes updates of Application binary (in srecord format) from the output of the e^2 studio development process and encrypts and signs the binary in preparation for Secure Update. The rules for how updates are managed can be set.

The keys used to sign the binary are stored in the key database.

The keys used to encrypt and sign the binary are passed directly to the Secure Update Application. In a production system, this would be handled through a secure channel or installation process.

2.4.8 Secure Update

The Secure Update Program is a PC program which emulates an Update server. Evaluation systems connected to the PC via a serial cable can be updated to a new version of the Application binary provided that the rules for Updates are met.

2.4.9 Key Database

The Key Database is an ASCII file that holds the keys used for the mastering parts of the Evaluation system. For the purposes of the evaluation only, the database is not encrypted. The same database is used for all parts of the evaluation. In particular, both public and private parts of all keys are present in the database including the ‘root’ key. This would not be the case in a production system.

2.4.10 Smart card

A smart card and two smart card readers are provided for the evaluation system. The smart card handles the key material during the Secure Mastering and Secure Manufacturing process. It is used to demonstrate how the key material can be handled securely during a Contract Manufacturing process. Once the key material is installed on the smart card, this can be passed to the manufacturer for production. The key material never leaves the smart card in unencrypted form. It is re-encrypted uniquely for each device programmed. In a real production system, the smart card could be replaced by an HSM (Hardware Security Module) or other means to protect the key material.

3.Example Walk Through

3.1 Assumptions

It is assumed that the user is familiar with the Renesas Synergy PK-S5D9 microcontroller and the PK-S5D9 development board together with the Renesas tools and design environment. See the PK-S5D9 Quick Start Guide and PK-S5D9 User's Manual for more details.

The Security Kernel must be installed on the PK-S5D9. If the Security Kernel is not installed yet, please see Security Kernel Install Application Note (synergy-secure-kernel.pdf) for instructions.

3.2 Requirements

PC Requirements: Microsoft® Windows® 10 with Intel® Core™ family processor running at 2.0 GHz or higher (or equivalent processor), 8 GB memory, 1TB hard disk or SSD, 5 free USB 2.0 ports, Connection to the Internet

- e^2 studio Integrated Solution Development Environment (ISDE) from Renesas.
- Renesas Synergy^{TM} Software Package (SSP) version 1.2.0.

3.3 Installation of Evaluation Software

The evaluation software requires the pre-installation of drivers for the smart card readers and Microsoft Visual Studio 2008 runtime redistributable.

3.3.1 Install Smart Card Reader Drivers

Before connecting the smart card reader to the PC, download and install the drivers from support.gemalto.com.

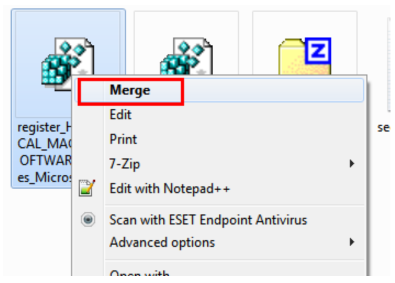
http://support.gemalto.com/index.php?id=pc_usb_tr_and_pc_twin#.VyzJzYQrLIU

Then connect a card reader to a USB socket on the PC (without a smart card installed). Edit two registry entries (create the entries if they do not exist):

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\ScPnP] EnableScPnP=dword:00000000
```

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Policies\Microsoft\Windows\ScPnP]
EnableScPnP=dword:00000000
```

Alternatively, import the two registry keys included in C:\Renesas\Synergy\SecurityKernel\regkey by selecting "Merge" from Popup menu



The PC will need to be re-booted for the changes to take effect.

Insert the smart card into the reader, ensuring that the contacts are oriented correctly. The light on the reader should turn green.

3.3.2 Install VS2008 Runtime

It is necessary to install Microsoft Visual C++ 2008 Redistributable package. This available for download from Microsoft in both 32bit (x86) and 64bit (x64) Windows forms

https://www.microsoft.com/en-us/download/details.aspx?id=29
https://www.microsoft.com/en-us/download/details.aspx?id=15336

3.4 Install Evaluation Software

The Evaluation Software is provided in Renesas_Synergy_Security_Kit.zip.
Extract the contents of the zip file to **C:\Renesas\Synergy**.

3.4.1 Setup of Development Board

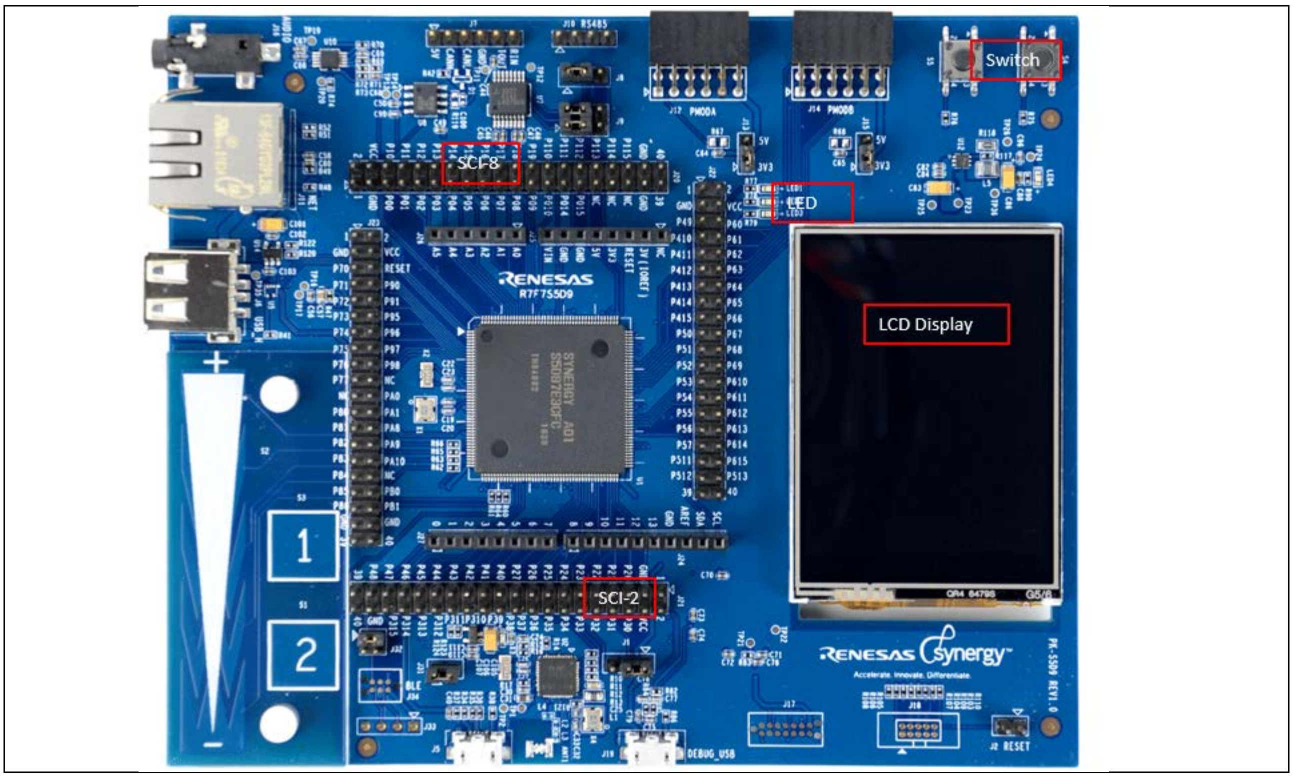


Figure 3 PK-S5D9 v1.0 Configuration

3.5 Connectivity

- Connect USB to RS232 Serial TTL adapters to SCI-2 and SCI-8 (see Figure 4)
- Connect serial adapters to a USB port on the PC (see Figure 4)
- Make sure Jumper J1 is in 1-2(default) positon
- Connect both smart card readers to the PC.

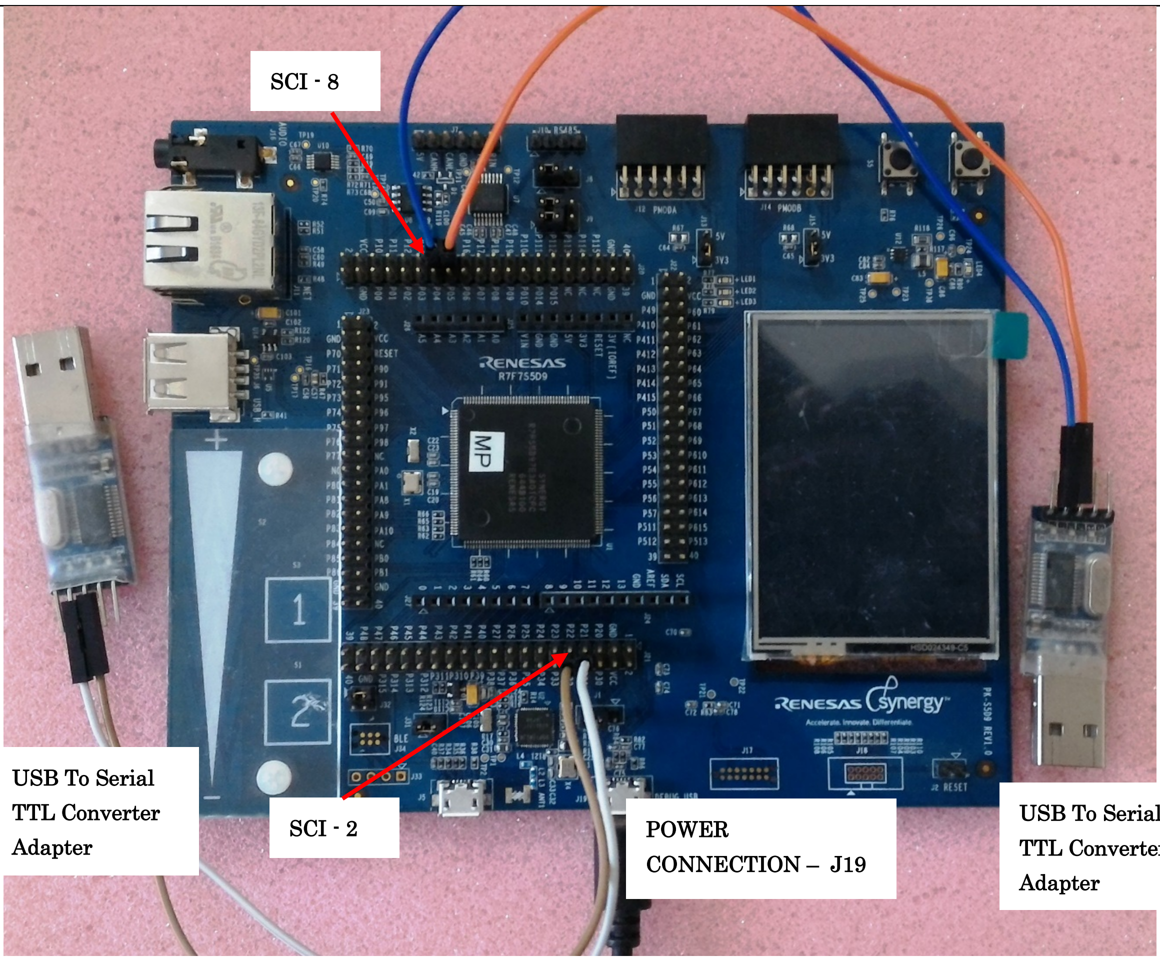

SCI - 8
USB To Serial TTL Converter Adapter
SCI - 2
POWER CONNECTION – J19
USB To Serial TTL Converter Adapter

Figure 4 PK - S5D9 Connectivity

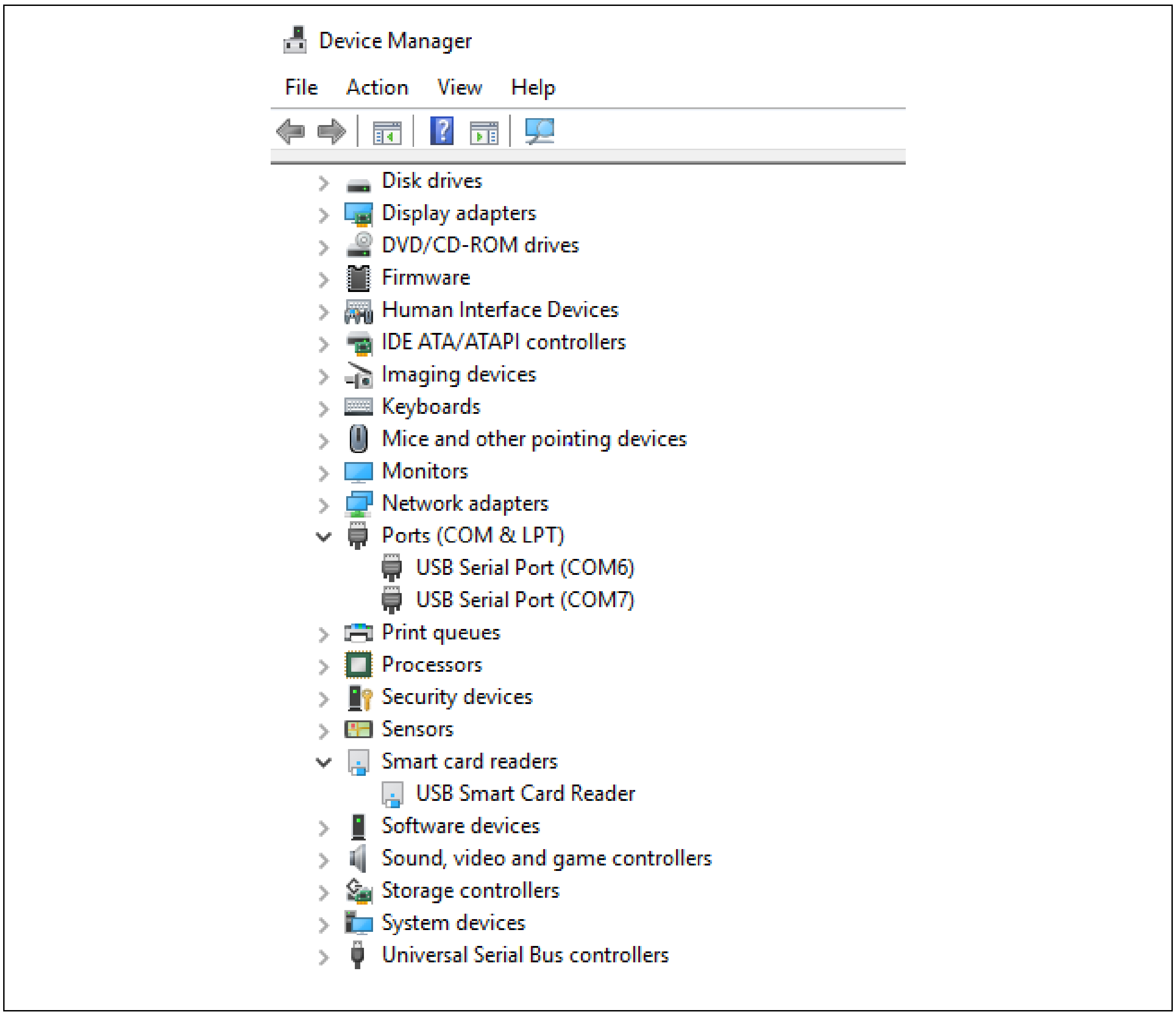
Use below pins from Breakout header J20 for SCI8.

PK - S5D9			USB To Serial TTL Converter Adapter
S5D9 Pins	**Label on PK-S5D9 Board**	**Function Name**	
P105	P15	SCI8-TX	Pin -RX
P104	P14	SCI8-RX	Pin -TX

Use below pins from Breakout header J21 for SCI2.

PK - S5D9			USB To Serial TTL Converter Adapter
S5D9 Pins	**Label on PK-S5D9 Board**	**Function Name**	
P302	P32	SCI2-TX	Pin -RX
P301	P31	SCI2-RX	Pin -TX

Use the Windows Device manager to identify the Serial Port being used by the USB-to-Serial converter adapters.



- Disk drives
- Display adapters
- DVD/CD-ROM drives
- Firmware
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Imaging devices
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
  - USB Serial Port (COM6)
  - USB Serial Port (COM7)
- Print queues
- Processors
- Security devices
- Sensors
- Smart card readers
  - USB Smart Card Reader
- Software devices
- Sound, video and game controllers
- Storage controllers
- System devices
- Universal Serial Bus controllers

Figure 5 Identifying the Serial Port in Device Manager

In this case, two USB to Serial TTL converter adapters becomes COM6 and COM7. One is used for console debug messages and another one is used to deploy application image.

Note also that the USB Smart Card Reader device is installed.

3.6 Secure Manufacturing

The Secure Manufacturing walkthrough consists of two stages.

1. The Secure Mastering tool will be used to package a pre-generated application binary into a signed and encrypted format. The key material for this will be stored on a smart card.
2. The Secure Manufacturing tool will be take the encrypted binary and key material and securely program the validated development board with the application.

The PC will be used both for the mastering tool and also as the Secure Manufacturing programmer

3.6.1 Secure Mastering

3.6.1.1 Preparation of Signed and Encrypted Application

Run C:\Renesas\Synergy\SecurityKernel\evaluation\gui\MasterManufacturingGUI.exe to bring up the window in Figure 6

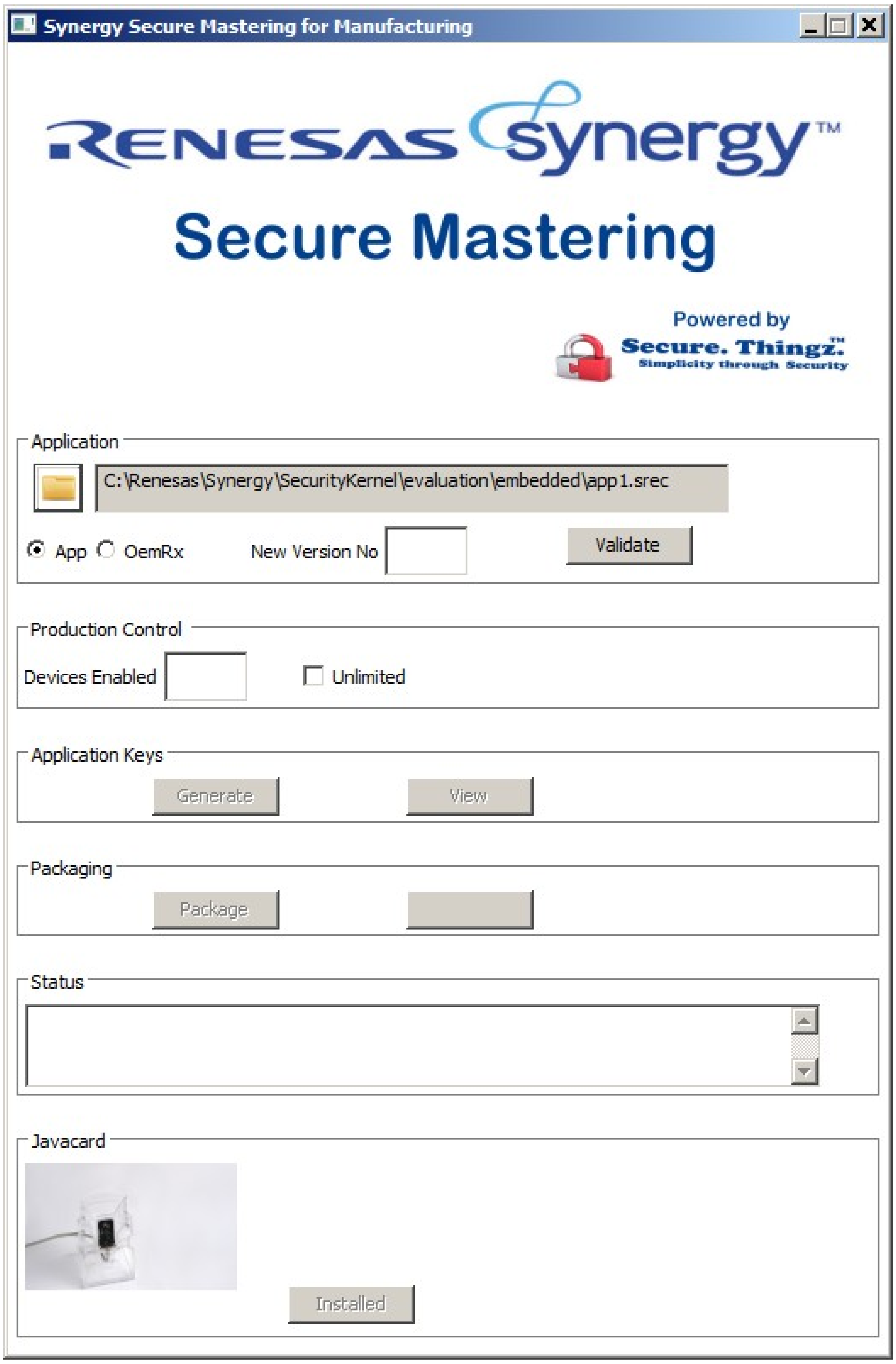


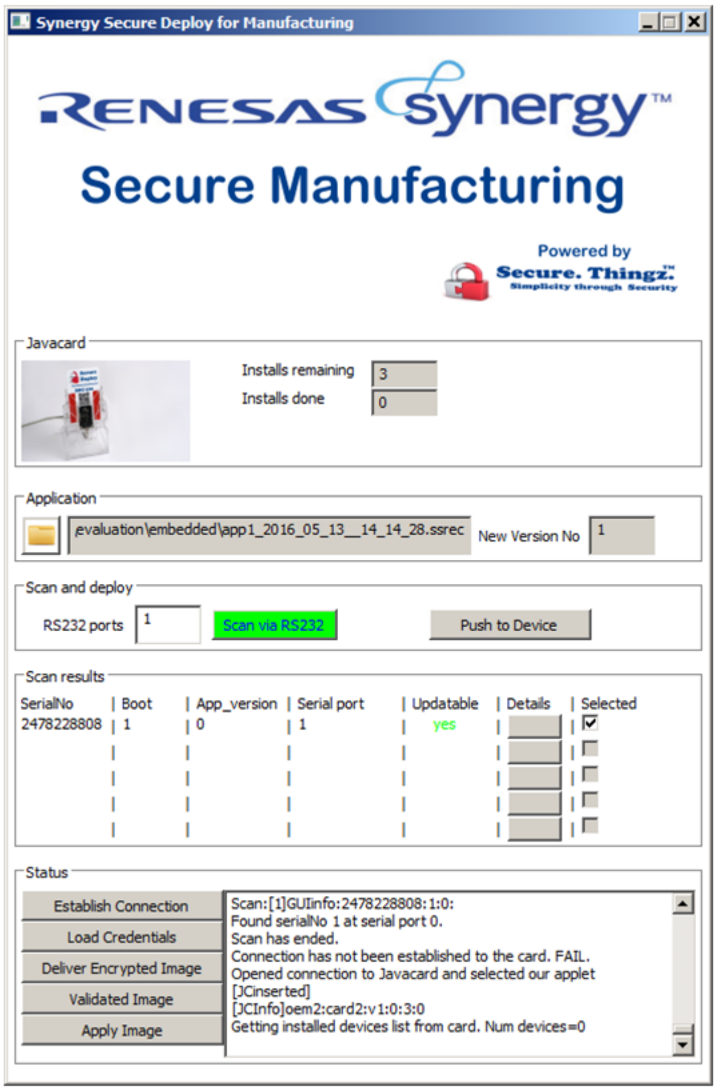
Figure 6 Secure Mastering Window

1. Place the Renesas Synergy Security Evaluation smart card into the smart card reader.
2. Click on the application folder and browse to **C:\Renesas\Synergy\SecurityKernel\evaluation\embedded** and select app1.srec. This is the example application to be packaged.

3. Check the App button (and not OemRX).
4. Select the New version number to be 1.
5. Click ‘Validate’ to check the .srec file parses correctly and the Key Database is consistent with a new version number of 1.
6. Select the number of devices to be enabled as 3. (This controls the maximum number of devices that can be manufactured on this run.
7. Click ‘Generate’ and then ‘Package’. This will generate the encryption and signature keys, encrypt and sign the application binary and write it out as app1_YYYY_MM_DD_HH_mm_ss.ssrec (proprietary srecord encrypted binary format). The key material and Device Enable count are stored securely on the smart card.
8. Clicking the ‘View’ button after ‘Generate’ will show the AES and HMAC keys that will be used to encrypt and generate HMAC of the binary.
9. Clicking the ‘Complete’ button once it has turned green will display the log file in a separate window.

3.6.2 Programming of Signed Application

Run **C:\Renesas\SynergySecureKernel\evaluation\gui\deployManufacturingGUI.exe** to bring up the window in Figure 7.


Synergy Secure Deploy for Manufacturing
RENESAS Synergy™
Secure Manufacturing
Powered by
Secure. Thingz.
Simplicity through Security

Javacard
Installs remaining: 3
Installs done: 0

Application
evaluation\embedded\app1_2016_05_13__14_14_28.ssrec
New Version No: 1

Scan and deploy
RS232 ports: 1

Scan results

SerialNo	Boot	App_version	Serial port	Updatable	Details	Selected
2478228808	1	0	1	yes		☑
						☐
						☐
						☐
						☐

Status

```
Scan:[1]GUIinfo:2478228808:1:0:
Found serialNo 1 at serial port 0.
Scan has ended.
Connection has not been established to the card. FAIL.
Opened connection to Javacard and selected our applet
[JCinserted]
[JCInfo]oem2:card2:v1:0:3:0
Getting installed devices list from card. Num devices=0
```

Figure 7 Signed Application Window

1. Transfer the smart card from the first reader to the second reader.
2. The image of the smart card reader in the Manufacturing tool will change to include the smart card.
3. The ‘Installs remaining’ box will change to 3 (or whatever number was selected during the mastering phase). The ‘Installs done’ box will read 0.
4. Enter the COM port number of the USB to Serial TTL converter adapter connected to SCI-2 into the RS232 ports box. (1 indicates COM1 in the screenshot above).
5. Click the ‘Scan via RS232’ button. In the example above, a board with device serial number 2478228808 has been found on Serial port 1. It has version 1 of the Security Kernel installed. App version = 0 indicates that no application is installed. The Updateable field is a green ‘yes’ indicating the application can be installed.
6. Check the ‘Selected’ box for this board and then click ‘Push to Device’ and the process of deploying the software will commence:
   - ✓ The “Establish Connection” box will go green.
   - ✓ A green bar will progress across the “Load Manufacturing Credentials” box as the UPK is downloaded to the board. It will then become solid green.

- ✓ The “Load Encrypted Image” button will go green as the encrypted srecord file is loaded into the GUI.
- ✓ A green bar will progress across the “Validate and Apply Image” button as the application is sent to the board.
- ✓ The device selection button will become green and the “Scan Results” group will be updated to App_Version 1 and the Updateable field with be set to a red ‘no’. (A version 1 application can’t be ‘updated’ with itself.)
- ✓ The “Installs remaining” field will decrement every time a device is successfully deployed to.
- ✓ The Installs done’ field will increment

3.6.3 Secure Boot of Device

At the end of device programming, the board will reboot and perform a Secure Boot on restart. The application App1 will be validated for integrity and authenticity before it is allowed to run.

The LCD Display will indicate that ‘Synergy Security Kernel demo app 1 (default version0)’ is installed along with some status information.

This development version of the Security Kernel has debug enabled and provides debug output on SCI8 (connected through breakout header). Connect a terminal program at 115200 baud 8N1 to view the diagnostic output from the kernel.

3.7 Secure Update

The Secure Update walkthrough consists of 2 stages:

1. The Update Mastering tool will be used to package a pre-generated application binary which is an update to the application installed on the device. Control is provided for handling rollback permission. The encrypted and signed update and key material is made available to the Update Server.
2. The Update Server tool will be used to install the update onto the development board.

3.7.1 Preparation of Signed Update

Run **C:\Renesas\Synergy\SecureKernel\evaluation\gui\masterUpdateGUI.exe** to bring up the window in Figure 8

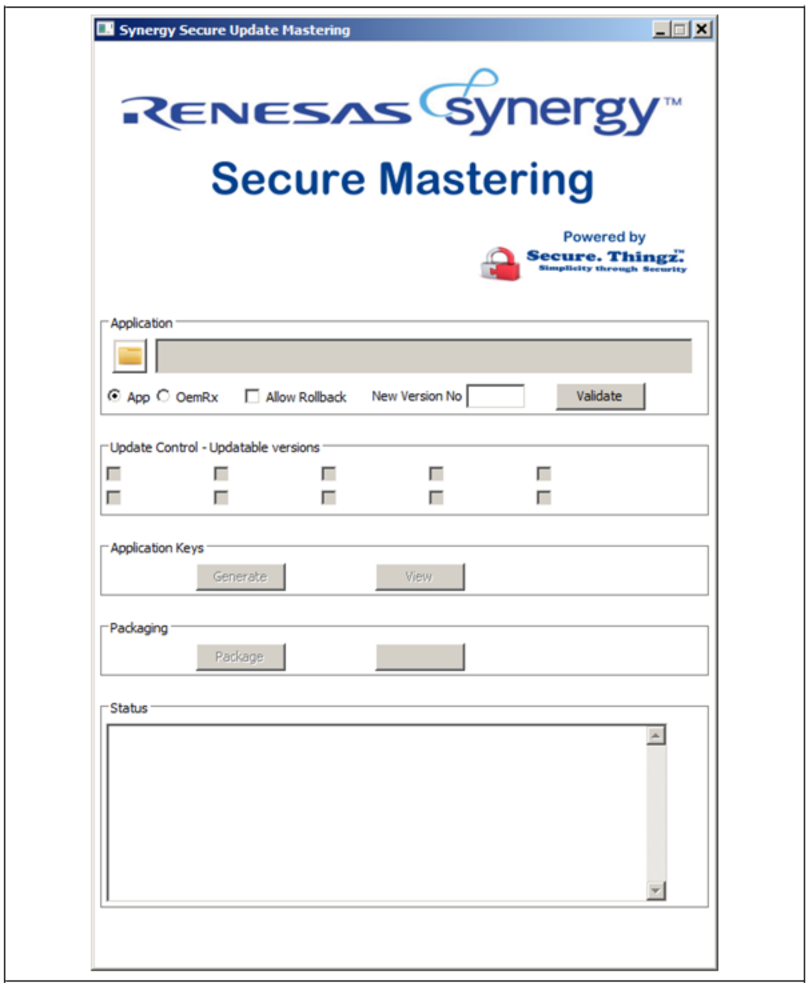
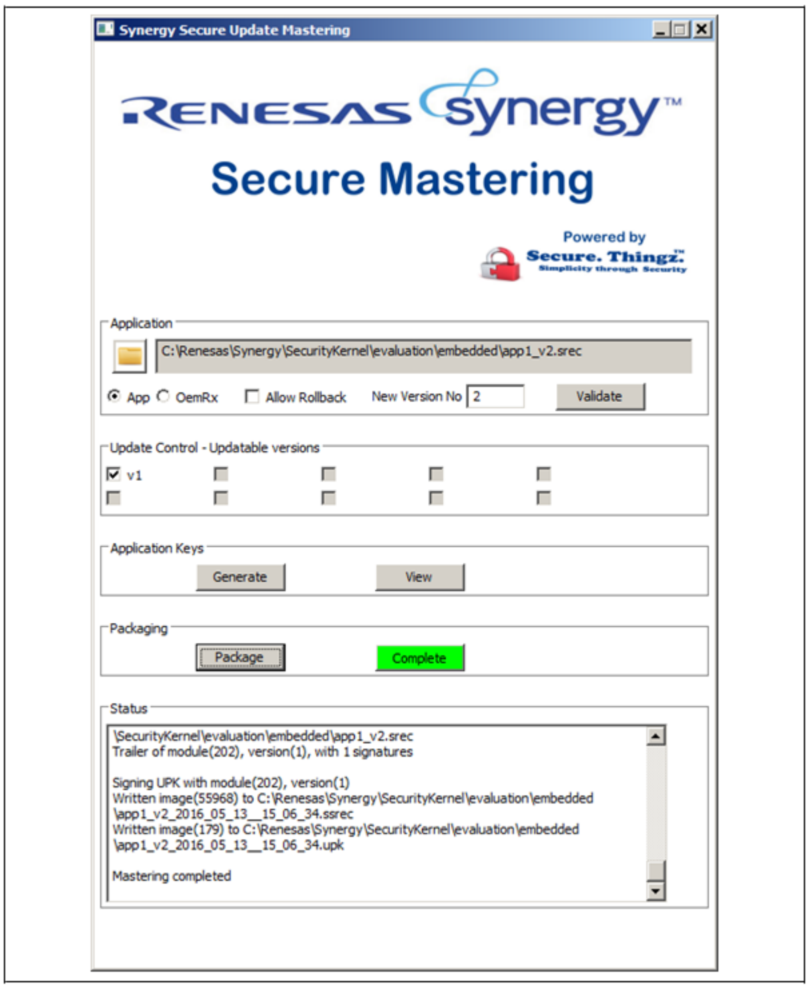


Figure 8 Signed Update Preparation Window

1. Click the application folder and browse to **C:\Renesas\Synergy\SecurityKernel\evaluation\embedded** and select App1_v2.srec. This is the example application update to be packaged – a slight variant on aApp1.
2. Select ‘App’ rather than ‘OemRX’.
3. Do NOT check ‘Allow Rollback’ at this time. For this version of the walk-through, strict control of updates is applied. Only specified versions of the app can be updated to the new version. No rollback to previous versions will be allowed.
4. Set New Version number to 2 and select ‘Validate’. This checks that the .srec file can be parsed and that Update 2 has not already been used in the Key database. If the ‘Validate’ button does not appear, please check the Windows Display setting in the Control Panel. Make sure that **Smaller - 100%(default)** is selected
5. The Update Control – Updateable versions field will be populated only with a v1 checkbox. This is the only existing version installed in the key database.
6. Check the v1 box.
7. Click ‘Generate’ and then ‘Package’. This will generate the encryption and signature keys, encrypt and sign the application binary and write it out as app1_v2_YYYY_MM_DD_HH_mm_ss.ssrec (proprietary srecord encrypted binary format).

As part of the evaluation, the encrypted binary and the key material are made directly available to the update server. In a real system, the key material must be passed to the update server in a secure manner.

The final window should display as shown in Figure 9.



Application: C:\Renesas\Synergy\SecurityKernel\evaluation\embedded\app1_v2.srec

[✓ App] OemRx ☐ Allow Rollback New Version No 2

Update Control - Updatable versions: ☑ v1

Status:

```
\SecurityKernel\evaluation\embedded\app1_v2.srec
Trailer of module(202), version(1), with 1 signatures

Signing UPK with module(202), version(1)
Written image(55968) to C:\Renesas\Synergy\SecurityKernel\evaluation\embedded
\app1_v2_2016_05_13__15_06_34.ssrec
Written image(179) to C:\Renesas\Synergy\SecurityKernel\evaluation\embedded
\app1_v2_2016_05_13__15_06_34.upk

Mastering completed
```

Figure 9 Secure Update Complete.

3.7.2 Installation of Signed Update

Run **C:\Renesas\Synergy\SecurityKernel\evaluation\gui\deployUpdateGUI.exe** to bring up the window in Figure 10.

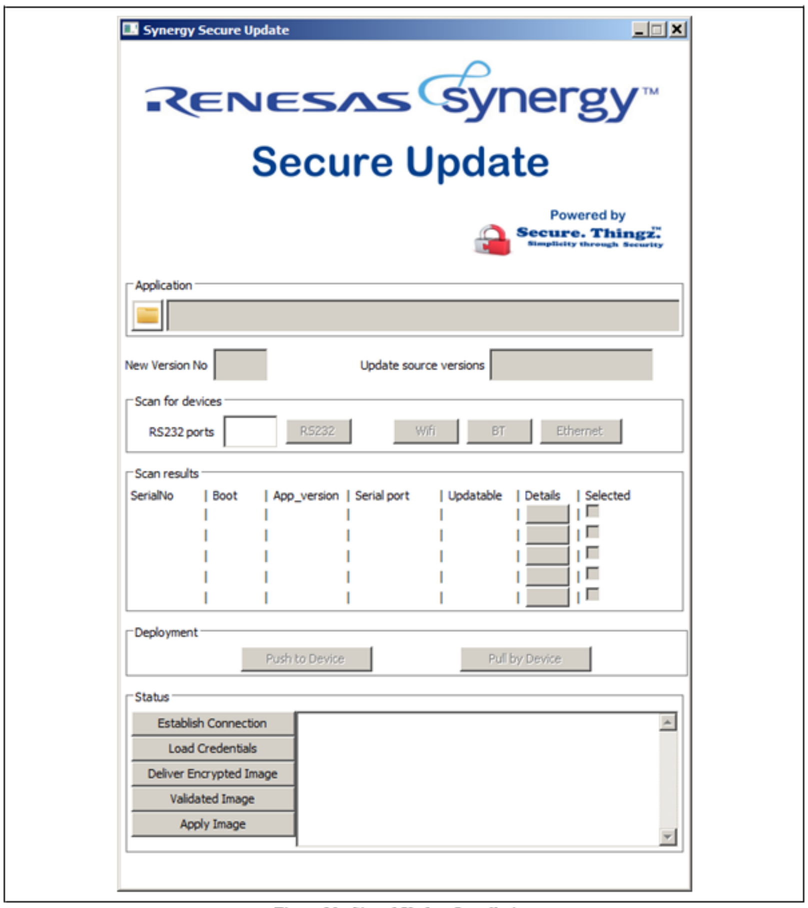


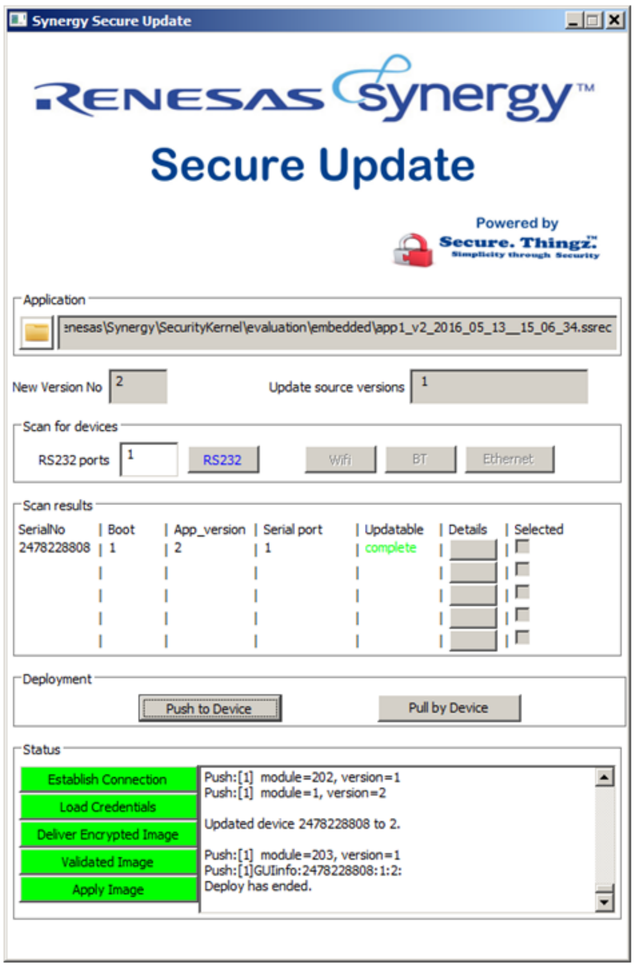
Figure 10 Signed Update Installation

Click the folder and select

C:\Renesas\Synergy\SecurityKernel\evaluation\embedded\app1_v2_YYYY_MM_DD_HH_mm_ss.ssrec.

1. The new version number is extracted from the .ssrec file and set to 2.
2. The Update Source versions field is extracted from the file. In this case it is set to 1. This is the only version of the application that can be updated to version 2.
3. Set RS232 Ports to the number of the COM port assigned to the USB-to-Serial adapter attached to the board.
4. Select RS232 as the scan mode. The update server will check for valid Synergy devices connected to the Update server and list them in the scan results table.
5. One board should be found with App_version 1 and the Updateable field set to a green 'yes'.
6. The Update Source versions data is informational. It cannot be manipulated or overridden by the update server or during the update process. The device on the evaluation board will only decrypt and install the update if it is one of the versions listed. If an attempt is made to apply the update to a board with the wrong version, then the device Security Kernel will refuse to install the update based on the key material it already holds and the signatures on the update.
7. Select the device to Update.
8. Select 'Push to Device' button.
9. The Update will be deployed to the device

   The display should look as follows



Application: :nesas\Synergy\SecurityKernel\evaluation\embedded\app1_v2_2016_05_13__15_06_34.ssrec

New Version No: 2

Update source versions: 1

RS232 ports: 1

SerialNo	Boot	App_version	Serial port	Updatable	Details	Selected
2478228808	1	2	1	complete		☐

```
Push:[1]  module=202, version=1
Push:[1]  module=1, version=2

Updated device 2478228808 to 2.

Push:[1]  module=203, version=1
Push:[1]GUIinfo:2478228808:1:2:
Deploy has ended.
```

Figure 11 Signed Update Complete

3.7.3 Device Reboot

Upon completion of the update installation, the device reboots. The secure boot code checks the integrity and authenticity of the application. If it is correct, it will reboot the device. The Green LED and Red LED will blink alternatively.

3.7.4 Evaluation Walk-Through So Far

The walk through has achieved the following steps:

1. Packaged App1 for Installation on the development board as version 1 of the App
2. Deployed App1 to the development board using the Secure Manufacturing Process
3. Packaged App1_v2 for update ready for the development board as Version 2 of the App
4. Updated the development board to App1 version 2 using the Secure Update Process

3.8 Next Steps

There are a range of possible extensions to the walk-through

3.8.1 Reset the Evaluation

Because of the Secure Manufacturing and Secure Update processes, it is not possible to simply re-run the Secure Manufacturing process on a board that an application already installed. A script has been provided for the evaluation that uses the update process to remove the application installed on the board and reset the key database held on the PC. This allows the walk-through to be re-run.

1. Open a command window in C:\Renesas\Synergy\SecurityKernel\evaluation\resetEvaluation
2. Run resetEvaluation.bat <COMportNo> where <COMportNo > is the USB-to-Serial adapter port number

3.8.2 Package and Apply Further Updates

At the end of the walk-through, you can apply further updates to App1_v3 and App1_v4. The srec files are also found in

C:\Renesas\Synergy\SecurityKernel\evaluation\embedded\bin

a) Do not check the 'Allow Rollback' box.

b) Check the boxes of all previous versions installed.

3.8.3 Attempt to Break the Rollback Rules

Attempting to re-install version 2 of App1 onto the board which has been updated to version 3 or version 4 will fail. The tool will not permit this as this is attempting to 'rollback the device'. Lower level mechanisms could be used to attempt to re-install the previous version using the underlying commands, but the SecurityKernel will refuse to install the update as the update information doesn't match its internal key material.

3.8.4 Package and Apply Updates that Allow Rollback

The rollback control of the underlying update mechanism is very flexible. For the Evaluation system, only 2 methods are available.

a) No rollback Permitted: Downgrades to a previous version number is not permitted. This is the default packaging.

b) Rollback allowed: Any mastered version can be installed over any other version.

This is achieved by ticking the 'Allow Rollback' box on the Secure Update Mastering tool for every update version that is mastered. This can be demonstrated by Updating to version 3 and then Updating to version 2 as long as both versions are mastered with 'Allow rollback'.

3.8.5 Add an OEM Download Receiver

As well as installing the application binary during Secure Manufacture, it is also possible to install an OEM Download Receiver. If the application becomes corrupted or an update fails partway through the update process, then the OEM download Receiver can be used to recover the device. An example OEM Download Receiver is provided as **C:\Renesas\Synergy\SecureKernel\evaluation\embedded\OemRx.srec**.

The OemRx will be invoked if installed and at boot time the SecurityKernel cannot locate a valid application image (correctly signed and authenticated). This example OemRx uses SCI-2 serial port to receive an application/update onto the device.

To install the OemRx and App1:

1. Reset the evaluation environment with resetEvaluation.bat
2. Follow the Secure mastering walk-through and to the point where App1 is installed on the device
3. Repeat the mastering steps, but selecting **C:\Renesas\Synergy\SecureKernel\evaluation\embedded\OemRx.srec**

   as the binary to be mastered. Select OemRx as the type (rather than App).
4. Repeat the Secure Manufacturing stage but selecting

   C:\Renesas\Synergy\SecureKernel\evaluation\embedded\OemRx.ssrec

Both OemRx and App1 are now installed on the device.

For evaluation purposes, the Security Kernel will run the OemRx binary at boot time if button 2 is held down while the board is being reset to mimic a failed update/corrupted application image.

3.9 Troubleshooting

3.9.1 Mastering for Secure Manufacture

Two common problems that occur are:

- The walk-through cannot be re-run without resetting the applications installed on the board and the key database stored on the PC. When packaging is attempted for a second time, the red FAILED button appears. The status window indicates 'failed to make new keys for add of module….'
  1. Open a command window in C:\Renesas\Synergy\SecurityKernel\evaluation\resetEvaluation
  2. Run resetEvaluation.bat <COMportNo>
  3. Where <COMportNo > is the USB-to-Serial converter adapter port number
  4. Restart the mastering process.
- The JAVA card is not inserted correctly or inserted into the wrong smart card reader. When Packaging is attempted, a red FAILED button appears. The status window indicates that the Javacard cannot be accessed.
  1. Insert the smart card correctly or into the correct reader. The Package button can be checked again directly. (There is no need to 'Validate' or 'Generate' again.)

4.Embedded Component Description

For the Evaluation system, the Program Flash address map is allocated as in Figure 12

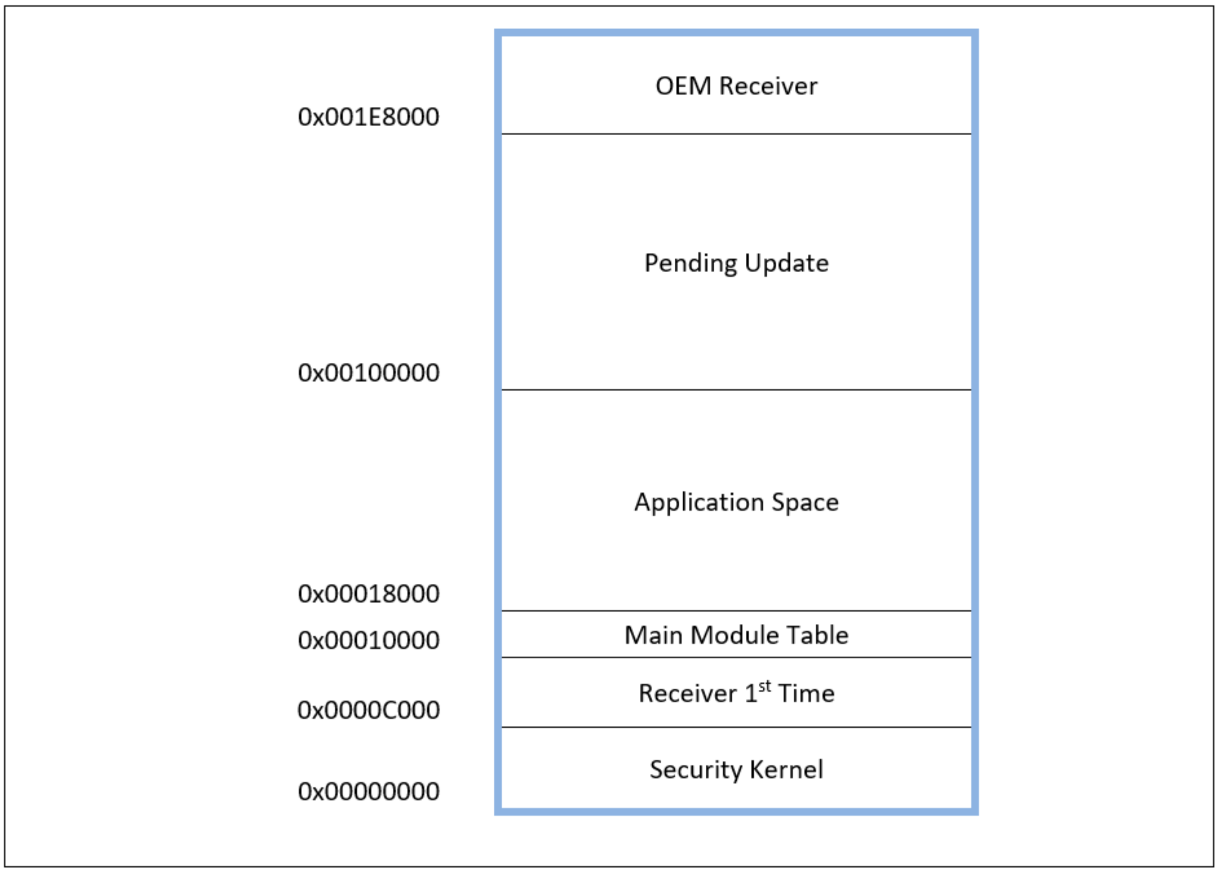


Figure 12 Program Flash Address Map

The firmware is divided into a number of modules. A module is simply a normal fully linked block of code with additional information appended which must be located at a particular location in program flash. The information that is appended is an ECDSA P256 signature and the Module Update Key.

For the Evaluation, the modules used are:

- Security Kernel
- Receiver 1st Time
- OEM Receiver
- Application firmware

(The Application firmware can be a number of modules, but the mastering tool GUI only allows one application module at this time.)

The Receiver 1st Time module downloads the application firmware onto the device during manufacture. This uses the SCI-2 serial port.

The OEM Receiver is an optional module that can be used to recover the device if for some reason an update fails or the application or main module table becomes corrupt. The example OEM Receiver also uses the SCI-2 serial port.

The Main Module Table contains the information necessary for the Security Kernel to validate the integrity of each of the Application modules installed on the device.

The main module table serves the following purposes:

- To validate the integrity of modules
- To identify the locations of modules and their entry points
- To authenticate updates to the modules

To be a root of trust for the addition of new modules

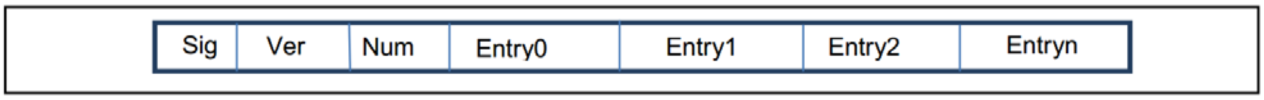


Figure 13 Module Table Format

Sig is the ECDSA P256 signature on the whole of the module table which is maintained by the Security Kernel. Whenever the table is updated the signature is also updated.

Ver is the Module Table version number.

Num is the number of entries in the module table (set to 25 for the evaluation).

Entry0 … Entryn are individual module entries.

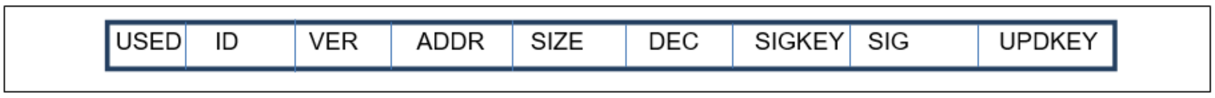


Figure 14 Module Table Entry

USED indicates that the entry is used.

ID is the ID of the module.

VER is the current version of the module installed.

ADDR is the flash address of the start of the module.

SIZE is the size in bytes of the module.

DEC indicates if the module is encrypted and needs to be decrypted before use (not supported in evaluation).

SIGKEY is the ECDSA P256 public key used to verify the signature of the module.

SIG is the ECDSA P256 signature of the module.

UPDKEY is the Module Update Key - ECDSA P256 key which must be used to sign updates to this module.

4.1 Security Kernel

The Security Kernel resides at the reset vector. At device boot, it checks for a pending update. If an update is required, then the update is installed and the device rebooted. If no update is required, the kernel checks the validity of the Main Module table by checking the table signature, then the validity of each module table entry by checking the signature on each module. If successful, the execution jumps to the application.

If the checks fail, then the security kernel checks the OEM Receiver signature and invokes that if successful.

If that check fails, the Security kernel checks the Receiver 1st Time signature and invokes that if successful.

If all the signatures fail, the device is usually bricked (rendered nonfunctional). For the purposes of evaluation, the kernel has some backup mechanisms to avoid device bricking.

If button 1 is held down during reset, execution will jump to the Receiver 1st Time (if valid) and the Main Module Table will be reset.

If button 2 is held down during reset, execution will jump to the OEM Receiver (if valid). Security Kernel Debug information is output on serial console.

4.1.1 Security Kernel API

The Security kernel API consists of only 3 calls which can be invoked by the application or Receiver 1st time or OEM Receiver.

4.1.1.1 Interface to obtain the contents of the device certificate

extern bool securityKernelApiGetDevCert(uint8_t *pBuffer, uint32_t *pLength);

input - pBuffer is a pointer to an empty buffer to *pLength bytes. output - pointer to the devCert.

If return value == true

pBuffer is a pointer to a buffer of *pLength bytes containing the device certificate. Else

*pLength = requiredLengthForDevCert -- Application needs to re-call the API with a pointer to a buffer of sufficient size.

4.1.1.2 Interface to validate and (if valid) apply an update

Prior to calling this function the application must have shutdown, storing all relevant context to persistent memory and put all peripherals into a safe state.

This function will never return. If valid, the update will be applied. The device will always reboot.

extern void securityKernelApiApplyUpdate(uint8_t *pBuffer, uint32_t length);

4.1.1.3 Interface to obtain the version numbers of all the installed modules

extern bool securityKernelApiGetModuleVersionList(uint8_t *pQuadAlignedBuffer, uint32_t *pLength);

If return value == true

pBuffer is a pointer to a buffer of *pLength bytes containing module version info.. Else

*pLength = requiredLengthForModuleVersionInfo. -- Application needs to re-call the API with a pointer to a buffer of sufficient size.

4.1.2 Device certificate Format

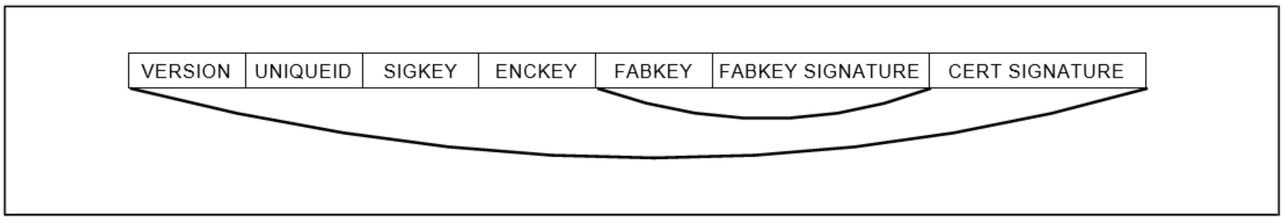


Figure 15 Device Certificate Format

VERSION is the Certificate Version number.

UNIQUEID is unique ID for the device/Certificate (16 bytes).

SIGKEY is the Device Signature Key: the ECDSA P256 public key of a device-specific signature key derived from the Device Seed using a label of “DevSign”. This is used to sign data to prove it originated from this particular device.

ENCKEY is the Device Encryption Key: the ECIES P256 public key of a device-specific ECIES encryption key derived from the Device Seed using a label of “DevEnc”. This is used to allow information such as the encryption keys used to protect modules for installation to be delivered securely to this particular device.

FABKEY is the ECDSA P256 public key of a FAB specific signature key owned by Renesas. This has been used to sign the certificate.

FABKEY SIGNATURE is the ECDSA P256 signature of the FAB specific public key using OEM root signature key. This delegates authority from the OEM root key to the preceding key to sign in its name.

CERT SIGNATURE is the ECDSA P256 signature over the preceding values generated using the FAB public key included in the signature.

The Device Certificate is generated when the Security Kernel is installed on the device. The certificate is tied to the Device Seed.

The Device Seed is generated and the Public parts of the Device Signature Key and Device Encryption Key are generated using the Key derivation function from the seed.

These keys along with a device unique ID are included in the certificate together with the public part of the FAB key (and the FAB key signature signed by the OEM Root Key). The Certificate is signed with the FAB Key and then installed on the device along with the Device Seed and Security Kernel.

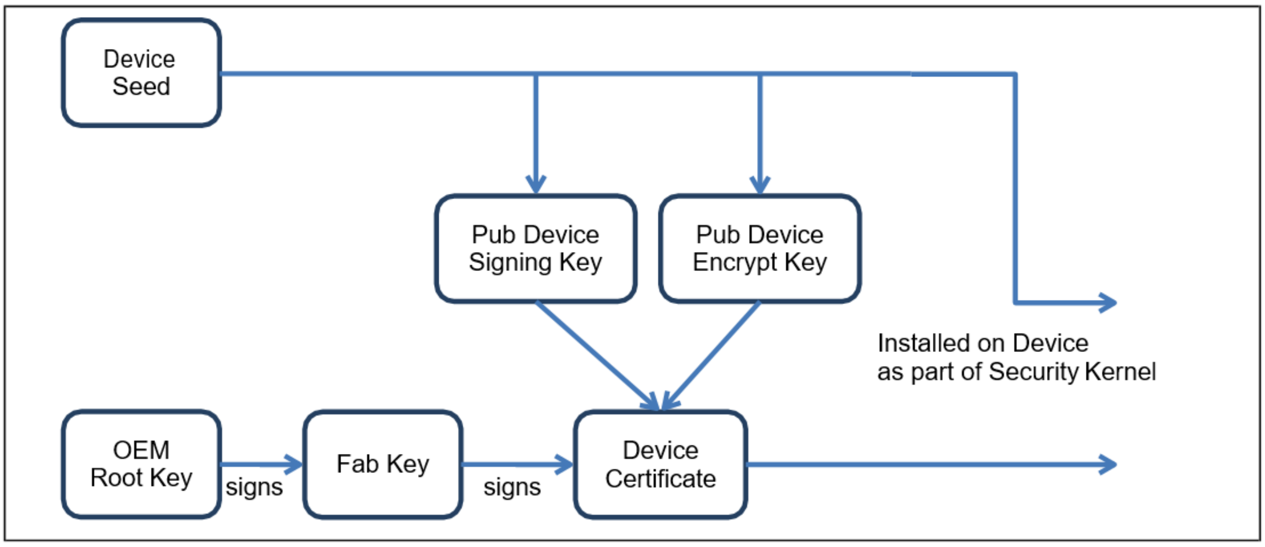


Figure 16 Use of OEM Root Key and Fab Key

4.2 Receiver 1st Time

The Receiver 1st Time module is the mechanism to load the encrypted application binary to the device during Secure Manufacturing. SCI-2 serial port is used to interface to the Secure Manufacturing process.

The Encrypted application binary and key material are loaded into the Pending Update address space. Then securityKernelApiApplyUpdate() is called.

For evaluation purposes, the Receiver 1st Time is located outside the Flash access window to avoid accidental deletion or corruption. This allows the evaluation device to be reset and the secure manufacturing process to be re-initialized. In a production system, the Receiver 1st Time could be removed or overwritten during the installation of the application firmware.

4.3 Main Module Table

The Module table is only updated when modules are installed and un-installed so it is not written so frequently as to wear out the flash.

Each Module table entry contains at least the following information:

- Module Address in program Flash or 0x00000000 if no module at this index
- Module size in bytes
- ECDSA P256 public key which is used to verify the signature on the module
- ECDSA P256 public key which must be used to sign updates to this module - the Module Update Key
- Pointer to location in memory that module should be decrypted to or 0 if unencrypted.

There is an ECDSA P256 signature on the whole of the module table which is maintained by the Security Kernel. Whenever the table is updated the signature is also updated.

4.4 OEM Receiver

The OEM Receiver is an example application for recovering the device if the main application becomes corrupted or the module tables become corrupted during an update. The OEM receiver uses SCI-2 serial port to receive a valid update for the device. The process of installing an update is the same as a normal update.

1. Return the device certificate to the update server.
2. Return the module table info to the update server.
3. Update server uses the device public encryption key to encrypt the update protection key.
4. Update server sends the encrypted Update and the encrypted update protection key to the device. Stored in the Pending Update address space (0x0010 0000–0x001E 7FFF).
5. Call to the Security Kernel API to install the update.

Any transport mechanism can be used to load the encrypted update and the Encrypted Update protection key into the Pending Update address space. For simplicity, the example provided uses the SCI-2 serial port. This could be Wi-Fi, Bluetooth, or Ethernet. The channel itself doesn’t need to be protected as the update and the Update Protection key are both encrypted.

5.Evaluation Tools

5.1 Secure Mastering

The Secure Mastering tool takes an application binary and processes it so that it can be loaded securely onto a device with Security Kernel during the Secure Manufacturing process. Loading an application onto a new device is just a special version of an update to a device. For this evaluation, the mastering tool allows only one module with module ID 1 to be installed on a device. This is a simplification enforced by the GUI. The underlying mechanism supports a complex install/update mechanism which allows the install of multiple modules each at different versions.

For this evaluation, the Update passed to the device consists of 3 update operations:

1. Update Header Operation signed by the Security Kernel Module Update Key
2. Install Module Operation for Module 1 (with the selected version no.) signed by the Security Kernel Module Update Key
3. Update Trailer Operations signed by the Security Kernel Module Update key

The Whole Update is protected using an Update Protection Key.

The Update Protection Key (UPK) consists of randomly generated 128-bit AES key with 128-bit IV and 256-bit HMAC key. These are used to encrypt and sign the Update.

The UPK is also signed using the Module Update Key of the Security Kernel.

The UPK is stored on the Smart card together with the Number of devices enabled for programming.

5.1.1 UPK Format

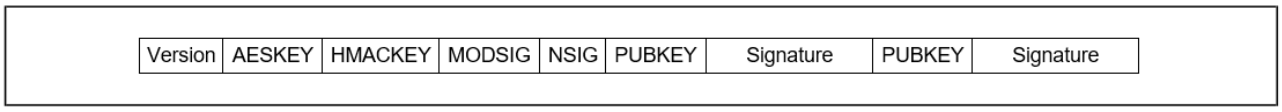


Figure 17 Format of UPK

Version is the version number of the message.

AESKEY is the AES key to use to decrypt the Update.

HMACKEY is the HMAC Key to use to verify the Update.

MODSIG is the Module ID of the module used to sign the Update with its Module Update Key(s).

NSIG shows how many public key/signature pairs follow. In each case, the Module Update Key (PUBKEY) of a version of the module authorizing the update is followed by an ECDSA P256 signature using that key (Signature). Each signature is made over the VERSION, AESKEY, HMACKEY, MODSIG and NSIG fields of this structure.

When mastering an initial image, then NSIG is 1 and the Module ID used is the Module ID of the Security Kernel (202)

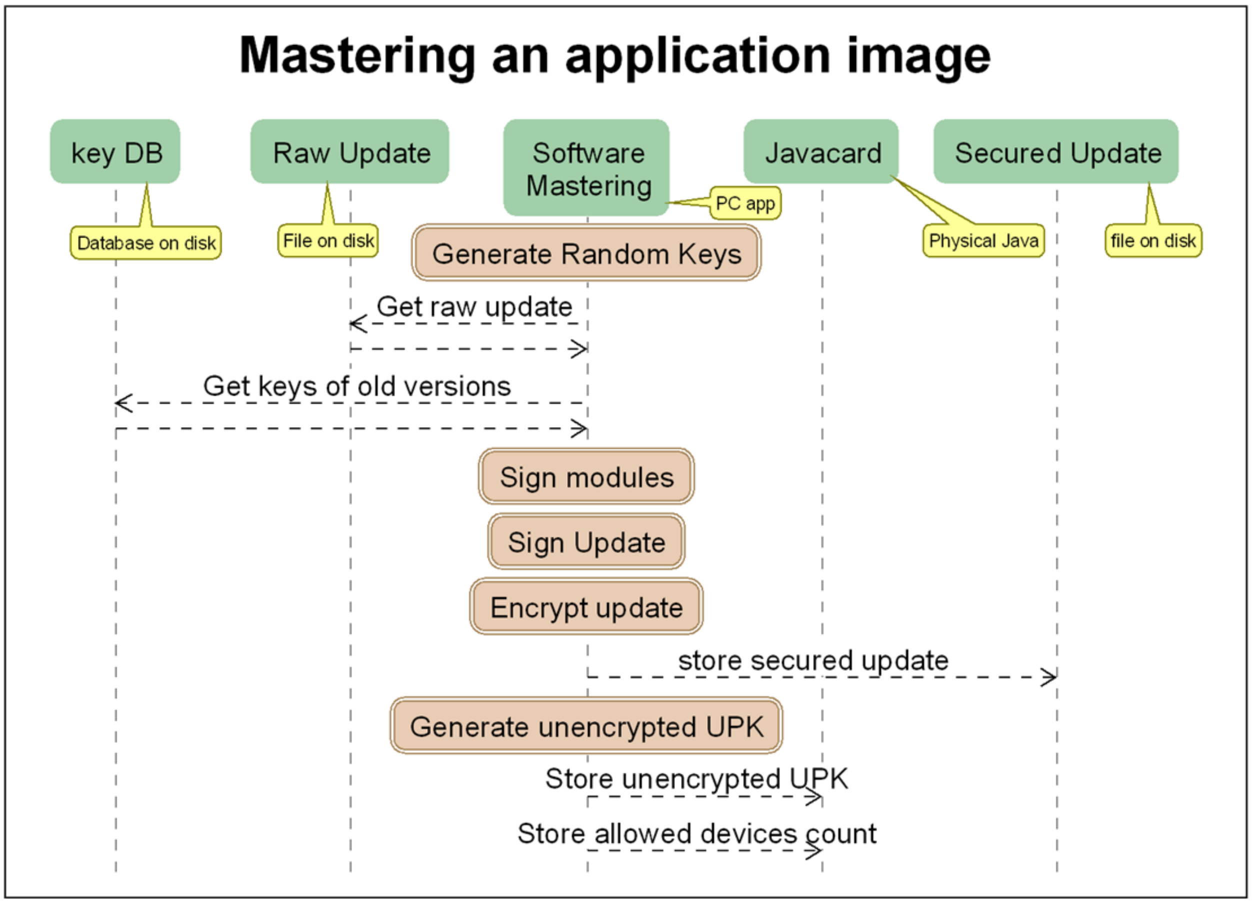


Figure 18 Mastering an Application Image

In the case of mastering an image for the first time, the key Database is populated with new keys for the module (and version being mastered).

5.2 Secure Manufacturing

During Secure Manufacturing, the following steps occur.

1. Select the Encrypted application binary to be installed (.ssrec file)
2. The UPK is already stored on the Smart card along with production count limits
3. Scan Serial Ports of PC for valid Synergy Devices
4. Select Device to be programmed
5. Program the device which does the following:
6. Read the Device Certificate and send to the Javacard
7. Javacard validates the device certificate using the Fab public key
8. Javacard encrypts the UPK using the Device public encryption key (from the device certificate) and passes back to programmer. The production count is incremented.
9. The programmer attaches the encrypted UPK to the secured update and sends to the device
10. The Security Kernel is called to install the Update

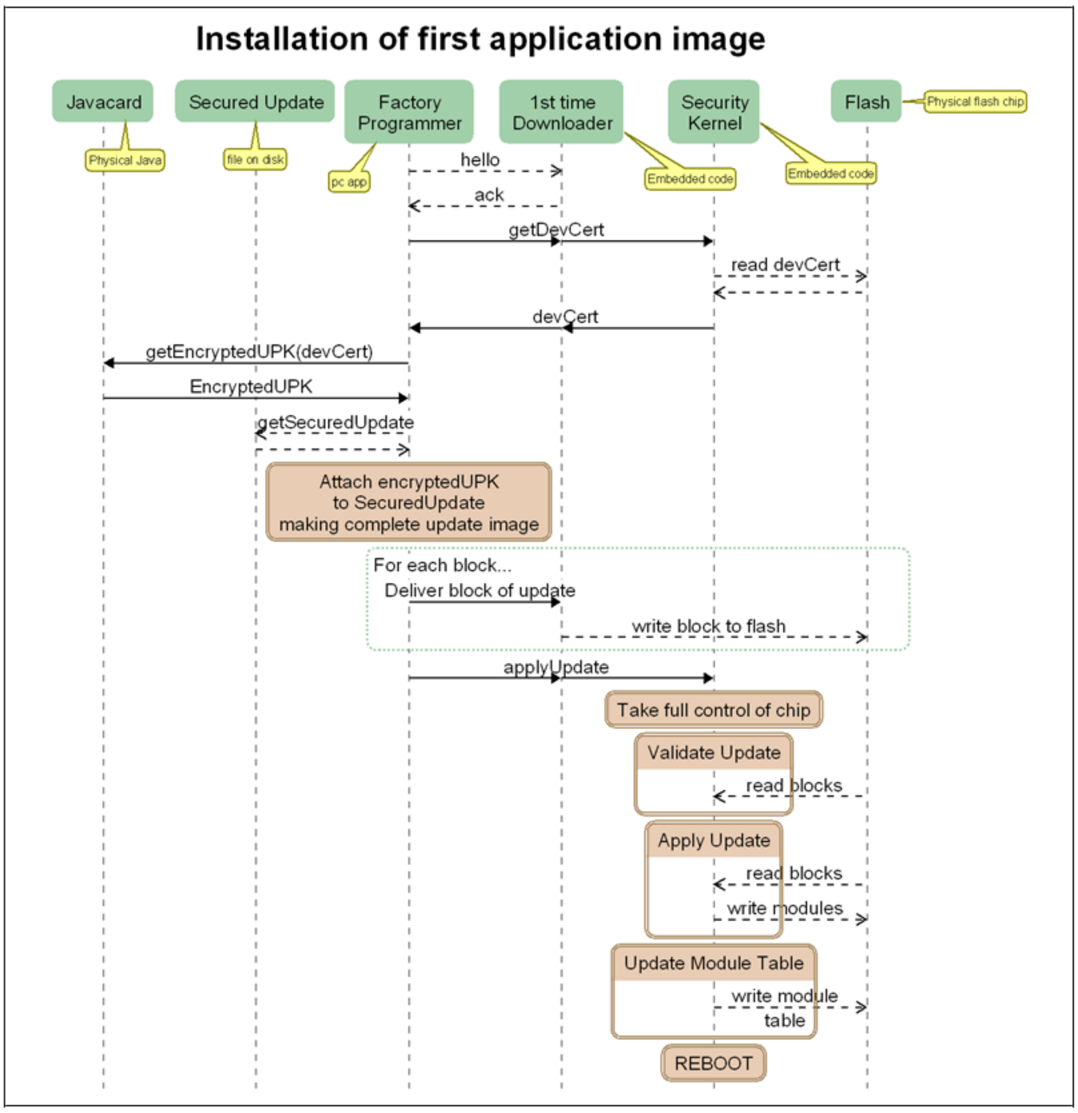
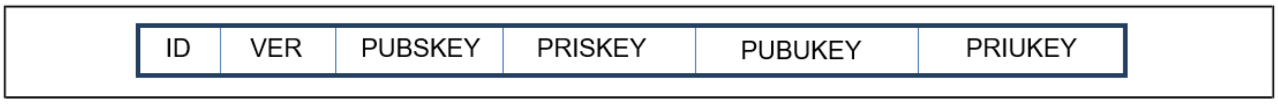


Figure 19 Installation of First Application Image

5.3 Key Database

The Key Database stores all the encryption and signing keys used during the mastering phases of the evaluation. The database is stored in text format for ease of understanding, both public and private parts of all keys are stored in the database. This includes the dummy OEM Root key and Fab key. In a real system, only the public part of these keys would be available.

The database format is shown in Figure 20.



ID	VER	PUBSKEY	PRISKEY	PUBUKEY	PRIUKEY

Figure 20 Key Database Format

ID is the ID of the Module or Key Handle as described in Table 1.

VER is the version number of the database

PUBSKEY is the public part of the Signing Key associated with the module

PRISKEY is the Private part of the Signing Key

PUBUKEY is the public part of the Update Key associated with the Module

PRIUKEY is the private part of the Update Key

Table 1 Module or Key Handle IDs

Module ID	Usage
1…127	Available for use by Application
128…191	Used by OEM Recovery Receiver Modules
192	OEM Root Key
193	Fab Key
194	Reserved
195	Reserved
202	Security Kernel Key
203	Receiver 1st Time Key

SHA256 Hash is stored at the end of the database to check the integrity of the keys and is updated whenever a new key is added to the database.

See section 4.1.2 for information on the use of the OEM Root Key and the Fab key during the generation of device certificates.

5.4 Update Mastering

The difference between mastering for updates and mastering an application for initial install is about the control of rollback. If no rollback protection is required, then all updates can be signed with the same Module Update Key stored in the key database.

If rollback protection is required, then an update to a new version will be signed using the Module Update Key of each previous version that it is required to update. The GUI allows selection of previous versions as stored in the key database.

For this evaluation, the mastering tool allows only the application with module ID 1 or the OEM Recovery Receiver to be updated on a device. This is a simplification enforced by the GUI. The underlying mechanism supports a complex install/update mechanism which allows the install of multiple modules each at different versions. In the case of no rollback protection for App1, the Update Operation consists of:

1. Update Module Operation for Module 1 signed by the Module Update Key for Module 1 version 1 taken from the key database.
2. The Update is protected using an Update Protection Key.
3. The Update Protection Key (UPK) consists of randomly generated 128-bit AES key with 128-bit IV and 256-bit HMAC key. These are used to encrypt and sign the Update.
4. The UPK is also signed using the Module Update Key of Module 1 version 1.
5. In the case of Rollback protection, a new Module Update Key is generated for the new version number and stored in the Key database.

The Update Operation consists of:

1. Update Module Operation for Module 1 signed by the Module Update Keys of all the Module 1 versions selected for update in the GUI. The Keys are taken from the key database.
2. The Update is protected using an Update Protection Key.

3. The Update Protection Key (UPK) consists of randomly generated 128-bit AES key with 128-bit IV and 256-bit HMAC key. These are used to encrypt and sign the Update.
4. The UPK is also signed using the Module Update Keys of all the Module 1 versions selected from Update in the GUI.

5.5 Secure Update

The Deploy Update program is very similar to the Secure Manufacturing program. The differences are that the UPK is stored locally on the PC rather than in the smart card and the UPK encryption is handled by the program itself.

The Deploy Update program connects over the SCI-2 serial port to the device, but in this case the application is running on the device. App1 is handling the transportation of the update, its installation into the Pending Update address space, and the calling of the security kernel to handle the update installation.

5.6 System Limitations

5.6.1 Debug

JTAG debug is enabled across the whole address map including the Security Kernel and the Device Seed.

5.6.2 Confidentiality

The Security MPU is set across the address space occupied by the Security Kernel and the Receiver 1st Time in order to protect the confidentiality of the Security Kernel and Device Seed.

5.6.3 Integrity

The Flash Access Window has been set on the devices when the Security Kernel is installed. This prevents accidental erasure of the Security Kernel and the Receiver 1st Time. Flash Write is only permitted above 0x0001 0000.

The FSPR bit has not been set for the evaluation set meaning that the Flash Access Window could be reset.

In a real system, the FSPR bit would be set, preventing the erasure of the Security Kernel.

5.6.4 OSF registers

The OSF registers are in the address space covered by the Security kernel and are outside the Flash Access window. The registers are not visible/writeable by the user. The values set in the Evaluation system Security Kernel are:

```
{{ // OFS0
    0x1,
    0x1, // IWDTSTRT    : 1;
    0x3, // IWDTTOPS    : 2;
    0xf, // IWDTCKS     : 4;
    0x3, // IWDTRPES    : 2;
    0x3, // IWDTRPSS    : 2;
    0x1, // IWDTRSTIRQS : 1;
    0x1,
    0x1, // IWDTSTPCTL  : 1;
    0x3,
    0x1, // WDTSTRT     : 1;
    0x3, // WDTTOPS     : 2;
    0xf, // WDTCKS      : 4;
    0x3, // WDTRPES     : 2;
    0x3, // WDTRPSS     : 2;
    0x1, // WDTRSTIRQS  : 1;
    0x1,
    0x1, // WDTSTPCTL   : 1;
    0x1
}},

{{// OFS1
    0x3, // VDSEL    : 2;
    0x1, // LVDAS    : 1;
    0x7, // VDSEL1   : 3;
    0x3,
    0x1, // HOCOEN   : 1;
    0x3, // HOCOFRQ0 : 2;
    0x1,
    0x7, // HOCOFRQ1 : 3;
    0x1ffff
}},
```

5.6.5 Cryptography

Random Numbers and Cryptographic Algorithms are currently not production ready.

5.7 Extension of App1 Example

App1 is an example application which can be installed on devices which include the security Kernel. App1 also includes the SCI-2 serial code to receive a new update to the device, load it into the Pending Update Address space and invoke the Security Kernel to install the update.

6.Program Flash Address Map

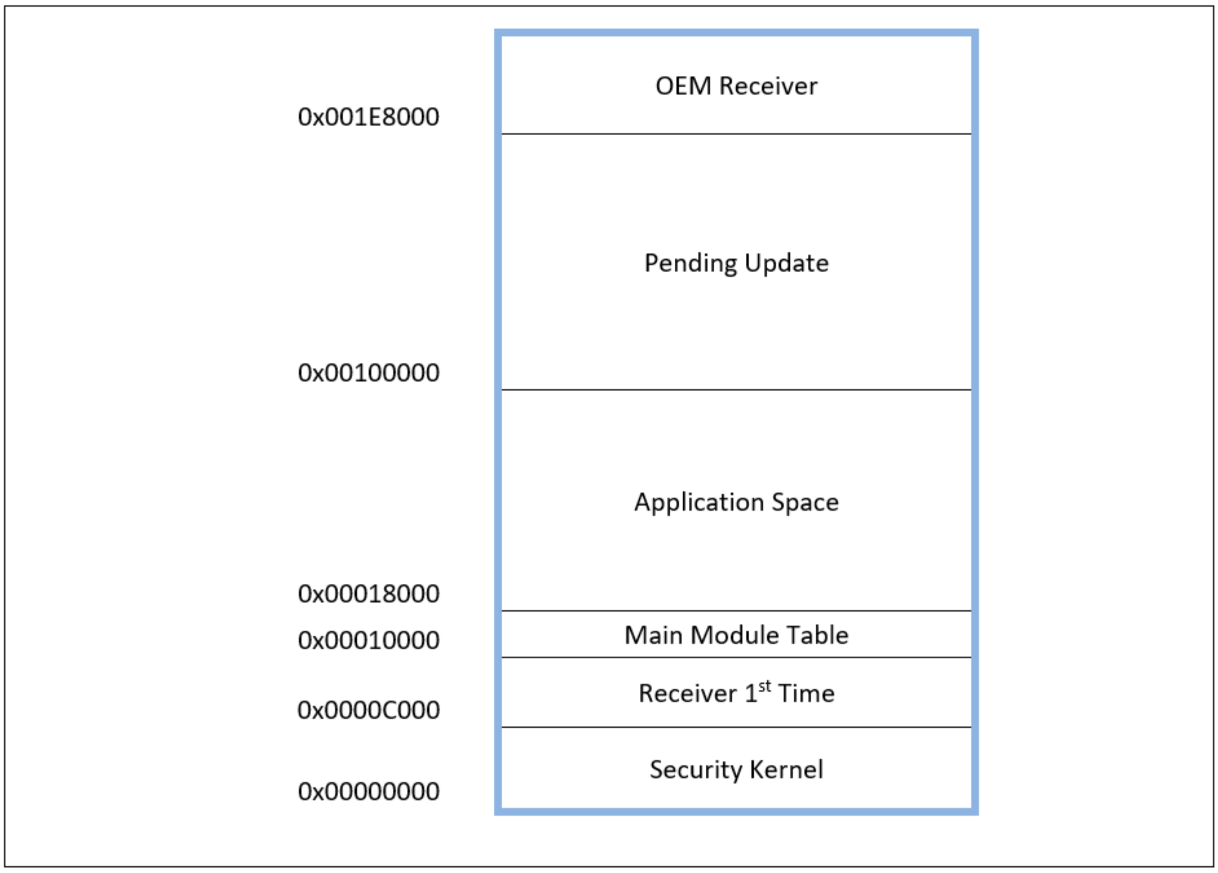


Figure 21 Program Flash Address Map

The App1 codebase can be used as a basis to build other application images that can be installed and run on the evaluation system.

The App1 codebase is based on the PK-S5D9 Blinky with ThreadX example project.

The Codebase can be found at: **C:\Renesas\Synergy\SecurityKernel\evaluation\src\embedded\programs\app1**

The Codebase of OemRx application can be found at:

C:\Renesas\Synergy\SecurityKernel\evaluation\src\embedded\programs\oemRx

To use either codebase as the basis for a new application:

1. Copy the tree from C:\Renesas\Synergy\SecurityKernel\evaluation\src to a new location
2. Open e^2 studio
3. Select File -> Import… -> General -> Existing projects into workspace -> Next
4. For Root directory choose the copied src directory and select App1

App1 will now appear in the workspace and can be edited/modified as required.

As well as the Blinky Thread, there is also a PMOD thread for handling the UART for updates and a button thread to check for button presses during the update.

The Vector table for the Security Kernel resides at address 0x0000 0000.

For user applications, the Vector table resides as the base address of the application flash region. This is set to 0x0001 8000 and a pair of modified linker scripts, S5D9.ld and memoryMap.ld, are used to link the application to the correct location.

- **SecurityKernel\evaluation\src\embedded\programs\app1\script\S5D9.ld**
- **SecurityKernel\evaluation\src\embedded\common\loaderScript\memoryMap.ld**

7.Extension to Production Devices

7.1 Update

The data interfaces supported by the final product will be unknown at the time that the Security Kernel is installed on the devices. Each product could contain different interfaces. They may be proprietary interfaces running proprietary transport protocols. Hence it is impossible to include transports over those (unknown) interfaces within the security kernel.

The unsecured Synergy chips support three mechanisms that the OEM could use to install the initial application image onto the chip. They are:

- JTAG
- Serial over SCI2
- USB over USB0

JTAG enables complete uncontrolled access to the processor enabling the user to view any memory and to modify most memory. JTAG would need to be disabled by the process of installing the security kernel.

To provide the nearest equivalent functionality to unsecured Synergy MCUs, a “Receiver 1st Time” downloader is installed on the secured Synergy MCUs as part of installing the security kernel. This application supports “Serial over SCI2” using the same protocol at the standard Synergy bootloader. Once the Application has been installed on the device, the OEM can choose to:

a) Keep the “Receiver 1st Time” downloader as the mechanism for installing updates (and recovering from update failures).

b) Add an “OEM Recover Downloader” to install updates (and perform recoveries) over the OEM defined interfaces, with the “Receiver 1st Time” downloader as backup if the “OEM Recovery Downloader” is corrupted.

c) Replace the “Receiver 1st Time” downloader with the “OEM Recovery Downloader”. This frees up the code spaced used by “Receiver 1st Time” downloader so that it be used by the application]

The OEM should implement their update transport mechanism as part of their application. Updates are delivered by the application and the kernel called to install. An OEM Recovery Downloader is recommended to recover the system (in the event of an update failure). The OEM Recovery Downloader can use sophisticated transport interfaces to deliver the recovery image. The OEM could choose not to produce a Recovery Downloader relying upon the 1st time downloader as the recovery mechanism. The OEM must be aware that in this configuration the recovery delivery is by serial connection only.

7.1.1 Update Strategies

The choice of update strategy affects the maximum application size that can be supported by the chip. The security kernel supports all the update strategies described below and OEM can choose the strategy to follow.

Dual-application. Up to 50% of the application flash is used for the application. Updates are performed by delivering the security-protected new version of the application into the other half of the application flash and then invoking the security kernel to install it. Maximum application size is about 1000 Kbytes on an S5D9 (This includes any code flash used for a flash file system, calibration tables, data, etc.).

Two-stage-update. In the first stage, the OEM Recovery Downloader installs an update that deletes all the application modules. The device then reboots with just the OEM Recovery Downloader running. The large update is then stored into flash with the modules stored in increasing memory order. At least 128 Kbytes must be left free at the bottom of the application flash. The Secure Installer can then apply the update from the bottom up overwriting the image in flash as it goes.
Updating the OEM Recovery Downloader. Possible approaches include:

a) If the “Receiver 1st Time” downloader is present then the OEM Recovery Downloader can be simply used to download and install an update to itself. If the update fails the “Receiver 1st Time” downloader has to be used to recover the OEM Recovery Downloader.

b) Otherwise the OEM Recovery Downloader can still try to install an update over itself. If the update fails the device is “ bricked” (permanently nonfunctional)!

c) Another option is to install a temporary OEM Recovery Downloader updater in place of the application. When this is running it can safely replace the OEM Recovery Downloader and then replace itself with the updated application.

d) Application contains a download mechanism that can be utilized to deliver an image for updating the OEM Recovery Downloader

Renesas Synergy Security Evaluation User’s Manual

Publication Date: Rev.1.00 April 27, 2017

Published by: Renesas Electronics Corporation

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.1

Renesas Synergy™
Security Evaluation

Renesas Electronics Corporation

Rxxxxxxxxxxxxxxxxx