RZ/T2, RZ/N2

How to use Mbed TLS

Introduction

This application note describes how to use Mbed TLS, an open-source SSL/TLS library, on RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H. Mbed TLS is a C library that implements cryptographic primitives, X.509 certificate manipulation and the SSL/TLS and DTLS protocols.

This package includes the full software Mbed TLS and Mbed TLS with accelerated cryptographic operations using RSIP, the Security IP.

The sample programs included in this package use FreeRTOS+TCP and lwIP Protocol Stack Sample Program Package as communication protocol libraries to be combined with Mbed TLS.

Target Device

RZ/T2M Group (Include security support product)

RZ/T2L Group (Include security support product)

RZ/T2H Group (Include security support product)

RZ/N2L Group (Include security support product)

RZ/N2H Group (Include security support product)

Contents

1. Overview

1.1 Introduction

The RZ/T2, RZ/N2 Mbed TLS sample program package shows how to use Mbed TLS, an open-source SSL/TLS library using the Renesas Starter Kit+ for the RZ/T2M, RZ/T2L and RZ/N2L or the RZ/T2H and RZ/N2H Evaluation Board.

This sample program package uses the Flexible Software Package, the Renesas Secure IP driver, and the RZ/T2, RZ/N2 Secure Boot sample program package. For more information on the Flexible Software Package, refer to *RZ/T2, RZ/N2 Getting Started with Flexible Software Package (R01AN6434EJ****)*, and for more information on the Renesas Secure IP driver included in the Flexible Software Package, refer to *RZ/T2, RZ/N2 Renesas Secure IP Driver (R01AN6547EJ****)* or *RZ/T2H and RZ/N2H Groups Renesas Secure IP Driver (R01AN7451EJ****)*. The RZ/T2, RZ/N2 Secure Boot sample program package must be used to run this sample program with Renesas Secure IP driver. For details, refer to *RZ/T2, RZ/N2 Device Setup Guide for Secure boot (R01AN6526EJ****)* or *RZ/T2H and RZ/N2 Groups Device Setup Guide for Secure boot (R01AN7452EJ****)* . The Renesas Secure IP Driver and Secure Boot sample program package are included in *RZ/T2M, T2L, N2L Group Security Solution Package (R01AN6956EJ****) or RZ/T2H and RZ/N2H Groups Security Solution Package (R01AN7594EJ****)*.

Also, the sample programs included in this package use FreeRTOS+TCP and lwIP Protocol Stack Sample Program Package as communication protocol libraries to be combined with Mbed TLS. For more information on the FreeRTOS+TCP, refer to *Readme_E.txt* included in the FreeRTOS+TCP package. For more information on the lwIP Protocol Stack Sample Program Package, refer to *RZ/T2M, RZ/N2L, RZ/T2L Group Quick Start Guide: lwIP Protocol Stack Sample Program (R01AN6560EJ****)* or *RZ/T2H, RZ/N2H Group Quick Start Guide: lwIP Protocol Stack Sample Program (R01AN7587EJ****)*.

This sample program package uses Mbed TLS 3.6.0 LTS. Mbed TLS files are provided under a dual [Apache-2.0](https://spdx.org/licenses/Apache-2.0.html) OR [GPL-2.0-or-later](https://spdx.org/licenses/GPL-2.0-or-later.html) license.

This sample program package uses Mbed TLS under Apache-2.0. For more information about the license information of Mbed TLS, please refer to the following:

https://github.com/Mbed-TLS/mbedtls/blob/v3.6.0/LICENSE

1.2 Features

1.2.1 Demonstration Features

TLS communication specifications for the demonstration software are as follows.

Table 1.1 shows the parameters to be agreed upon for TLS communication in the demonstration. Table 1.2 shows the Application Data sent and received between the server and client after the TLS connection is established in the demonstration: HTTP Request is sent by the client, and HTTP Response is sent by the server.

Table 1.1 Agreement parameters for TLS communication

Item	Agreement parameters
SSL/TLS Version	TLS1.2
Cipher Suites	TLS_ECDHE_ECDSA_WITH_AES128_GCM_SHA256
Key Exchange Scheme	ECDHE
Signature Algorithm	ECDSA
Cipher Algorithms	AES128 GCM
Hash Functions	SHA256
Peer Authentication	Client Authentication enabled
Supported TLS extensions	Extended Master Secret, Server Name Extension, ALPN Extension, Supported Groups Extension, Signature Algorithms Extension, Supported Point Formats extension, Max Fragment Length extension

Table 1.2 Demonstration request/response messages

Request/Response	Application Data
HTTP Request	GET / HTTP/1.0
HTTP Response	HTTP/1.0 200 OK Content-Type: text/html mbed TLS Test Server Successful connection using: TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256

1.2.2 Features of Mbed TLS for RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H

Mbed TLS for RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H includes an RSIP driver to accelerate cryptographic operations, which can be enabled or disabled. It supports the cipher suites listed in Table 1.3.

Table 1.3 List of Cipher Suites Supported by Mbed TLS for RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H

No	Cipher Suites	SSL/TLS Version
1	TLS_RSA_WITH_AES_128_CBC_SHA256	TLS1.2
2	TLS_RSA_WITH_AES_256_CBC_SHA256	TLS1.2
3	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLS1.2
4	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLS1.2
5	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLS1.2
6	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS1.2

1.3 Limitations

Mbed TLS for RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H has the following limitations:

- When hardware acceleration with RSIP is enabled, multiple TLS sessions cannot be used.

1.4 Package Contents

RZ/T2, RZ/N2 Mbed TLS sample program package contains several files with software and documents. The following table lists their contents.

Table 1.4 RZ/T2, RZ/N2 Mbed TLS Sample Program Package Contents

No.	File Name	Classification	Remarks
1	RZT2M_RSK_freertostcp_mbedtls_Rev220.zip	Software	FreeRTOS+TCP demo with Mbed TLS for RZ/T2M Note1
2	RZT2M_RSK_lwIP_mbedtls_Rev220.zip	Software	FreeRTOS+lwIP demo with Mbed TLS for RZ/T2M Note1
3	RZT2L_RSK_freertostcp_mbedtls_Rev220.zip	Software	FreeRTOS+TCP demo with Mbed TLS for RZ/T2L Note1
4	RZT2L_RSK_lwIP_mbedtls_Rev220.zip	Software	FreeRTOS+lwIP demo with Mbed TLS for RZ/T2L Note1
5	RZT2H_EVB_freertostcp_mbedtls_Rev230.zip	Software	FreeRTOS+TCP demo with Mbed TLS for RZ/T2H Note1
6	RZT2H_EVB_lwIP_mbedtls_Rev220.zip	Software	FreeRTOS+lwIP demo with Mbed TLS for RZ/T2H Note1
7	RZN2L_RSK_freertostcp_mbedtls_Rev230.zip	Software	FreeRTOS+TCP demo with Mbed TLS for RZ/N2L Note1
8	RZN2L_RSK_lwIP_mbedtls_Rev230.zip	Software	FreeRTOS+lwIP demo with Mbed TLS for RZ/N2L Note1
9	RZN2H_EVB_freertostcp_mbedtls_Rev230.zip	Software	FreeRTOS+TCP demo with Mbed TLS for RZ/N2H Note1
10	RZN2H_EVB_lwIP_mbedtls_Rev230.zip	Software	FreeRTOS+lwIP demo with Mbed TLS for RZ/N2H Note1
11	MbedTLS_ClientServer_demo.zip	Tool	Mbed TLS Client/Server tool source code for PC
12	ssl_client1.exe	Tool	Mbed TLS client tool for PC
13	ssl_server.exe	Tool	Mbed TLS server tool for PC
14	mbedtls-v3.6.0_with_rsip_rev210.zip	Software	Mbed TLS and RSIP driver source code included in each demo project
15	cert_info_rootca_ecc.bin	Certificate	Sample key data stored in the “Key Data” folder. These files are used in Section 2.1.1 of this document as a reference for building the environment.
16	cert_info_own_ecc.bin	Certificate	
17	key_info_own_ecc_priv.bin	Key	
18	r01an7372ej0230-rzt2-n2-security-tls.pdf	Document	This document
19	r01an7373ej0230-rzt2-n2-security-releasenote.pdf	Document	Release Note

Note1 Contains sample program projects for GCC and IAR compilers.

1.5 Related Documents

Table 1.5 lists documents related to this document.

Table 1.5 Related Documents

Title	Document Number
RZ/T2, RZ/N2 Getting Started with Flexible Software Package	R01AN6434EJ****
RZ/T2 Group Renesas Starter Kit+ for RZ/T2M CPU Board (Prototype) User's Manual	R20UT4936EG****
RZ/T2M Group Renesas Starter Kit+ for RZ/T2M User's Manual	R20UT4939EG****
RZ/T2M Group Renesas Starter Kit+ for RZ/T2M Quick Start Guide	R20UT4941EG****
RZ/T2M Group User's Manual: Hardware	R01UH0916EJ****
RZ/T2L Group Renesas Starter Kit+ for RZ/T2L User's Manual	R20UT5163EJ****
RZ/T2L Group User's Manual: Hardware	R01UH0985EJ****
RZ/N2L Group Renesas Starter Kit+ for RZ/N2L User's Manual	R20UT4984EG****
RZ/N2L Group User's Manual: Hardware	R01UH0955EJ****
RZ/T2H Group RZ/T2H Evaluation Board (Prototype) User's Manual	R20UT5316EJ****
RZ/T2H Group RZ/T2H Evaluation Board User's Manual	R20UT5405EJ****
RZ/N2H Group RZ/N2H Evaluation Board (Prototype) User's Manual	R20UT5356EJ****
RZ/N2H Group RZ/N2H Evaluation Board User's Manual	R20UT5522EJ****
RZ/T2H and RZ/N2H Groups User's Manual: Hardware	R01UH1039EJ****
RZ/T2M, RZ/N2L and RZ/T2L Group Security Features User's Manual	R01UH1017EJ****
RZ/T2H and RZ/N2H Groups Security Features User's Manual	R01UH1144EJ****
RZ/T2M, T2L, N2L Group Security Solution Package	R01AN6956EJ****
RZ/T2H and RZ/N2H Group Security Solution Package	R01AN7594EJ*****
RZ/T2, RZ/N2 Renesas Secure IP Driver	R01AN6547EJ****
RZ/T2H and RZ/N2H Groups Renesas Secure IP Driver	R01AN7451EJ****
RZ/T2, RZ/N2 Device Setup Guide for Secure boot	R01AN6526EJ****
RZ/T2H and RZ/N2H Groups Device Setup Guide for Secure boot	R01AN7452EJ****
RZ/T2M, RZ/N2L, RZ/T2L Group Quick Start Guide: lwIP Protocol Stack Sample Program	R01AN6560EJ****
RZ/T2H, RZ/N2H Group Quick Start Guide: lwIP Protocol Stack Sample Program	R01AN7587EJ****
Readme_E.txt	(Included in the FreeRTOS+TCP package)

1.6 Explanation of Terms

The meanings of terms used in this document are indicated below.

Term Used in This Document	Meaning of Term
Mbed TLS	An open-source SSL/TLS library. Mbed TLS is a C library that implements cryptographic primitives, X.509 certificate manipulation and the SSL/TLS and DTLS protocols.
lwIP	lightweight IP An open-source embedded TCP/IP protocol stack.
TCP	Transmission Control Protocol
RSIP	Renesas Secure IP
User Key	Encryption key used in the user application AES key, RSA public key, RSA private key, etc
User Factory Programming Key (UFPK)	256-bit symmetric key data used to wrap User Keys
W-UFPK	UFPK wrapped by the Renesas Key Wrapping service
Renesas Key Wrap Service	A service that wraps a UFPK with a device-specific key to generate a W-UFPK https://dlm.renesas.com/keywrap/

2. Quick Start

The procedure for configuring and run secure communication demonstration system with Mbed TLS for each device using the demonstration program included in this package is described below.

The demonstration program runs on the Renesas Starter Kit+ for RZ/T2M, RZ/T2L and RZ/N2L (hereinafter referred to as RSK+) or RZ/T2H and RZ/N2H Evaluation Board (hereinafter referred to as EVB). If you want to enable cryptographic acceleration with the RSIP driver in the Mbed TLS for each device, please use the socket version of RSK+ or EVB and the security support products.

For details on how to install and use tools, refer to the document "Getting Started with Flexible Software Package".

2.1 Board Settings for RZ/T2M

Table 2.1 shows the environment required for configuring the RSK+.

Table 2.1 Setup Environment for RSK+ RZ/T2M

Name	Remarks
Evaluation board	RSK+ RZ/T2M (or its socket version)
USB cables	1, For USB to Serial
	1, For power supply
Ether cable	Directly connect the board to the PC
Windows host PC	Windows 10 IAR Embedded Workbench or e^2 studio, Tera Term, Mbed TLS client tool (ssl_client1.exe) and Mbed TLS server tool (ssl_server.exe) The environment of this sample program was expected to be as below: - IAR Embedded Workbench for ARM:9.60.2, Smart Configurator + RZT_FSP_Packs_v2.2.0 - e^2 studio 2024-10 (GCC Compiler: 12.2 rel1), RZT_FSP_Packs_v2.2.0
Debugger	I-jet or J-LINK or USB cable USB cable is required when using J-Link™ OB on RSK+ as debugger
secure_device_setup.py Note1	Command sending tool for secure device setup
RZT2M_RSK_SecureDeviceSetup_*.out.srec Note1	Pre-built secure device setup program for RZ/T2M

Note1 Included in RZ/T2, RZ/N2 Secure Boot sample program package (R01AN6526EJ****).

2.1.1 Inject TLS Certificate and Key

If you want to enable cryptographic acceleration using the RSIP driver in Mbed TLS for RZ/T2M, the certificates for TLS and its private key handled by the RSIP driver must be injected into flash beforehand.

The addresses of the flashes to inject the certificate and key are listed in Table 2.2.

Table 2.2 Placement of certificates and keys on flash memory for RZ/T2M

Certificates and Keys	Address	File Name
RootCA Certificate	0x600001000	cert_info_rootca_ecc.bin
Own Certificate	0x600002000	cert_info_server_ecc.bin
Own Private Key	0x600004000	key_info_server_ecc_priv.bin

Inject the certificates for TLS and its private key into the RSK+ flash according to Chapter 2 of the *RZ/T2, RZ/N2 Device Setup Guide for Secure boot (R01AN6526EJ****)*. *RZ/T2, RZ/N2 Secure Boot sample program package (R01AN6526EJ****)* is included in *RZ/T2M, T2L, N2L Group Security Solution Package (R01AN6956EJ****)*.

After setting up the Host PC and board according to section 2.1 of the Device Setup Guide for Secure boot Device Setup Guide for Secure boot, follow the steps below to inject the certificates for TLS and its private key into the RSK+ flash.

In this section, you will use the pre-wrapped certificate and key files for injection into the flash. Refer to section 2.4 of the Device Setup Guide for Secure boot for details on certificate and key injection, including how to wrap the certificate and key.

1. Boot the device in SCI boot mode or USB boot mode and load the secure device setup program (RZT2M_RSK_SecureDeviceSetup_*.out.srec) into the RAM of the device.
   If the program is successfully loaded, the secure device setup program will start.

   In SCI boot mode:

   a-1) Set SW4 on the RSK+ board to the following and press S3 RESET.
   RZ/T2M:

SW	Setting	Description
SW4.1	OFF	SCI (UART) boot mode.
SW4.2	ON	
SW4.3	OFF	
SW4.4	ON	JTAG Authentication by Hash is disabled.
SW4.5	OFF	ATCM wait cycle = 1 wait.

   a-2) Load the secure device setup program (RZT2M_RSK_SecureDeviceSetup_*.out.srec) into the device using the secure device setup tool (secure_device_setup.py).
   The following command loads RZT2M_RSK_SecureDeviceSetup_SCI.out.srec:

```
> python secure_device_setup.py start --port COM3 --boot_mode sci -i
RZT2M_RSK_SecureDeviceSetup_SCI.out.srec

SCI Download mode.
Send program data. (S0)
-- Load Program to BTCM  ---------------
Send program data. (S3)
-- Start Boot Program on BTCM ---------
```

   In USB boot mode:

   b-1) Set SW4 on the RSK+ board to the following and press S3 RESET.
   RZ/T2M:

SW	Setting	Description
SW4.1	ON	USB boot mode.
SW4.2	OFF	
SW4.3	OFF	
SW4.4	ON	JTAG Authentication by Hash is disabled.
SW4.5	OFF	ATCM 1 wait

b-2) Load the secure device setup program (RZT2M_RSK_SecureDeviceSetup_*.out.srec) into the device using the secure device setup tool (secure_device_setup.py).
The following command loads RZT2M_RSK_SecureDeviceSetup_USB.out.srec:

```
> python secure_device_setup.py start --port COM3 --boot_mode usb -i
RZT2M_RSK_SecureDeviceSetup_USB.out.srec

USB Open.
USB Download mode (Normal USB boot)
Send program data. (S0)
-- Load Program to BTCM  ---------------
Send program data. (S3)
-- Start Boot Program on BTCM ---------
```

2. Use secure_device_setup.py to inject the certificates and keys.
Program the certificates and keys into the external flash on the RSK+ using the secure device setup tool (secure_device_setup.py).
If you have started the secure device setup program in USB boot mode, the COM port of the device may change from the boot time; to check the COM port again.

```
> python secure_device_setup.py instcert --port COM3 -
i ./rzt2m/cert_info_rootca_ecc.bin
instcert : Setup success.

> python secure_device_setup.py instcert --port COM3 -
i ./rzt2m/cert_info_server_ecc.bin
instcert : Setup success.

> python secure_device_setup.py instkey  --port COM3 -
i ./rzt2m/key_info_server_ecc_priv.bin
instkey : Setup success.
```

2.1.2 Board Settings for Demo

1. Set the switch as follows:

Table 2.3 Operation Mode Switch Settings for RSK+ RZT2M

SW	Setting	Description
SW4.1	ON	16-bit bus boot mode (NOR Flash) External flash memory must be blank to start this sample program (RAM execution). NOR flash memory on this board is blank, so set the operating mode switch for 16-bit bus boot mode (NOR flash).
SW4.2	OFF	
SW4.3	ON	
SW4.4	ON	JTAG Authentication by Hash is disabled.
SW4.5	OFF	ATCM wait cycle = 1 wait. This sample program runs on TCM and CPU frequency is set to 800MHz in the startup process. Therefore, the following settings (MDW = 1) are required.

2. Connect the PC to the RSK+ with an Ethernet cable. Ethernet uses ETH0 for RZ/T2M.
3. Connect the PC to the RSK+ USB-Serial connector (CN16) with USB cable (MiniB, type-A).
4. Connect the debugger to the PC and the RSK+. For details on connecting the debugger, refer to the document “Getting Started with Flexible Software Package”.

2.2 Board Settings for RZ/T2L

Table 2.4 shows the environment required for configuring the RSK+.

Table 2.4 Setup Environment for RSK+ RZ/T2L

Name	Remarks
Evaluation board	RSK+ RZ/T2L (or its socket version)
USB cables	1, For USB to Serial
	1, For power supply
Ether cable	Directly connect the board to the PC
Windows host PC	Windows 10 IAR Embedded Workbench or e^2 studio, Tera Term, Mbed TLS client tool (ssl_client1.exe) and Mbed TLS server tool (ssl_server.exe) The environment of this sample program was expected to be as below: - IAR Embedded Workbench for ARM:9.60.2, Smart Configurator + RZT_FSP_Packs_v2.2.0 - e^2 studio 2024-10 (GCC Compiler: 12.2 rel1), RZT_FSP_Packs_v2.2.0
Debugger	I-jet or J-LINK or USB cable USB cable is required when using J-Link™ OB on RSK+ as debugger
secure_device_setup.py Note1	Command sending tool for secure device setup
RZT2L_RSK_SecureDeviceSetup_*.out.srec Note1	Pre-built secure device setup program for RZ/T2L

Note1 Included in RZ/T2, RZ/N2 Secure Boot sample program package (R01AN6526EJ****).

2.2.1 Inject TLS Certificate and Key

If you want to enable cryptographic acceleration using the RSIP driver in Mbed TLS for RZ/T2L, the certificates for TLS and its private key handled by the RSIP driver must be injected into flash beforehand.

The addresses of the flashes to inject the certificate and key are listed inTable 2.5.

Table 2.5 Placement of certificates and keys on flash memory for RZ/T2L

Certificates and Keys	Address	File Name
RootCA Certificate	0x680001000	cert_info_rootca_ecc.bin
Own Certificate	0x680002000	cert_info_server_ecc.bin
Own Private Key	0x680004000	key_info_server_ecc_priv.bin

Inject the certificates for TLS and its private key into the RSK+ flash according to Chapter 2 of the *RZ/T2, RZ/N2 Device Setup Guide for Secure boot (R01AN6526EJ****)*. *RZ/T2, RZ/N2 Secure Boot sample program package (R01AN6526EJ****)* is included in *RZ/T2M, T2L, N2L Group Security Solution Package (R01AN6956EJ****)*.

After setting up the Host PC and board according to section 2.1 of the Device Setup Guide for Secure boot Device Setup Guide for Secure boot, follow the steps below to inject the certificates for TLS and its private key into the RSK+ flash.

In this section, you will use the pre-wrapped certificate and key files for injection into the flash. Refer to section 2.4 of the Device Setup Guide for Secure boot for details on certificate and key injection, including how to wrap the certificate and key.

1. Boot the device in SCI boot mode or USB boot mode and load the secure device setup program (RZT2L_RSK_SecureDeviceSetup_*.out.srec) into the RAM of the device.
   If the program is successfully loaded, the secure device setup program will start.

   In SCI boot mode:

   a-1) Set SW4 on the RSK+ board to the following and press S3 RESET.
   RZ/T2L:

SW	Setting	Description
SW4.1	OFF	SCI (UART) boot mode.
SW4.2	ON	
SW4.3	OFF	
SW4.4	OFF	ATCM wait cycle = 1 wait.
SW4.5	ON	JTAG Authentication by Hash is disabled.

   a-2) Load the secure device setup program (RZT2L_RSK_SecureDeviceSetup_*.out.srec) into the device using the secure device setup tool (secure_device_setup.py).
   The following command loads RZT2L_RSK_SecureDeviceSetup_SCI.out.srec:

```
> python secure_device_setup.py start --port COM3 --boot_mode sci -i
RZT2L_RSK_SecureDeviceSetup_SCI.out.srec

SCI Download mode.
Send program data. (S0)
-- Load Program to BTCM  ---------------
Send program data. (S3)
-- Start Boot Program on BTCM ---------
```

   In USB boot mode:

   b-1) Set SW4 on the RSK+ board to the following and press S3 RESET.
   RZ/T2L:

SW	Setting	Description
SW4.1	ON	USB boot mode.
SW4.2	OFF	
SW4.3	OFF	
SW4.4	OFF	ATCM wait cycle = 1 wait.
SW4.5	ON	JTAG Authentication by Hash is disabled.

   b-2) Load the secure device setup program (RZT2L_RSK_SecureDeviceSetup_*.out.srec) into the device using the secure device setup tool (secure_device_setup.py).
   The following command loads RZT2L_RSK_SecureDeviceSetup_USB.out.srec:

```
> python secure_device_setup.py start --port COM3 --boot_mode usb -i
RZT2L_RSK_SecureDeviceSetup_USB.out.srec

USB Open.
USB Download mode (Normal USB boot)
Send program data. (S0)
-- Load Program to BTCM  ---------------
Send program data. (S3)
-- Start Boot Program on BTCM ---------
```

2. Use secure_device_setup.py to inject the certificates and keys.
   Program the certificates and keys into the external flash on the RSK+ using the secure device setup tool (secure_device_setup.py).
   If you have started the secure device setup program in USB boot mode, the COM port of the device may change from the boot time; to check the COM port again.

```
> python secure_device_setup.py instcert --port COM3 -
i ./rzt2l/cert_info_rootca_ecc.bin
instcert : Setup success.

> python secure_device_setup.py instcert --port COM3 -
i ./rzt2l/cert_info_server_ecc.bin
instcert : Setup success.

> python secure_device_setup.py instkey  --port COM3 -
i ./rzt2l/key_info_server_ecc_priv.bin
instkey : Setup success.
```

2.2.2 Board Settings for Demo

1. Set the switch as follows:

Table 2.6 Operation Mode Switch Settings for RSK+ RZT2L

SW	Setting	Description
SW4.1	ON	16-bit bus boot mode (NOR Flash) External flash memory must be blank to start this sample program (RAM execution). NOR flash memory on this board is blank, so set the operating mode switch for 16-bit bus boot mode (NOR flash).
SW4.2	OFF	
SW4.3	ON	
SW4.4	OFF	ATCM wait cycle = 1 wait. This sample program runs on TCM and CPU frequency is set to 800MHz in the startup process. Therefore, the following settings (MDW = 1) are required.
SW4.5	ON	JTAG Authentication by Hash is disabled.

2. Connect the PC to the RSK+ with an Ethernet cable. Ethernet uses ETH2 for RZ/T2L.
3. Connect the PC to the RSK+ USB-Serial connector (CN16) with USB cable (MiniB, type-A).
4. Connect the debugger to the PC and the RSK+. For details on connecting the debugger, refer to the document “Getting Started with Flexible Software Package”.

2.3 Board Settings for RZ/N2L

Table 2.7 shows the environment required for configuring the RSK+.

Table 2.7 Setup Environment for RSK+ RZ/N2L

Name	Remarks
Evaluation board	RSK+ RZ/N2L (or its socket version)
USB cables	1, For USB to Serial
	1, For power supply
Ether cable	Directly connect the board to the PC
Windows host PC	Windows 10 IAR Embedded Workbench or e^2 studio, Tera Term, Mbed TLS client tool (ssl_client1.exe) and Mbed TLS server tool (ssl_server.exe) The environment of this sample program was expected to be as below: - IAR Embedded Workbench for ARM:9.60.2, Smart Configurator + RZN_FSP_Packs_v2.1.0 - e^2 studio 2024-01 (GCC Compiler: 12.2 rel1), RZN_FSP_Packs_v2.1.0
Debugger	I-jet or J-LINK or USB cable USB cable is required when using J-Link™ OB on RSK+ as debugger
secure_device_setup.py Note1	Command sending tool for secure device setup
RZN2L_RSK_SecureDeviceSetup_*_qspi.out.srec Note1	Pre-built secure device setup program for RZ/N2L

Note1 Included in RZ/T2, RZ/N2 Secure Boot sample program package (R01AN6526EJ****).

2.3.1 Inject TLS Certificate and Key

If you want to enable cryptographic acceleration using the RSIP driver in Mbed TLS for RZ/N2L, the certificates for TLS and its private key handled by the RSIP driver must be injected into flash beforehand.

The addresses of the flashes to inject the certificate and key are listed in Table 2.8.

Table 2.8 Placement of certificates and keys on flash memory for RZ/N2L

Certificates and Keys	Address	File Name
RootCA Certificate	0x600001000	cert_info_rootca_ecc.bin
Own Certificate	0x600002000	cert_info_server_ecc.bin
Own Private Key	0x600004000	key_info_server_ecc_priv.bin

Inject the certificates for TLS and its private key into the RSK+ flash according to Chapter 2 of the *RZ/T2, RZ/N2 Device Setup Guide for Secure boot (R01AN6526EJ****)*. *RZ/T2, RZ/N2 Secure Boot sample program package (R01AN6526EJ****)* is included in *RZ/T2M, T2L, N2L Group Security Solution Package (R01AN6956EJ****)*.

After setting up the Host PC and board according to section 2.1 of the Device Setup Guide for Secure boot Device Setup Guide for Secure boot, follow the steps below to inject the certificates for TLS and its private key into the RSK+ flash.

In this section, you will use the pre-wrapped certificate and key files for injection into the flash. Refer to section 2.4 of the Device Setup Guide for Secure boot for details on certificate and key injection, including how to wrap the certificate and key.

1. Boot the device in SCI boot mode or USB boot mode and load the secure device setup program (RZN2L_RSK_SecureDeviceSetup_*.out.srec) into the RAM of the device.
   If the program is successfully loaded, the secure device setup program will start.

   In SCI boot mode:

   a-1) Set SW4 on the RSK+ board to the following and press S3 RESET.
   RZ/N2L:

SW	Setting	Description
SW4.1	OFF	SCI (UART) boot mode.
SW4.2	ON	
SW4.3	OFF	
SW4.4	ON	JTAG Authentication by Hash is disabled.

   a-2) Load the secure device setup program (RZN2L_RSK_SecureDeviceSetup_*.out.srec) into the device using the secure device setup tool (secure_device_setup.py).
   The following command loads RZN2L_RSK_SecureDeviceSetup_SCI.out.srec:

```
> python secure_device_setup.py start --port COM3 --boot_mode sci -i
RZN2L_RSK_SecureDeviceSetup_SCI.out.srec

SCI Download mode.
Send program data. (S0)
-- Load Program to BTCM  ---------------
Send program data. (S3)
-- Start Boot Program on BTCM ---------
```

   In USB boot mode:

   b-1) Set SW4 on the RSK+ board to the following and press S3 RESET.
   RZ/N2L:

SW	Setting	Description
SW4.1	ON	USB boot mode.
SW4.2	OFF	
SW4.3	OFF	
SW4.4	ON	JTAG Authentication by Hash is disabled.

   b-2) Load the secure device setup program (RZN2L_RSK_SecureDeviceSetup_*.out.srec) into the device using the secure device setup tool (secure_device_setup.py).
   The following command loads RZN2L_RSK_SecureDeviceSetup_USB.out.srec:

```
> python secure_device_setup.py start --port COM3 --boot_mode usb -i
RZN2L_RSK_SecureDeviceSetup_USB.out.srec

USB Open.
USB Download mode (Normal USB boot)
Send program data. (S0)
-- Load Program to BTCM  ---------------
Send program data. (S3)
-- Start Boot Program on BTCM ---------
```

2. Use secure_device_setup.py to inject the certificates and keys.
   Program the certificates and keys into the external flash on the RSK+ using the secure device setup tool (secure_device_setup.py).
   If you have started the secure device setup program in USB boot mode, the COM port of the device may change from the boot time; to check the COM port again.

```
> python secure_device_setup.py instcert --port COM3 -
i ./rzn2l/cert_info_rootca_ecc.bin
instcert : Setup success.

> python secure_device_setup.py instcert --port COM3 -
i ./rzn2l/cert_info_server_ecc.bin
instcert : Setup success.

> python secure_device_setup.py instkey  --port COM3 -
i ./rzn2l/key_info_server_ecc_priv.bin
instkey : Setup success.
```

2.3.2 Board Settings for Demo

1. Set the switch as follows:

Table 2.9 Operation Mode Switch Settings for RSK+ RZN2L

SW	Setting	Description
SW4.1	ON	16-bit bus boot mode (NOR Flash) External flash memory must be blank to start this sample program (RAM execution). NOR flash memory on this board is blank, so set the operating mode switch for 16-bit bus boot mode (NOR flash).
SW4.2	OFF	
SW4.3	ON	
SW4.4	ON	JTAG Authentication by Hash is disabled.

2. Connect the PC to the RSK+ with an Ethernet cable. Ethernet uses ETH0 for RZ/N2L.
3. Connect the PC to the RSK+ USB-Serial connector (CN16) with USB cable (MiniB, type-A).
4. Connect the debugger to the PC and the RSK+. For details on connecting the debugger, refer to the document “Getting Started with Flexible Software Package”.

2.4 Board Settings for RZ/T2H Cortex®-R52

Table 2.10 shows the environment required for configuring the EVB.

Table 2.10 Setup Environment for Cortex®-R52 on RZ/T2H EVB

Name	Remarks
Evaluation board	RZ/T2H EVB (or its socket version)
USB cable	For USB to Serial
Power supply cable	2.0-mm center-positive power jack, 15-V/3-A DC
Ether cable	Directly connect the board to the PC
Windows host PC	Windows 10 IAR Embedded Workbench or e^2 studio, Tera Term, Mbed TLS client tool (ssl_client1.exe) and Mbed TLS server tool (ssl_server.exe) The environment of this sample program was expected to be as below: - IAR Embedded Workbench for ARM:9.60.2, Smart Configurator + RZT_FSP_Packs_v2.2.0 - e^2 studio 2024-10 (GCC Compiler: 12.2 rel1), RZT_FSP_Packs_v2.2.0
Debugger	I-jet or J-LINK or USB cable USB cable is required when using J-Link™ OB on EVB as debugger
secure_device_setup.py Note1	Command sending tool for secure device setup
RZT2H_EVB_SecureDeviceSetup_*.out.srec Note1	Pre-built secure device setup program for RZ/T2H

Note1 Included in RZ/T2H and RZ/N2H Secure Boot sample program package (R01AN7452EJ****).

2.4.1 Inject TLS Certificate and Key

If you want to enable cryptographic acceleration using the RSIP driver in Mbed TLS for RZ/T2H, the certificates for TLS and its private key handled by the RSIP driver must be injected into flash beforehand.

The addresses of the flashes to inject the certificate and key are listed in Table 2.11.

Table 2.11 Placement of certificates and keys on flash memory for RZ/T2H

Certificates and Keys	Address	File Name
RootCA Certificate	0x400001000	cert_info_rootca_ecc.bin
Own Certificate	0x400002000	cert_info_server_ecc.bin
Own Private Key	0x400004000	key_info_server_ecc_priv.bin

Inject the certificates for TLS and its private key into the EVB flash according to Chapter 2 of the *RZ/T2H and RZ/N2H Groups Device Setup Guide for Secure boot (R01AN7452EJ****)*. *RZ/T2H and RZ/N2H Groups Device Setup Guide for Secure boot (R01AN7452EJ****)* is included in *RZ/T2H and RZ/N2H Group Security Solution Package (R01AN7594EJ****)*.

After setting up the Host PC and board according to section 2.1 of the Device Setup Guide for Secure boot Device Setup Guide for Secure boot, follow the steps below to inject the certificates for TLS and its private key into the EVB flash.

In this section, you will use the pre-wrapped certificate and key files for injection into the flash. Refer to section 2.4 of the Device Setup Guide for Secure boot for details on certificate and key injection, including how to wrap the certificate and key.

1. Boot the device in SCI boot mode or USB boot mode and load the secure device setup program (RZT2H_EVB_SecureDeviceSetup_*.out.srec) into the RAM of the device.
   If the program is successfully loaded, the secure device setup program will start.

   In SCI boot mode:

   a-1) Set SW14 on the EVB board to the following and press SW13 RESET.
   Cortex®-R52 on RZ/T2H

SW	Setting	Description
SW14.1	OFF	SCI (UART) boot mode.
SW14.2	ON	
SW14.3	OFF	
SW14.4	OFF	Cortex®-R52 CPU0 ATCM wait cycle = 1 wait.
SW14.5	OFF	Cortex®-R52 CPU1 ATCM wait cycle = 1 wait.
SW14.6	OFF	Supply voltage of boot peripheral is 3.3 V
SW14.7	ON	JTAG Authentication by Hash is disabled.

   a-2) Load the secure device setup program (RZT2H_EVB_SecureDeviceSetup_*.out.srec) into the device using the secure device setup tool (secure_device_setup.py).
   The following command loads RZT2H_EVB_SecureDeviceSetup_SCI.out.srec:

```
> python secure_device_setup.py start --port COM3 --boot_mode sci -i
RZT2H_EVB_SecureDeviceSetup_SCI.out.srec

SCI Download mode.
Send program data. (S0)
-- Load Program to RAM  ---------------
Send program data. (S3)
-- Start Boot Program on RAM ---------
```

   In USB boot mode:

   b-1) Set SW14 on the EVB board to the following and press SW13 RESET.
   Cortex®-R52 on RZ/T2H:

SW	Setting	Description
SW14.1	ON	USB boot mode.
SW14.2	OFF	
SW14.3	OFF	
SW14.4	OFF	Cortex®-R52 CPU0 ATCM wait cycle = 1 wait.
SW14.5	OFF	Cortex®-R52 CPU1 ATCM wait cycle = 1 wait.
SW14.6	OFF	Supply voltage of boot peripheral is 3.3 V
SW14.7	ON	JTAG Authentication by Hash is disabled.

   b-2) Load the secure device setup program (RZT2H_EVB_SecureDeviceSetup_*.out.srec) into the device using the secure device setup tool (secure_device_setup.py).
   The following command loads RZT2H_EVB_SecureDeviceSetup_USB.out.srec:

```
> python secure_device_setup.py start --port COM3 --boot_mode usb -i
RZT2H_EVB_SecureDeviceSetup_USB.out.srec

USB Open.
USB Download mode (Normal USB boot)
Send program data. (S0)
-- Load Program to RAM  ---------------
Send program data. (S3)
-- Start Boot Program on RAM ---------
```

2. Use secure_device_setup.py to inject the certificates and keys.
   Program the certificates and keys into the external flash on the EVB using the secure device setup tool (secure_device_setup.py).
   If you have started the secure device setup program in USB boot mode, the COM port of the device may change from the boot time; to check the COM port again.

```
> python secure_device_setup.py instcert --port COM3 -
i ./rzt2h/cert_info_rootca_ecc.bin
instcert : Setup success.

> python secure_device_setup.py instcert --port COM3 -
i ./rzt2h/cert_info_server_ecc.bin
instcert : Setup success.

> python secure_device_setup.py instkey  --port COM3 -
i ./rzt2h/key_info_server_ecc_priv.bin
instkey : Setup success.
```

2.4.2 Board Settings for Demo

1. Set the switch as follows:

Table 2.12 Operation Mode Switch Settings for Cortex®-R52 on RZT2H EVB

SW	Setting	Description
SW14.1	ON	xSPI1 boot mode (QSPI Flash) External flash memory must be blank to start this sample program (RAM execution). QSPI flash memory on this board is blank, so set the operating mode switch for xSPI1 boot mode (QSPI Flash)
SW14.2	OFF	
SW14.3	ON	
SW14.4	OFF	Cortex®-R52 CPU0 ATCM wait cycle = 1 wait.
SW14.5	OFF	Cortex®-R52 CPU1 ATCM wait cycle = 1 wait.
SW14.6	OFF	Supply voltage of boot peripheral is 3.3 V
SW14.7	ON	JTAG Authentication by Hash is disabled.

2. Connect the PC to the EVB with an Ethernet cable. Ethernet uses ETH0 for RZ/T2H.

3. Connect the PC to the USB-Serial connector (CN34) with USB cable (MiniB, type-A).

4. Connect the debugger to the PC and the EVB. For details on connecting the debugger, refer to the document “Getting Started with Flexible Software Package”.

2.5 Board Settings for RZ/N2H Cortex®-R52

Table 2.13 shows the environment required for configuring the EVB.

Table 2.13 Setup Environment for Cortex®-R52 on RZ/N2H EVB

Name	Remarks
Evaluation board	RZ/N2H EVB (or its socket version)
USB cable	For USB to Serial
Power supply cable	2.0-mm center-positive power jack, 15-V/3-A DC
Ether cable	Directly connect the board to the PC
Windows host PC	Windows 10 IAR Embedded Workbench or e^2 studio, Tera Term, Mbed TLS client tool (ssl_client1.exe) and Mbed TLS server tool (ssl_server.exe) The environment of this sample program was expected to be as below: - IAR Embedded Workbench for ARM:9.60.2, Smart Configurator + RZN_FSP_Packs_v2.1.0 - e^2 studio 2024-10 (GCC Compiler: 12.2 rel1), RZN_FSP_Packs_v2.1.0
Debugger	I-jet or J-LINK or USB cable USB cable is required when using J-Link™ OB on EVB as debugger
secure_device_setup.py Note1	Command sending tool for secure device setup
RZN2H_EVB_SecureDeviceSetup_*.out.srec Note1	Pre-built secure device setup program for RZ/N2H

Note1 Included in RZ/T2H and RZ/N2H Secure Boot sample program package (R01AN7452EJ****).

2.5.1 Inject TLS Certificate and Key

If you want to enable cryptographic acceleration using the RSIP driver in Mbed TLS for RZ/N2H, the certificates for TLS and its private key handled by the RSIP driver must be injected into flash beforehand.

The addresses of the flashes to inject the certificate and key are listed in Table 2.14.

Table 2.14 Placement of certificates and keys on flash memory for RZ/N2H

Certificates and Keys	Address	File Name
RootCA Certificate	0x400001000	cert_info_rootca_ecc.bin
Own Certificate	0x400002000	cert_info_server_ecc.bin
Own Private Key	0x400004000	key_info_server_ecc_priv.bin

Inject the certificates for TLS and its private key into the EVB flash according to Chapter 2 of the *RZ/T2H and RZ/N2H Groups Device Setup Guide for Secure boot (R01AN7452EJ****)*. *RZ/T2H and RZ/N2H Groups Device Setup Guide for Secure boot (R01AN7452EJ****)* is included in *RZ/T2H and RZ/N2H Group Security Solution Package (R01AN7594EJ****)*.

After setting up the Host PC and board according to section 2.1 of the Device Setup Guide for Secure boot Device Setup Guide for Secure boot, follow the steps below to inject the certificates for TLS and its private key into the EVB flash.

In this section, you will use the pre-wrapped certificate and key files for injection into the flash. Refer to section 2.4 of the Device Setup Guide for Secure boot for details on certificate and key injection, including how to wrap the certificate and key.

1. Boot the device in SCI boot mode or USB boot mode and load the secure device setup program (RZN2H_EVB_SecureDeviceSetup_*.out.srec) into the RAM of the device.
   If the program is successfully loaded, the secure device setup program will start.

   In SCI boot mode:

   a-1) Set DSW3 on the EVB board to the following and press SW5 RESET.
   Cortex®-R52 on RZ/N2H の h:

SW	Setting	Description
DSW3.1	OFF	SCI (UART) boot mode.
DSW3.2	ON	
DSW3.3	OFF	
DSW3.4	OFF	Cortex®-R52 CPU0 ATCM wait cycle = 1 wait.
DSW3.5	OFF	Cortex®-R52 CPU1 ATCM wait cycle = 1 wait.
DSW3.6	OFF	Supply voltage of boot peripheral is 3.3 V
DSW3.7	ON	JTAG Authentication by Hash is disabled.

   a-2) Load the secure device setup program (RZN2H_EVB_SecureDeviceSetup_*.out.srec) into the device using the secure device setup tool (secure_device_setup.py).
   The following command loads RZN2H_EVB_SecureDeviceSetup_SCI.out.srec:

```
> python secure_device_setup.py start --port COM3 --boot_mode sci -i
RZN2H_EVB_SecureDeviceSetup_SCI.out.srec

SCI Download mode.
Send program data. (S0)
-- Load Program to RAM  ---------------
Send program data. (S3)
-- Start Boot Program on RAM ---------
```

   In USB boot mode:

   b-1) Set DSW3 on the EVB board to the following and press SW5 RESET.
   Cortex®-R52 on RZ/N2H:

SW	Setting	Description
DSW3.1	ON	USB boot mode.
DSW3.2	OFF	
DSW3.3	OFF	
DSW3.4	OFF	Cortex®-R52 CPU0 ATCM wait cycle = 1 wait.
DSW3.5	OFF	Cortex®-R52 CPU1 ATCM wait cycle = 1 wait.
DSW3.6	OFF	Supply voltage of boot peripheral is 3.3 V
DSW3.7	ON	JTAG Authentication by Hash is disabled.

   b-2) Load the secure device setup program (RZN2H_EVB_SecureDeviceSetup_*.out.srec) into the device using the secure device setup tool (secure_device_setup.py).
   The following command loads RZN2H_EVB_SecureDeviceSetup_USB.out.srec:

```
> python secure_device_setup.py start --port COM3 --boot_mode usb -i
RZN2H_EVB_SecureDeviceSetup_USB.out.srec

USB Open.
USB Download mode (Normal USB boot)
Send program data. (S0)
-- Load Program to RAM  ---------------
Send program data. (S3)
-- Start Boot Program on RAM ---------
```

2. Use secure_device_setup.py to inject the certificates and keys.
   Program the certificates and keys into the external flash on the EVB using the secure device setup tool (secure_device_setup.py).
   If you have started the secure device setup program in USB boot mode, the COM port of the device may change from the boot time; to check the COM port again.

```
> python secure_device_setup.py instcert --port COM3 -
i ./rzn2h/cert_info_rootca_ecc.bin
instcert : Setup success.

> python secure_device_setup.py instcert --port COM3 -
i ./rzn2h/cert_info_server_ecc.bin
instcert : Setup success.

> python secure_device_setup.py instkey  --port COM3 -
i ./rzn2h/key_info_server_ecc_priv.bin
instkey : Setup success.
```

2.5.2 Board Settings for Demo

1. Set the switch as follows:

Table 2.15 Operation Mode Switch Settings for Cortex®-R52 on RZ/N2H EVB

SW	Setting	Description
DSW3.1	ON	xSPI1 boot mode (QSPI Flash) External flash memory must be blank to start this sample program (RAM execution). QSPI flash memory on this board is blank, so set the operating mode switch for xSPI1 boot mode (QSPI Flash)
DSW3.2	OFF	
DSW3.3	ON	
DSW3.4	OFF	Cortex®-R52 CPU0 ATCM wait cycle = 1 wait.
DSW3.5	OFF	Cortex®-R52 CPU1 ATCM wait cycle = 1 wait.
DSW3.6	OFF	Supply voltage of boot peripheral is 3.3 V
DSW3.7	ON	JTAG Authentication by Hash is disabled.

2. Connect the PC to the EVB with an Ethernet cable. Ethernet uses ETH0 for RZ/N2H.

3. Connect the PC to the USB-Serial connector (CN27) with USB cable (MiniB, type-A).

4. Connect the debugger to the PC and the EVB. For details on connecting the debugger, refer to the document “Getting Started with Flexible Software Package”.

2.6 Host PC Setup

For details on how to install and use tools, refer to the document "Getting Started with Flexible Software Package".

2.6.1 Network Adapter Settings

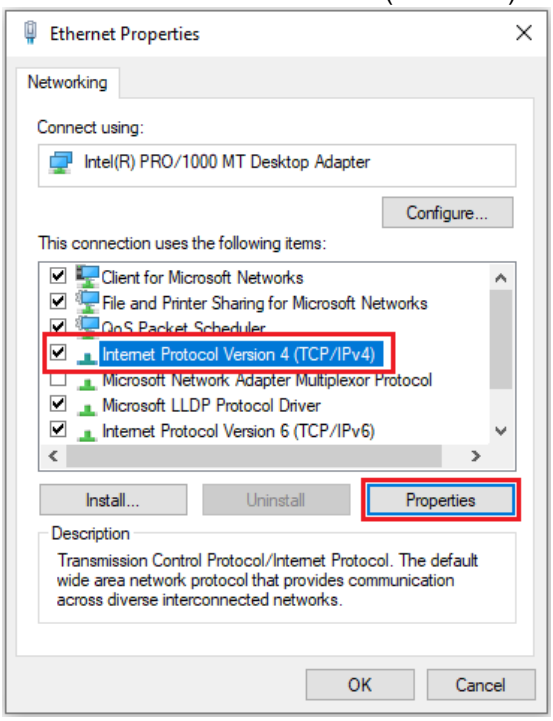
In order to use Mbed TLS client tool (ssl_server.exe) and Mbed TLS server tool (ssl_server.exe) to communicate to a target device, the host PC and the device must be connected to the same network. Table 2.16 lists the address settings for the device and the host PC.

Table 2.16 Network Address Settings

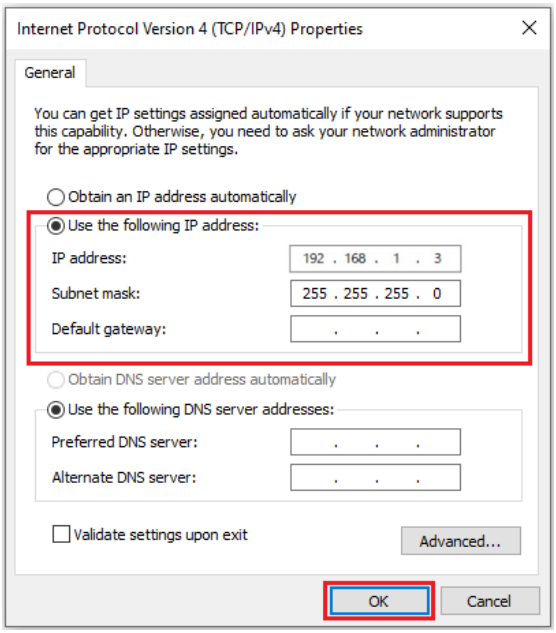
Device	IP Address	Net Mask
Target device	192.168.1.2	255.255.255.0
Host PC	192.168.1.3	255.255.255.0

Example host PC network adapter settings are shown below (example of settings on Windows 10).

1. Open the network adapter properties window on the host PC.
2. Select Internet Protocol Version (TCP/IPv4) and open the properties window.



3. In the Use following IP address section, enter settings for the IP address and subnet mask, then click the OK button.



2.7 Build and Run FreeRTOS+TCP Demo Project

IAR Embedded Workbench for ARM or e² studio is used as the development environment of FreeRTOS+TCP demo.
When you unzip RZ*_*_freertostcp_mbedtls_Rev210.zip, you will find folders named gcc and iccarm, which contain the projects for each compiler.

1. Open the FreeRTOS+TCP demo project and build the project.
   When using EWARM, open the following workspace, select the FreeRTOS+TCP demo project, generate FSP-related files and then build the project.

   ```
   iccarm\RZ*_*_freertostcp_mbedtls.eww
   ```

   After selecting a project, click “Tool -> FSP Smart Configurator” on tool bar, and click ‘Generate Project Content’ button to generate rz*, rz*_gen, rz*_cfg folders. Then build the project.

   When using e² studio, import the project (RZ*_*_freertostcp_mbedtls) below, generate FSP-related files and then build the project.

   ```
   gcc\
   ```

   After project import, click Configuration.xml and click the 'Generate Project Content' button to generate rz*, rz*_gen, and rz*_cfg folders. Then build the project.

   In the default configuration, the FreeRTOS+TCP demo is for Server mode and MbedTLS's HW accelarete disable. The configurations of the FreeRTOS+TCP demo and MbedTLS for RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H can be changed with the following files.
   src\tlsapp_mbedtls_config.h:

   ```
   #define TLSAPP_MBEDTLS_ENDPOINT_ROLL_SERVER
   ```

   If you want to run the application in client mode, set TLSAPP_MBEDTLS_ENDPOINT_ROLL_SERVER to undefined.

   oss\mbedtls\include\mbedtls\mbedtls_config.h:

   ```
   //#define TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE

   #if defined(TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE)
   #define PSA_CRYPTO_DRIVER_TEST
   #define PSA_CRYPTO_ACCELERATOR_DRIVER_PRESENT
   #endif /* TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE */
   ```

   If you want to eable MbedTLS acceleration by the RSIP driver, set TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE to defined.

2. Run the FreeRTOS+TCP demo
   2-1. Connect the Terminal Emulator (Tera Term) and RSK+ USB to Serial.
   2-1-1. Start Tera Term on your PC.

   2-1-2. Select the “New Connection” menu, select the “Serial” radio button on the New Connection dialog box, and select “COMx: USB Serial Device (COMx)” from the drop-down menu.

   2-1-3. Select “Setup” - “Terminal...” menu and check “Local echo” on the Terminal Setup dialog box.

   2-1-4. Select the “Setup” - “Serial port...” menu and select “115200” for “Speed:”.

2-2. To use the device as Client, start a command prompt on your PC and run the Server demo application (ssl_server.exe).

```
  > ssl_server.exe
```

2-3. Download the pre-built program to the device with the debugger and run it.
2-3-1. When using EWARM,
While the board and I-jet are connected, click on the "Download and debug" button in the "Project" toolbar.
The program will break at the first code in "main".
Press the "Go" button again to execute the program.

2-3-2. When using e^2 studio,
While the board and J-LINK are connected, in [Project Explorer] view, right click the node of the project to be debugged and select [Debug As]->[Renesas DBG Hardware Debugging] menu.

Click "Switch" in the "Confirm Perspective Switch" window.
The program will break at "system_init();"in startup_core.c .

Press the "Resume" button. The program will break at "main();"in main.c .

Press the "Resume" button again to execute the application program.

2-4. To use the device as Server, start a command prompt on your PC and run the Client demo application (ssl_client1.exe).

```
  > ssl_client1.exe
```

If the Client demo application is finished with an error and the following error message is output on the command prompt, the device is still initialization the TCP module.
Wait for approximately 5 seconds and run the Client demo application again.

```
  Last error was: -68 - NET - The connection to the given server / port
  failed
```

If the demo runs successfully, the following will be output to the PC console.

TeraTerm (To use the device as Client):

```
  /** Started sample application for FreeRTOS TCP Port with Mbed TLS. **/
    > Send Get Request to server:
   18 bytes written

  GET / HTTP/1.0

    < Receive Get Response from server:
   145 bytes read

  HTTP/1.0 200 OK
  Content-Type: text/html

  <h2>Mbed TLS Test</h2>
  <p>Successful connection using: TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256</p>
  /** Successfully Finished sample application for FreeRTOS TCP Port with Mbed
  TLS. **/
```

Command prompt (ssl_server.exe):

```
    . Seeding the random number generator... ok

   . Loading the server cert. and key... ok
   . Bind on https://192.168.1.3:65000/ ... ok
   . Setting up the SSL data.... ok
   . Waiting for a remote connection ... ok
  ok
   . Performing the SSL/TLS handshake... ok
   < Read from client: 18 bytes read

GET / HTTP/1.0



   > Write to client: 145 bytes written

HTTP/1.0 200 OK

Content-Type: text/html



<h2>Mbed TLS Test</h2>

<p>Successful connection using: TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256</p>


   . Closing the connection... ok
   . Waiting for a remote connection ...
```

TeraTerm (To use the device as Server):

```
/** Started sample application for FreeRTOS TCP Port with Mbed TLS. **/
  < Receive Get Request from client:
 18 bytes read

GET / HTTP/1.0

  > Send Get Response to client:
 145 bytes written

HTTP/1.0 200 OK
Content-Type: text/html

<h2>Mbed TLS Test</h2>
<p>Successful connection using: TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256</p>

/** Successfully Finished sample application for FreeRTOS TCP Port with Mbed
TLS. **/
```

Command prompt (ssl_client1.exe):

```
    . Seeding the random number generator... ok
    . Loading the CA root certificate ... ok (0 skipped)
    . Connecting to tcp/192.168.1.2/65000... ok
    . Setting up the SSL/TLS structure... ok
    . Performing the SSL/TLS handshake... ok
    . Verifying peer X.509 certificate... ok
    > Write to server: 18 bytes written

  GET / HTTP/1.0



    < Read from server: 145 bytes read

  HTTP/1.0 200 OK

  Content-Type: text/html



  <h2>Mbed TLS Test</h2>

  <p>Successful connection using: TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256</p>
```

2.8 Build and Run FreeRTOS+lwIP Demo Project

IAR Embedded Workbench for ARM or e^2 studio is used as the development environment of FreeRTOS+TCP demo.
When you unzip RZ*_*_lwIP_mbedtls_Rev210.zip, you will find folders named gcc and iccarm, which contain the projects for each compiler.

1. Open the FreeRTOS+lwIP demo project and build the project.
   When using EWARM, open the following workspace, select the FreeRTOS+lwIP demo project, generate FSP-related files and then build the project.

Project\rz*_rsk\lwip_RAM_single\ewarm\RZ*_*_lwIP_mbedtls.eww

   After selecting a project, click “Tool -> FSP Smart Configurator” on tool bar, and click ‘Generate Project Content’ button to generate rz*, rz*_gen, rz*_cfg folders. Then build the project.

   When using e^2 studio, import the project (RZ*_*_lwIP_mbedtls) below, generate FSP-related files and then build the project.

Project\rz*_rsk\lwip_RAM_single\e2studio

   After project import, click Configuration.xml and click the 'Generate Project Content' button to generate rz*, rz*_gen, and rz*_cfg folders. Then build the project.

   In the default configuration, the FreeRTOS+lwIP demo is for Server mode and MbedTLS's HW accelarete disable. The configurations of the FreeRTOS+lwIP demo and MbedTLS for RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H can be changed with the following files.
   common\renesas\application\tlsapp_mbedtls_config.h:

   ```
   #define TLSAPP_MBEDTLS_ENDPOINT_ROLL_SERVER
   ```

   If you want to run the application in client mode, set TLSAPP_MBEDTLS_ENDPOINT_ROLL_SERVER to undefined.

   common\oss\mbedtls\include\mbedtls\mbedtls_config.h:

   ```
   //#define TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE

   #if defined(TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE)
   #define PSA_CRYPTO_DRIVER_TEST
   #define PSA_CRYPTO_ACCELERATOR_DRIVER_PRESENT
   #endif /* TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE */
   ```

   If you want to eable MbedTLS acceleration by the RSIP driver, set TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE to defined.

2. Run the FreeRTOS+lwIP demo

2-1. Connect the Terminal Emulator (Tera Term) and RSK+ USB to Serial.

2-1-1. Start Tera Term on your PC.

2-1-2. Select the “New Connection” menu, select the “Serial” radio button on the New Connection dialog box, and select “COMx: USB Serial Device (COMx)” from the drop-down menu.

2-1-3. Select “Setup” - “Terminal...” menu and check “Local echo” on the Terminal Setup dialog box.

2-1-4. Select the “Setup” - “Serial port...” menu and select “115200” for “Speed:”.

2-2. To use the device as Client, start a command prompt on your PC and run the Server demo application (ssl_server.exe).

```
> ssl_server.exe
```

2-3. Download the pre-built program to the device with the debugger and run it.

2-3-1. When using EWARM,

While the board and I-jet are connected, click on the "Download and debug" button in the "Project" toolbar.

The program will break at the first code in "main".

Press the "Go" button again to execute the program.

2-3-2. When using e^2 studio,

While the board and J-LINK are connected, in [Project Explorer] view, right click the node of the project to be debugged and select [Debug As]->[Renesas DBG Hardware Debugging] menu.

Click "Switch" in the "Confirm Perspective Switch" window.

The program will break at "system_init();"in startup_core.c .

Press the "Resume" button. The program will break at "main();"in main.c .

Press the "Resume" button again to execute the application program.

2-4. To use the device as Server, start a command prompt on your PC and run the Client demo application (ssl_client1.exe).

```
> ssl_client1.exe
```

If the Client demo application is finished with an error and the following error message is output on the command prompt, the device is still initialization the TCP module.

Wait for approximately 5 seconds and run the Client demo application again.

```
Last error was: -68 - NET - The connection to the given server / port
failed
```

If the demo runs successfully, the following will be output to the PC consoles.

TeraTerm (To use the device as Client on RZ/T2M, RZ/T2L and RZ/N2L):

```
>>Info : Started Serial I/O interface.
>>Info : Initialized Ethernet network interface.
>>Info : Started Ethernet network interface.
>>Info : Initialized lwIP TCP/IP stack.
>>Info : /** Started sample application for lwIP Port with Mbed TLS. **/
>>Info : Set the callback function into TCP/IP stack.
>>Info : Created sample application task.
  > Send Get Request to server:
 18 bytes written

GET / HTTP/1.0

  < Receive Get Response from server:
 145 bytes read

HTTP/1.0 200 OK
Content-Type: text/html

<h2>Mbed TLS Test</h2>
<p>Successful connection using: TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256</p>

>>Info : /** Successfully Finished sample application for lwIP Port with
Mbed TLS. **/
```

TeraTerm (To use the device as Client on RZ/T2H and RZ/N2H):

```
Started Serial I/O interface.
>>Info : Initialized Ethernet network interface.
>>Info : Started Ethernet network interface.
>>Info : Initialized lwIP TCP/IP stack.
>>Info : /** Started sample application for lwIP Port with Mbed TLS. **/
>>Info : Set the callback function into TCP/IP stack.
>>Info :  18 bytes written

GET / HTTP/1.0

 145 bytes read

HTTP/1.0 200 OK
Content-Type: text/html

<h2>Mbed TLS Test</h2>
<p>Successful connection using: TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256</p>

>>Info :
```

Command prompt (ssl_server.exe):

```
    . Seeding the random number generator... ok

    . Loading the server cert. and key... ok
    . Bind on https://192.168.1.3:65000/ ... ok
    . Setting up the SSL data.... ok
    . Waiting for a remote connection ... ok
   ok
    . Performing the SSL/TLS handshake... ok
    < Read from client: 18 bytes read

  GET / HTTP/1.0



    > Write to client: 145 bytes written

  HTTP/1.0 200 OK

  Content-Type: text/html



  <h2>Mbed TLS Test</h2>

  <p>Successful connection using: TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256</p>


    . Closing the connection... ok
    . Waiting for a remote connection ...
```

TeraTerm (To use the device as Server on RZ/T2M, RZ/T2L and RZ/N2L):

```
  >>Info : Started Serial I/O interface.
  >>Info : Initialized Ethernet network interface.
  >>Info : Started Ethernet network interface.
  >>Info : Initialized lwIP TCP/IP stack.
  >>Info : /** Started sample application for lwIP Port with Mbed TLS. **/
  >>Info : Set the callback function into TCP/IP stack.
  >>Info : Created sample application task.
    < Receive Get Request from client:
   18 bytes read

  GET / HTTP/1.0

    > Send Get Response to client:
   145 bytes written

  HTTP/1.0 200 OK
  Content-Type: text/html

  <h2>Mbed TLS Test</h2>
  <p>Successful connection using: TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256</p>

  >>Info : /** Successfully Finished sample application for lwIP Port with
  Mbed TLS. **/
```

TeraTerm (To use the device as Server on RZ/T2H and RZ/N2H):

```
Started Serial I/O interface.
>>Info : Initialized Ethernet network interface.
>>Info : Started Ethernet network interface.
>>Info : Initialized lwIP TCP/IP stack.
>>Info : /** Started sample application for lwIP Port with Mbed TLS. **/
>>Info : Set the callback function into TCP/IP stack.
>>Info :  18 bytes read

GET / HTTP/1.0

 145 bytes written

HTTP/1.0 200 OK
Content-Type: text/html

<h2>Mbed TLS Test</h2>
<p>Successful connection using: TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256</p>

>>Info :
```

Command prompt (ssl_client1.exe):

```
  . Seeding the random number generator... ok
  . Loading the CA root certificate ... ok (0 skipped)
  . Connecting to tcp/192.168.1.2/65000... ok
  . Setting up the SSL/TLS structure... ok
  . Performing the SSL/TLS handshake... ok
  . Verifying peer X.509 certificate... ok
  > Write to server: 18 bytes written

GET / HTTP/1.0



  < Read from server: 145 bytes read

HTTP/1.0 200 OK

Content-Type: text/html



<h2>Mbed TLS Test</h2>

<p>Successful connection using: TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256</p>
```

3. Sample Program

3.1 FreeRTOS+TCP Demo

The FreeRTOS+TCP Demo uses FreeRTOS TCP Stack and Mbed TLS to establish a TLS connection with Client/Server Demo running on a PC, send and receive fixed Application Data, and output the sent and received Application Data is logged.

The FreeRTOS+TCP Demo uses FreeRTOS Plus's socket communication function for the TCP protocol, and uses Mbed TLS for the TLS protocol.

In the FreeRTOS+TCP Demo, the input/output API to FreeRTOS socket is registered as a callback to Mbed TLS to establish a data path between the TLS layer and the TCP layer.

The FreeRTOS+TCP Demo can switch between server and client.

See section 1.2.1 for details on the demonstration features.

3.1.1 File Structure and Configuration

Table 3.1 lists the main files contained in the FreeRTOS+TCP Demo project for each compiler.

Table 3.1 File Structure of FreeRTOS+TCP Demo

Folder Name	File Name	Description
RZ*_*_freertostcp_mbedtls_Rev2*0\		
├ gcc\		
│ ├	*.jlink, *.launch, *project,	Project files
│ ├	*.xml, rz*_cfg.txt	Flexible Software Package Files
│ ├ oss\mbedtls	*.c, *.h mbedtls_config.h	Mbed TLS src
│ ├ renesas\oss_deps\mbedtls_drv	*.c, *.h	Driver porting source code
│ ├ script\	*.ld	Memory allocation
│ └ src	*.c, *.h tlsapp_mbedtls_config.h	FreeRTOS+TCP Demo application
└ iccarm\		
├	*.eww, *.ewd, *.ewp	Project files
├	*.ipcf, *.xml, rz*_cfg.txt	Flexible Software Package Files
├ oss\mbedtls	*.c, *.h mbedtls_config.h	Mbed TLS src
├ renesas\oss_deps\mbedtls_drv	*.c, *.h	Driver porting source code
├ script\	*.icf	Memory allocation
└ src	*.c, *.h tlsapp_mbedtls_config.h	FreeRTOS+TCP Demo application

In the default configuration, the FreeRTOS+TCP demo is for Server mode and MbedTLS's HW accelarete disable. The configurations of the FreeRTOS+TCP demo and MbedTLS for RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H can be changed with the following files.

src\tlsapp_mbedtls_config.h:

```
#define TLSAPP_MBEDTLS_ENDPOINT_ROLL_SERVER
```

If you want to run the application in client mode, set TLSAPP_MBEDTLS_ENDPOINT_ROLL_SERVER to undefined.

oss\mbedtls\include\mbedtls\mbedtls_config.h:

```
//#define TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE

#if defined(TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE)
#define PSA_CRYPTO_DRIVER_TEST
#define PSA_CRYPTO_ACCELERATOR_DRIVER_PRESENT
#endif /* TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE */
```

If you want to enable MbedTLS acceleration by the RSIP driver, set TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE to defined.

3.1.2 Network Configuration

The network configuration for the FreeRTOS+TCP Demo is located at the top of the following file.

If you want to change the IP address, etc., change this setting.

src\net_thread_entry.c:

```
static  uint8_t ucIPAddress[ 4 ]         = {192, 168, 1, 2};
static  uint8_t ucNetMask[ 4 ]           = {255, 255, 255, 0};
static  uint8_t ucGatewayAddress[ 4 ]    = {192, 168, 1, 1};
static  uint8_t ucDNSServerAddress[ 4 ] = {192, 168, 1, 1};
```

3.2 FreeRTOS+lwIP Demo

The FreeRTOS+lwIP Demo uses lwIP and Mbed TLS to establish a TLS connection with Client/Server Demo running on a PC, send and receive fixed Application Data, and output the sent and received Application Data is logged.

The FreeRTOS+lwIP Demo uses lwIP's socket communication function for the TCP protocol, and uses Mbed TLS for the TLS protocol.

In the FreeRTOS+lwIP Demo, the input/output API to lwIP is registered as a callback to Mbed TLS to establish a data path between the TLS layer and the TCP layer.

The FreeRTOS+lwIP Demo can switch between server and client.

See section 1.2.1 for details on the demonstration features.

3.2.1 File Structure and Configuration

Table 3.2 lists the main files contained in the FreeRTOS+lwIP Demo project for each compiler.

Table 3.2 Main File Structure of FreeRTOS+lwIP Demo

Folder Name	File Name	Description
RZ*_*_lwIP_mbedtls_Rev2*0\		
├ common\		
│ ├ oss\mbedtls	*.c, *.h mbedtls_config.h	Mbed TLS src
│ ├ renesas\oss_deps\mbedtls_drv	*.c, *.h	Driver porting source code
│ └ renesas\application\	*.c, *.h tlsapp_mbedtls_config.h	FreeRTOS+lwIP Demo application
└ project\rz*_*\lwip_RAM_single\		
├ e2studio\		e² studio project files
└ ewarm\		EWARM project files

In the default configuration, the FreeRTOS+lwIP demo is for Server mode and MbedTLS's HW accelarete disable. The configurations of the FreeRTOS+lwIP demo and MbedTLS for RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H can be changed with the following files.

common\renesas\application\tlsapp_mbedtls_config.h:

```
#define TLSAPP_MBEDTLS_ENDPOINT_ROLL_SERVER
```

If you want to run the application in client mode, set TLSAPP_MBEDTLS_ENDPOINT_ROLL_SERVER to undefined.

common\oss\mbedtls\include\mbedtls\mbedtls_config.h:

```
//#define TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE

#if defined(TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE)
#define PSA_CRYPTO_DRIVER_TEST
#define PSA_CRYPTO_ACCELERATOR_DRIVER_PRESENT
#endif /* TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE */
```

If you want to enable MbedTLS acceleration by the RSIP driver, set TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE to defined.

3.2.2 Network Configuration

The network configuration for the FreeRTOS+lwIP Demo is located at the top of the following file.

If you want to change the IP address, etc., change this setting.

common\renesas\application\lwip_port_instance.c:

```
static uint8_t gp_lwip_port0_hostname[16] = "LWIP_NETIF0";
static lwip_port_netif_cfg_t g_lwip_port0_netif_cfg =
{
        .dhcp            = LWIP_PORT_DHCP_DISABLE,
        .ip_address      = PP_HTONL(LWIP_MAKEU32( 192, 168,   1, 2)),
        .subnet_mask     = PP_HTONL(LWIP_MAKEU32( 255, 255, 255, 0)),
        .gateway_address = PP_HTONL(LWIP_MAKEU32( 192, 168,   1, 1)),
        .p_host_name     = gp_lwip_port0_hostname
};
```

3.3 Client/Server Demo for PC

Client/Server Demo is a program that runs on a Host PC (Windows environment) included in the Mbed TLS. By using Mbed TLS, it establishes a TLS connection with the device side and sends and receives HTTP requests and responses through a secure communication path.

The Client/Server Demo can switch between server and client.

The Client/Server Demo is based on the Client/Server Demo program included in Mbed TLS. This demonstration use Visual Studio 2017 as the development tool.

The Client/Server Demo uses WinSock and Mbed TLS to establish a TLS connection with FreeRTOS+TCP demo or FreeRTOS+lwIP demo, send and receive fixed Application Data, and output the sent and received Application Data is logged.

See section 1.2.1 for details on the demonstration features.

3.3.1 File Structure and Configuration

Table 3.1 lists the main files contained in the Client/Server Demo project for each compiler.

The Visual Studio 2017 solution included in this demo contains a client (ssl_client1) and server (ssl_server) project.

Table 3.3 File Structure of Client/Server Demo

Folder Name	File Name	Description
MbedTLS_ClientServer_demo\		
├ 3rdparty\		Source code provided by Mbed TLS
├ include\		
├ library\		
├ programs\ssl\	ssl_client1.c, ssl_server.c	
├ tests\		
└ visualc\VS2017	mbedTLS.sln, *.vcxproj	Client/Server Demo solution file and project files

3.3.2 Network Configuration

The network configuration for the Client/Server Demo is located at the top of the following file.

If you want to change the IP address, etc., change this setting.

programs\ssl\ssl_client1.c:

```
#define TLSAPP_MBEDTLS_HOST_PORTNUMBER      "65000"
#define TLSAPP_MBEDTLS_HOST_IPADDRESS       "192.168.1.2"
#define TLSAPP_MBEDTLS_HOST_SERVER_NAME     "renesas.com"
```

programs\ssl\ssl_server.c:

```
#define TLSAPP_MBEDTLS_HOST_PORTNUMBER      "65000"
#define TLSAPP_MBEDTLS_HOST_IPADDRESS       "192.168.1.3"
#define TLSAPP_MBEDTLS_HOST_SERVER_NAME     "renesas.com"
```

3.4 Mbed TLS for RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H

The Mbed TLS for RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H makes the following changes to the original Mbed TLS to accelerate cryptographic operations using the RSIP driver.

- Enables PSA Crypto's hardware accelerator feature built into the original Mbed TLS.

- The cryptographic primitive module used by psa_crypto_driver_wrapper was changed from an existing built-in module to a hardware accelerator driver.

The hardware accelerator driver is implemented using the RSIP driver.

Mbed TLS for RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H is based on Mbed TLS 3.6.0 LTS:

https://github.com/Mbed-TLS/mbedtls/releases/tag/v3.6.0
https://github.com/Mbed-TLS/mbedtls/tree/v3.6.0

This sample program package uses Mbed TLS under Apache-2.0. For more information about the license information of Mbed TLS, please refer to the following:

https://github.com/Mbed-TLS/mbedtls/blob/v3.6.0/LICENSE

3.4.1 File Structure and Configuration

Table 3.4 lists the main files contained in the Mbed TLS program project for each compiler.

Table 3.4 File Structure of Mbed TLS program

Folder Name	File Name	Description
mbedtls-v3.6.0_with_rsip_rev210\		
├ oss\		
│ └ mbedtls\	*.c, *.h mbedtls_config.h	Mbedtls src
└ renesas\		
└ oss_deps\		
└ mbedtls_drv\	*.c, *.h	Driver porting source code

The configuration of the Mbed TLS for RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H is shown in Table 3.5. Only security support products can set TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE to define.

Table 3.5 Configurations for Mbed TLS (mbedtls_config.h)

Configuration items	Configurable values	Description
TLSAPP_MBEDTLS_RSIPDRIVER_ENABLE	defined	Mbed TLS for RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H with RSIP driver acceleration enabled.
	undefined	Default settings. Mbed TLS for RZ/T2M, RZ/T2L, RZ/T2H, RZ/N2L and RZ/N2H with RSIP driver acceleration disabled, i.e. full software Mbed TLS works.

3.4.2 How to use MbedTLS

The usage in FreeRTOS+TCP Demo and FreeRTOS+lwIP Demo can be referred to the implementation in the following files.

FreeRTOS+TCP Demo: src\tlsapp_mbedtls_thread_entry.c

FreeRTOS+lwIP Demo: common\renesas\application\tlsapp_mbedtls_lwip_main.c

TLS communication with MbedTLS involves first establishing a TLS connection, then sending and receiving encrypted data.

The TLS connection step involves initializing Mbed TLS using the various MbedTLS APIs and handshaking with the other end of the connection using mbedtls_ssl_handshake(). Various settings are required depending on the TLS functionality and mode of operation you wish to use, so please use the demo project and the MbedTLS examples presented below to implement them.

The mbedtls_ssl_write() and mbedtls_ssl_read() APIs can be used to send and receive encrypted data.

There is also an example of SSL/TLS provided by MbedTLS: https://github.com/Mbed-TLS/mbedtls/tree/v3.6.0/programs/

Please refer to the SSL/TLS examples in Readme.md for the explanation. The following information is excerpted from Readme.md:

SSL/TLS examples

SSL/TLS sample applications

- ssl/dtls_client.c: a simple DTLS client program, which sends one datagram to the server and reads one datagram in response.
- ssl/dtls_server.c: a simple DTLS server program, which expects one datagram from the client and writes one datagram in response. This program supports DTLS cookies for hello verification.
- ssl/mini_client.c: a minimalistic SSL client, which sends a short string and disconnects. This is primarily intended as a benchmark; for a better example of a typical TLS client, see ssl/ssl_client1.c.
- ssl/ssl_client1.c: a simple HTTPS client that sends a fixed request and displays the response.
- ssl/ssl_fork_server.c: a simple HTTPS server using one process per client to send a fixed response. This program requires a Unix/POSIX environment implementing the fork system call.
- ssl/ssl_mail_client.c: a simple SMTP-over-TLS or SMTP-STARTTLS client. This client sends an email with fixed content.
- ssl/ssl_pthread_server.c: a simple HTTPS server using one thread per client to send a fixed response. This program requires the pthread library.
- ssl/ssl_server.c: a simple HTTPS server that sends a fixed response. It serves a single client at a time.

SSL/TLS feature demonstrators

Note: unlike most of the other programs under the programs/ directory, these two programs are not intended as a basis for writing an application. They combine most of the features supported by the library, and most applications require only a few features. To write a new application, we recommended that you start with ssl_client1.c or ssl_server.c, and then look inside ssl/ssl_client2.c or ssl/ssl_server2.c to see how to use the specific features that your application needs.

- ssl/ssl_client2.c: an HTTPS client that sends a fixed request and displays the response, with options to select TLS protocol features and Mbed TLS library features.
- ssl/ssl_server2.c: an HTTPS server that sends a fixed response, with options to select TLS protocol features and Mbed TLS library features.

In addition to providing options for testing client-side features, the ssl_client2 program has options that allow you to trigger certain behaviors in the server. For example, there are options to select ciphersuites, or to force a renegotiation. These options are useful for testing the corresponding features in a TLS server. Likewise, ssl_server2 has options to activate certain behaviors that are useful for testing a TLS client.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jun 19, 2024	-	First edition issued
2.10	Sep 4, 2024	- p4,p5,p9-p12, p4	Added sample programs for RZ/T2L and RZ/N2L. Acceleration of cryptographic operations for certificate verification during the TLS handshake is now supported. Added support for key exchange acceleration with RSA.
2.20	Nov 29, 2024	- p7-p18 p18	Added sample programs for RZ/T2H Cortex®-R52 core. The Board Settings has been changed to explain it for each target device. Installation of the RSIP Driver Pack was no longer necessary for RZ/T devices.
2.30	Dec 23, 2024	- -	Added sample programs for RZ/N2H Cortex®-R52 core. Deleted how to install RSIP Driver Pack because it was no longer necessary for all devices.

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/.