APPLICATION NOTE

RX Family

R20AN0548EJ0117
Rev.1.17
Jan. 20, 2023

TSIP (Trusted Secure IP) Module Firmware Integration Technology (Binary version)

Introduction

This application note describes the use of the software drivers for utilizing the TSIP (Trusted Secure IP) and TSIP-Lite capabilities on the RX Family of microcontrollers. This software is called the TSIP driver.

The TSIP driver provides APIs for performing the cryptographic capabilities summarized in Table 1 and Table 2, as well as for securely performing firmware updates.

The TSIP driver is provided as a Firmware Integration Technology (FIT) module. For an overview of FIT, refer to the URL below.

https://www.renesas.com/us/en/products/software-tools/software-os-middleware-driver/software-package/fit.html

Target Devices

RX231 Group, RX23W Group, RX65N, RX651 Group, RX66T Group, RX671 Group, RX72M Group, RX72N Group, and RX72T Group

For information regarding the model names of products that have TSIP capability, refer to the user's manuals of the respective RX microcontrollers.

There is an application note describing the details of the TSIP driver.

This application note will be explained using the key attached to the sample program. The key for mass production needs to be newly generated. An application note with the key details is available.

We will provide the product to customers who will be adopting or plan to adopt a Renesas microcontroller. Please contact your local Renesas Electronics sales office or distributor.

https://www.renesas.com/contact/

Table 1 TSIP Cryptographic Algorithms

Cipher Type		Algorithms
Public key cryptography	Encryption/decryption	RSAES-PKCS1-v1_5(1024/2048 bit) *1 : RFC8017
	Signature generation/verification	RSASSA-PKCS1-v1_5(1024/2048 bit) *1. : RFC8017 ECDSA(ECC P-192/224/256/384) : FIPS186-4
	Key generation	RSA (1024/2048 bit) ECC P-192/224/256/384
Common key cryptography	AES	AES (128/256 bit) ECB/CBC/CTR : FIPS 197, SP800-38A
	DES	Triple-DES (56/56x2/56x3 bit) ECB/CBC : FIPS 46-3
	ARC4	ARC4 (2048 bit)
Hashing	SHA	SHA-1, SHA-256 : FIPS 180-4
	MD5	MD5 : RFC1321
Authenticated Encryption(AEAD)		GCM/CCM : FIPS 197, SP800-38D
Message authentication		CMAC (AES), : FIPS 197, SP800-38B GMAC : RFC4543 HMAC (SHA) : RFC2104
Pseudo-random bit generation		SP 800-90A
Random number generation		Tested with SP 800-22
TLS cooperation function	TLS1.2	TLS1.2 : RFC5246 Supporting cipher suite for TLS1.2 is below: TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
	TLS1.3	TLS1.3 : RFC8446 Supporting cipher suite for TLS1.3 is below*2: TLS_AES_128_GCM_SHA256 TLS_AES_128_CCM_SHA256
Key update function		AES, RSA, DES, ARC4, ECC, HMAC
Key exchange		ECDH P-256, ECDHE P-512 : SP800-56A, SP800-56C DH (2048 bit)
Key Wrap		AES (128/256 bit)

Notes: 1. Supported functions for RSA(3072/4096 bit) are signature verification and exponential remainder calculation using public key.

2. Applicable devices are the RX65N Group, RX651 Group, RX66N Group, RX72M Group, and RX72N Group.

Table 2 Cryptographic Algorithms

Cipher Type		Algorithms
Common key cryptography	AES	AES (128/256 bit) ECB /CBC/CTR : FIPS 197, SP800-38A
Authenticated Encryption(AEAD)		GCM/CCM : FIPS 197, SP800-38D
Message authentication		CMAC (AES), : FIPS 197, SP800-38B GMAC : RFC4543 HMAC (SHA) : RFC2104
Pseudo-random bit generation		SP 800-90A
Random number generation		Tested with SP 800-22
Key update function		AES
Key Wrap		AES (128/256 bit)

Note:

RFC 2104: HMAC: Keyed-Hashing for Message Authentication (rfc-editor.org)
RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2 (rfc-editor.org)
RFC 4543: The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH (rfc-editor.org)
RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2 (rfc-editor.org)
RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3 (rfc-editor.org)

FIPS 46-3, Data Encryption Standard (DES) (withdrawn May 19, 2005) (nist.gov)
FIPS186-4: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
NIST SP 800-38A, Recommendation for Block Cipher Modes of Operation Methods and Techniques
NIST SP 800-38-B Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication (nist.gov)
NIST SP 800-38D, Recommendationfor Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC
NIST SP800-56A: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Lograrithm Cryptography (nist.gov)
NIST SP800-56C: Recommendation for Key-Derivation Methods in Key-Establishment Schemes (nist.gov)
NIST SP800-22: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf
NIST SP800-90A: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

Contents

1. Overview

1.1 Terminology

Terms used in this document are defined below. For terms related to keys, refer to "Key Installation Process" (reproduced below as Figure 1-1) in the section on TSIP or security functions of the hardware manual of the MCU.

Table 1.1 Terminology

Term	Description	Key Installation Process
Key Injection	Injecting key generation information into the device at the factory.	-
Key Update	Injecting key generation information into the device in the field.	-
User Key	Under AES, DES, ARC4, and HMAC a common key set by the user. Under RSA, ECC, a public key or secret key set by the user.	Key-1
Encrypted Key	Key information generated by AES128-encrypting the user key using a Provisioning Key.	eKey-1
Key Index	Data consisting of key information, such as the user key, that has been converted into a form that is usable by the TSIP driver. The user key is converted into the Key Index.	Index-1 or Index-2
Provisioning Key	An AES128 common keyring set by the user and used to encrypt the user key with AES128 and add a MAC value.	Key-2
Encrypted Provisioning Key	Key information used by the TSIP to decrypt an Encrypted Key and convert it into a Key Index. The Encrypted Provisioning Key is wrapped provis key by DLM server.	Index-2
Update Key Ring	A keyring set by the user used to generate an Encrypted Key from the user key in Key Update. Key Index for the Update Key Ring must be generated beforehand by key injection in order to perform Key Update on the device.	-
Hardware Unique Key (HUK)	A device-specific encryption key that exists only inside TSIP.	-
DLM server	The Renesas key management server. "DLM server" is short for "device lifecycle management server." It is used for Provisioning Key wrapping.	-

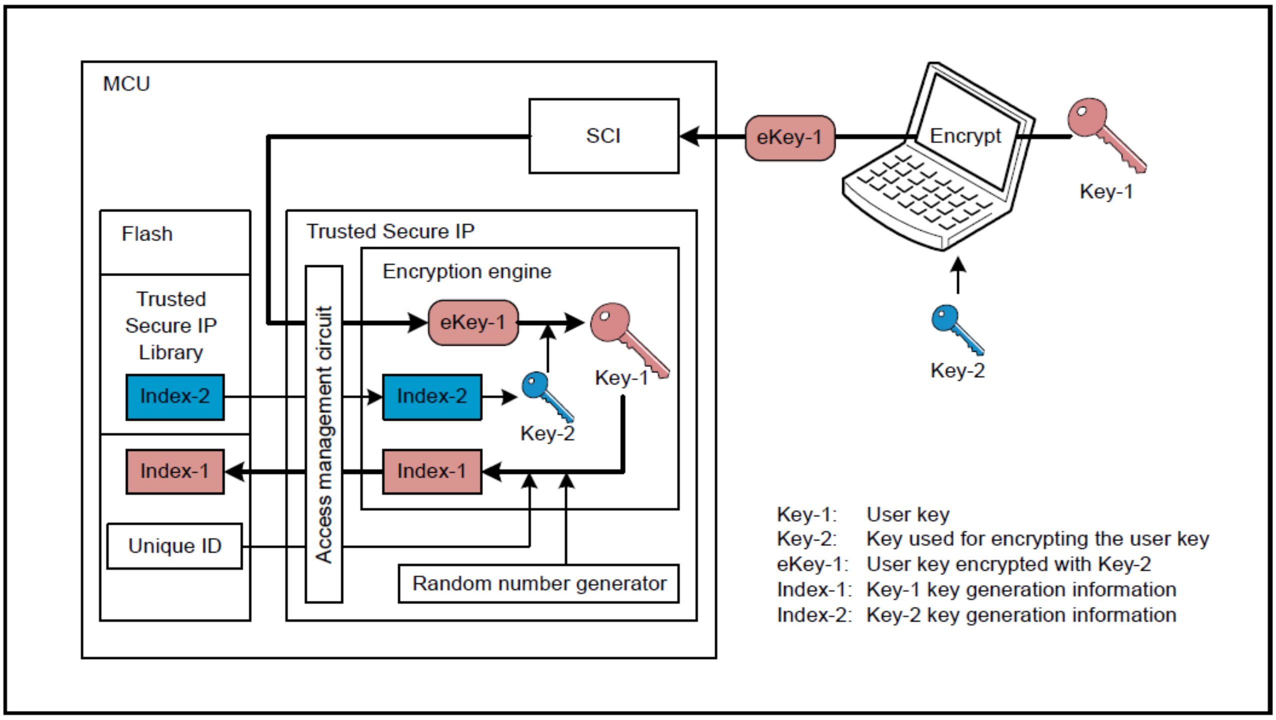

Figure 1-1 Key Installation Process
(RX65N Group, RX651 Group User's Manual: Hardware 52. Trusted Secure IP Figure 52.4)

1.2 Trusted Secure IP (TSIP)

The Trusted Secure IP (TSIP) block within the RX family creates a secure area inside the MCU by monitoring for unauthorized access attempts. It ensures that the encryption engine and encryption key can be utilized safely. The encryption key, the most important element in reliable and secure encryption, is linked to a unique ID and stored in the flash memory in a safe, undecipherable format.

Each TSIP devices include a safe area, which holds: an encryption engine, storage for raw keys, and a HUK, used to encrypt keys.

TSIP hardware generates Key Index from Encrypted Key inside the TSIP which is device-specific, and tied to a unique ID(HUK derived from it). Hence, the key from one device will not work on a different device. The TSIP driver software allows applications access to the TSIP hardware.

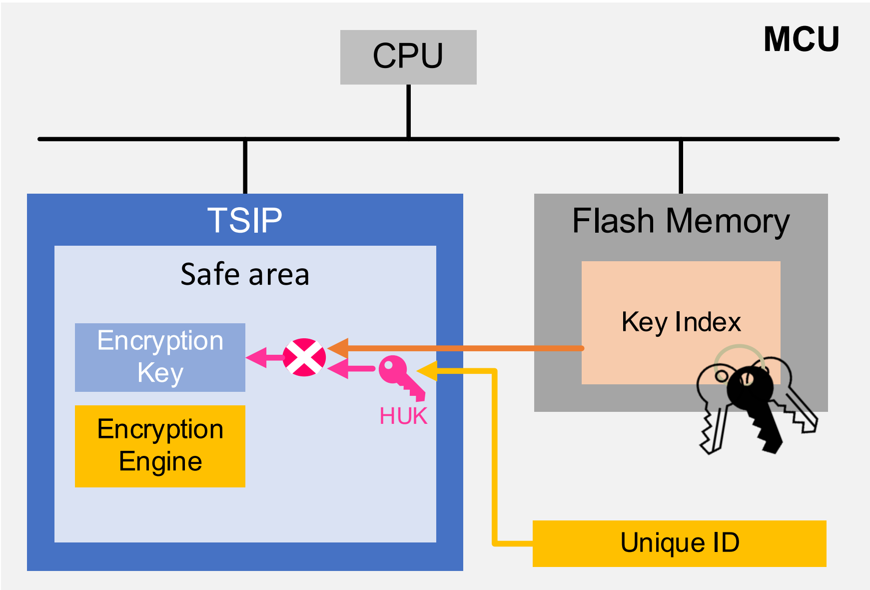

Figure 1.2 TSIP Hardware

1.3 Structure of Product Files

This product includes the files listed in Table1.2 below.

Table 1.2 Structure of Product Files

File/Directory (Bold) Names	Description
Readme.txt	Readme
RX_TSIP_SoftwareLicenceAgreement_ENG.pdf	Software License Agreement (English)
RX_TSIP_SoftwareLicenceAgreement_JPN.pdf	Software License Agreement (Japanese)
r20an0548ej0117-rx-tsip-security.pdf	TSIP driver Application Note (English)
r20an0548jj0117-rx-tsip-security.pdf	TSIP driver Application Note (Japanese)
reference_documents	Folder containing documentations such as how to use the FIT module with various integrated development environments
en	Folder containing documentations such as how to use the FIT module with various integrated development environments (English)
r01an1826ej0110-rx.pdf	How to add the FIT modules to CS+ Projects (English)
r01an1723eu0121-rx.pdf	How to add the FIT modules to e^2 studio Projects (English)
r20an0451es0140-e2studio-sc.pdf	Smart Configurator User Guide (English)
r01an5792ej0101-rx-tsip.pdf	Application note about how to use AES Cryptography with TSIP (English)
r01an5880ej0102-rx-tsip.pdf	Application note about how to implement TLS with TSIP (English)
ja	Folder containing documentations such as how to use the FIT module with various integrated development environments (Japanese)
r01an1826jj0110-rx.pdf	How to add the FIT modules to CS+ Projects (Japanese)
r01an1723ju0121-rx.pdf	How to add the FIT modules to e^2 studio Projects (Japanese)
r20an0451js0140-e2studio-sc.pdf	Smart Configurator User Guide (Japanese)
r01an5792jj0101-rx-tsip.pdf	Application note about how to use AES Cryptography with TSIP (Japanese)
r01an5880jj0102-rx-tsip.pdf	Application note about how to implement TLS with TSIP (Japanese)
FITModules	FIT module folder
r_tsip_rx_v1.17.l.zip	TSIP driver FIT Module
r_tsip_rx_v1.17.l.xml	TSIP driver FIT Module e^2 studio FIT plug-in XML file
r_tsip_rx_v1.17.l_extend.mdf	TSIP driver FIT Module Smart Configurator configuration file
FITDemos	Sample project folder
rx*XXX*_rsk_tsip_sample[*1]	Project showing the methods for writing and updating keys
rx65n_2mb_rsk_tsip_aes_sample	The sample indicates how to use AES cryptograpy in RX65N
rx72n_ek_tsip_aes_sample	The sample indicates how to use AES cryptograpy in RX72N
rx_tsip_freertos_mbedtls_sample	The sample indicates how to implement TLS
tool	
renesas_secure_flash_programmer	
Renesas Secure Flash Programmer.exe	The tool encrypts the key and user program.

Notes: 1. rxXXX contains the RX group name of the supported RSK boards.
Supported RSK boards: RX231, RX65N-2MB,RX66T,RX671,RX72N,RX72M,RX72T

1.4 Development Environment

The TSIP driver was developed using the environment shown below. When developing your own applications, use the versions of the software indicated below, or newer.

1. Integrated development environment
   Refer to the "Integrated development environment" item under 11.1, Confirmed Operation Environment.
2. C compiler
   Refer to the "C compiler" item under 11.1, Confirmed Operation Environment.
3. Emulator/debugger
   E1/E20/E2 Lite
4. Evaluation boards
   Refer to the "Board used" item under 11.1, Confirmed Operation Environment.
   All of the boards listed are special product versions with encryption functionality.
   Make sure to confirm the product model name before ordering. e^2 studio and CC-RX were used in combination for evaluation and to create the model project.

   The project conversion function can be used to convert projects from e^2 studio to CS+. If you encounter errors such as compiler errors, please contact your Renesas representative.

1.5 Code Size

The sizes of ROM, RAM and maximum stack usage associated with this module are listed below.

The values listed in the table below have been confirmed under the following conditions:

Module revision: r_tsip_rx rev1.16

Compiler version: Renesas Electronics C/C++ Compiler Package for RX Family V3.04.00
(integrated development environment default settings with "-lang = c99" option added)

GCC for Renesas RX 8.3.0.202104
(integrated development environment default settings with "-std=gnu99" option added)

IAR C/C++ Compiler for Renesas RX version 4.20.01
(integrated development environment default settings)

ROM, RAM, and Stack Code Sizes				
Device	Category	Memory Used		
		Renesas Compiler	GCC	IAR Compiler
TSIP-Lite	ROM	56,506 bytes	55,484 bytes	58,494 bytes
	RAM	804 bytes	804 bytes	804 bytes
	STACK	184 bytes	-	164 bytes
TSIP	ROM	421,410 bytes	403,822 bytes	416,595 bytes
	RAM	7,538 bytes	7,428 bytes	7,538 bytes
	STACK	1,684 bytes	-	1,376 bytes

1.6 Sections

The TSIP driver uses the default sections.

If user generates key files with Renesas Secure Flash Programmer, C_FIRMWARE_UPDATE_CONTROL section and C_FIRMWARE_UPDATE_CONTROL_BLOCK_MIRROR section are used. When the TSIP driver is added with Smart Configurator, these sections are set to the project. If these sections need to be changed, edit the section setting.

1.7 Performance

The performance information for TSIP-Lite drivers (RX231, RX23W, RX66T, RX72T) and TSIP drivers (RX65N, RX671, RX72M, RX72N) for each device group is shown below.

Performance is measured in cycles of ICLK, the core clock, and the operating clock PCLKB for TSIP-Lite and TSIP is set to ICLK : PCLKB = 2 : 1. The driver is built with the version listed in 5.1 Confirmed Operation Environment of CC-RX with optimization level 2.

1.7.1 RX231

Table 1.3 Performance of each APIs

API	Performance (Unit: cycle)
R_TSIP_Open	7,400,000
R_TSIP_Close	450
R_TSIP_GetVersion	30
R_TSIP_GenerateAes128KeyIndex	4,000
R_TSIP_GenerateAes256KeyIndex	4,400
R_TSIP_GenerateAes128RandomKeyIndex	2,300
R_TSIP_GenerateAes256RandomKeyIndex	3,100
R_TSIP_GenerateRandomNumber	940
R_TSIP_GenerateUpdateKeyRingKeyIndex	4,400
R_TSIP_UpdateAes128KeyIndex	3,600
R_TSIP_UpdateAes256KeyIndex	3,900

Table 1.4 Performance of Firmware Verify APIs

API	Performance (Unit: cycle)		
	2 KB processing	4 KB processing	6 KB processing
R_TSIP_VerifyFirmwareMAC	13,000	24,000	35,000

Table 1.5 Performance of AES

API	Performance (Unit: cycle)		
	16-byte processing	48-byte processing	80-byte processing
R_TSIP_Aes128EcbEncryptInit	1,400	1,400	1,400
R_TSIP_Aes128EcbEncryptUpdate	620	800	970
R_TSIP_Aes128EcbEncryptFinal	560	560	560
R_TSIP_Aes128EcbDecryptInit	1,400	1,400	1,400
R_TSIP_Aes128EcbDecryptUpdate	740	920	1,100
R_TSIP_Aes128EcbDecryptFinal	580	580	580
R_TSIP_Aes256EcbEncryptInit	1,700	1,700	1,700
R_TSIP_Aes256EcbEncryptUpdate	660	910	1,200
R_TSIP_Aes256EcbEncryptFinal	570	570	570
R_TSIP_Aes256EcbDecryptInit	1,700	1,700	1,700
R_TSIP_Aes256EcbDecryptUpdate	810	1,100	1,300
R_TSIP_Aes256EcbDecryptFinal	580	580	580
R_TSIP_Aes128CbcEncryptInit	1,400	1,400	1,400
R_TSIP_Aes128CbcEncryptUpdate	680	860	1,100
R_TSIP_Aes128CbcEncryptFinal	590	590	590
R_TSIP_Aes128CbcDecryptInit	1,400	1,400	1,400
R_TSIP_Aes128CbcDecryptUpdate	790	970	1,200
R_TSIP_Aes128CbcDecryptFinal	600	600	600
R_TSIP_Aes256CbcEncryptInit	1,700	1,700	1,700
R_TSIP_Aes256CbcEncryptUpdate	710	960	1,300
R_TSIP_Aes256CbcEncryptFinal	590	590	590
R_TSIP_Aes256CbcDecryptInit	1,700	1,700	1,700
R_TSIP_Aes256CbcDecryptUpdate	860	1,100	1,400
R_TSIP_Aes256CbcDecryptFinal	600	600	600

Table 1.6 Performance of GCM

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128GcmEncryptInit	5,500	5,500	5,500
R_TSIP_Aes128GcmEncryptUpdate	2,900	3,400	3,900
R_TSIP_Aes128GcmEncryptFinal	1,300	1,300	1,300
R_TSIP_Aes128GcmDecryptInit	5,500	5,500	5,500
R_TSIP_Aes128GcmDecryptUpdate	2,500	2,600	2,700
R_TSIP_Aes128GcmDecryptFinal	2,100	2,100	2,100
R_TSIP_Aes256GcmEncryptInit	6,200	6,200	6,200
R_TSIP_Aes256GcmEncryptUpdate	3,000	3,500	4,100
R_TSIP_Aes256GcmEncryptFinal	1,400	1,400	1,400
R_TSIP_Aes256GcmDecryptInit	6,200	6,200	6,200
R_TSIP_Aes256GcmDecryptUpdate	2,600	2,700	2,800
R_TSIP_Aes256GcmDecryptFinal	2,200	2,200	2,200

GCM performance was measured with parameters fixed as follows: ivec = 1,024 bits, AAD = 720 bits, and authentication tag = 128 bits.

Table 1.7 Performance of AES-CCM

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128CcmEncryptInit	2,700	2,700	2,700
R_TSIP_Aes128CcmEncryptUpdate	1,600	1,700	1,900
R_TSIP_Aes128CcmEncryptFinal	1,200	1,200	1,200
R_TSIP_Aes128CcmDecryptInit	2,500	2,500	2,500
R_TSIP_Aes128CcmDecryptUpdate	1,500	1,600	1,800
R_TSIP_Aes128CcmDecryptFinal	2,000	2,000	2,000
R_TSIP_Aes256CcmEncryptInit	3,000	3,000	3,000
R_TSIP_Aes256CcmEncryptUpdate	1,800	2,000	2,300
R_TSIP_Aes256CcmEncryptFinal	1,300	1,300	1,300
R_TSIP_Aes256CcmDecryptInit	3,000	3,000	3,000
R_TSIP_Aes256CcmDecryptUpdate	1,700	1,900	2,200
R_TSIP_Aes256CcmDecryptFinal	2,000	2,000	2,000

CCM performance was measured with parameters fixed as follows: nonce = 104 bits, AAD = 880 bits, and MAC = 128 bits.

Table 1.8 Performance of MAC (AES-CMAC)

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128CmacGenerateInit	920	920	920
R_TSIP_Aes128CmacGenerateUpdate	820	900	990
R_TSIP_Aes128CmacGenerateFinal	1,100	1,100	1,100
R_TSIP_Aes128CmacVerifyInit	910	920	920
R_TSIP_Aes128CmacVerifyUpdate	820	910	990
R_TSIP_Aes128CmacVerifyFinal	1,800	1,800	1,800
R_TSIP_Aes256CmacGenerateInit	1,300	1,300	1,300
R_TSIP_Aes256CmacGenerateUpdate	880	1,100	1,200
R_TSIP_Aes256CmacGenerateFinal	1,200	1,200	1,200
R_TSIP_Aes256CmacVerifyInit	1,300	1,300	1,300
R_TSIP_Aes256CmacVerifyUpdate	880	1,100	1,200
R_TSIP_Aes256CmacVerifyFinal	1,900	1,900	1,900

Table 1.9 Performance of AES Key Wrap

API	Performance (Unit: cycle)	
	Wrap target key = AES-128	Wrap target key = AES-256
R_TSIP_Aes128KeyWrap	9,600	16,000
R_TSIP_Aes256KeyWrap	11,000	17,000
R_TSIP_Aes128KeyUnwrap	12,000	18,000
R_TSIP_Aes256KeyUnwrap	13,000	19,000

1.7.2 RX23W

Table 1.10 Performance of each APIs

API	Performance (Unit: cycle)
R_TSIP_Open	7,400,000
R_TSIP_Close	670
R_TSIP_GetVersion	40
R_TSIP_GenerateAes128KeyIndex	4,400
R_TSIP_GenerateAes256KeyIndex	4,700
R_TSIP_GenerateAes128RandomKeyIndex	2,500
R_TSIP_GenerateAes256RandomKeyIndex	3,400
R_TSIP_GenerateRandomNumber	1,100
R_TSIP_GenerateUpdateKeyRingKeyIndex	4,700
R_TSIP_UpdateAes128KeyIndex	3,900
R_TSIP_UpdateAes256KeyIndex	4,200

Table 1.11 Performance of Firmware Verify APIs

API	Performance (Unit: cycle)		
	2 KB processing	4 KB processing	6 KB processing
R_TSIP_VerifyFirmwareMAC	13,000	24,000	35,000

Table 1.12 Performance of AES

API	Performance (Unit: cycle)		
	16-byte processing	48-byte processing	80-byte processing
R_TSIP_Aes128EcbEncryptInit	1,500	1,500	1,500
R_TSIP_Aes128EcbEncryptUpdate	750	920	1,200
R_TSIP_Aes128EcbEncryptFinal	650	650	650
R_TSIP_Aes128EcbDecryptInit	1,600	1,600	1,600
R_TSIP_Aes128EcbDecryptUpdate	860	1,100	1,300
R_TSIP_Aes128EcbDecryptFinal	670	670	670
R_TSIP_Aes256EcbEncryptInit	1,900	1,900	1,900
R_TSIP_Aes256EcbEncryptUpdate	780	1,100	1,300
R_TSIP_Aes256EcbEncryptFinal	670	670	670
R_TSIP_Aes256EcbDecryptInit	1,900	1,900	1,900
R_TSIP_Aes256EcbDecryptUpdate	930	1,200	1,500
R_TSIP_Aes256EcbDecryptFinal	690	690	690
R_TSIP_Aes128CbcEncryptInit	1,600	1,600	1,600
R_TSIP_Aes128CbcEncryptUpdate	820	1,000	1,200
R_TSIP_Aes128CbcEncryptFinal	690	690	690
R_TSIP_Aes128CbcDecryptInit	1,600	1,600	1,600
R_TSIP_Aes128CbcDecryptUpdate	930	1,200	1,300
R_TSIP_Aes128CbcDecryptFinal	700	700	700
R_TSIP_Aes256CbcEncryptInit	1,900	1,900	1,900
R_TSIP_Aes256CbcEncryptUpdate	860	1,100	1,400
R_TSIP_Aes256CbcEncryptFinal	700	700	700
R_TSIP_Aes256CbcDecryptInit	1,900	2,000	2,000
R_TSIP_Aes256CbcDecryptUpdate	1,000	1,300	1,500
R_TSIP_Aes256CbcDecryptFinal	720	720	720

Table 1.13 Performance of GCM

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128GcmEncryptInit	6,300	6,300	6,300
R_TSIP_Aes128GcmEncryptUpdate	3,400	4,000	4,500
R_TSIP_Aes128GcmEncryptFinal	1,500	1,500	1,500
R_TSIP_Aes128GcmDecryptInit	6,300	6,300	6,300
R_TSIP_Aes128GcmDecryptUpdate	2,900	3,000	3,100
R_TSIP_Aes128GcmDecryptFinal	2,400	2,400	2,400
R_TSIP_Aes256GcmEncryptInit	7,000	7,000	7,000
R_TSIP_Aes256GcmEncryptUpdate	3,500	4,100	4,700
R_TSIP_Aes256GcmEncryptFinal	1,600	1,600	1,600
R_TSIP_Aes256GcmDecryptInit	7,000	7,000	7,000
R_TSIP_Aes256GcmDecryptUpdate	3,000	3,100	3,200
R_TSIP_Aes256GcmDecryptFinal	2,400	2,400	2,400

GCM performance was measured with parameters fixed as follows: ivec = 1,024 bits, AAD = 720 bits, and authentication tag = 128 bits.

Table 1.14 Performance of AES-CCM

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128CcmEncryptInit	3,100	3,100	3,100
R_TSIP_Aes128CcmEncryptUpdate	1,800	2,000	2,200
R_TSIP_Aes128CcmEncryptFinal	1,500	1,500	1,500
R_TSIP_Aes128CcmDecryptInit	2,800	2,800	2,800
R_TSIP_Aes128CcmDecryptUpdate	1,700	1,900	2,000
R_TSIP_Aes128CcmDecryptFinal	2,300	2,300	2,300
R_TSIP_Aes256CcmEncryptInit	3,300	3,300	3,300
R_TSIP_Aes256CcmEncryptUpdate	2,000	2,300	2,500
R_TSIP_Aes256CcmEncryptFinal	1,500	1,500	1,500
R_TSIP_Aes256CcmDecryptInit	3,300	3,300	3,300
R_TSIP_Aes256CcmDecryptUpdate	1,900	2,200	2,400
R_TSIP_Aes256CcmDecryptFinal	2,300	2,300	2,300

CCM performance was measured with parameters fixed as follows: nonce = 104 bits, AAD = 880 bits, and MAC = 128 bits.

Table 1.15 Performance of MAC (AES-CMAC)

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128CmacGenerateInit	1,100	1,100	1,100
R_TSIP_Aes128CmacGenerateUpdate	960	1,100	1,200
R_TSIP_Aes128CmacGenerateFinal	1,300	1,300	1,300
R_TSIP_Aes128CmacVerifyInit	1,100	1,100	1,100
R_TSIP_Aes128CmacVerifyUpdate	950	1,100	1,200
R_TSIP_Aes128CmacVerifyFinal	2,100	2,100	2,100
R_TSIP_Aes256CmacGenerateInit	1,400	1,400	1,400
R_TSIP_Aes256CmacGenerateUpdate	1,100	1,200	1,300
R_TSIP_Aes256CmacGenerateFinal	1,400	1,400	1,400
R_TSIP_Aes256CmacVerifyInit	1,400	1,400	1,400
R_TSIP_Aes256CmacVerifyUpdate	1,100	1,200	1,300
R_TSIP_Aes256CmacVerifyFinal	2,200	2,200	2,200

Table 1.16 Performance of AES Key Wrap

API	Performance (Unit: cycle)	
	Wrap target key = AES-128	Wrap target key = AES-256
R_TSIP_Aes128KeyWrap	11,000	17,000
R_TSIP_Aes256KeyWrap	12,000	18,000
R_TSIP_Aes128KeyUnwrap	14,000	20,000
R_TSIP_Aes256KeyUnwrap	15,000	21,000

1.7.3 RX66T and RX72T

Table 1.17 Performance of each APIs

API	Performance (Unit: cycle)
R_TSIP_Open	7,400,000
R_TSIP_Close	290
R_TSIP_GetVersion	22
R_TSIP_GenerateAes128KeyIndex	4,000
R_TSIP_GenerateAes256KeyIndex	4,300
R_TSIP_GenerateAes128RandomKeyIndex	2,200
R_TSIP_GenerateAes256RandomKeyIndex	3,000
R_TSIP_GenerateRandomNumber	910
R_TSIP_GenerateUpdateKeyRingKeyIndex	4,300
R_TSIP_UpdateAes128KeyIndex	3,500
R_TSIP_UpdateAes256KeyIndex	3,900

Table 1.18 Performance of Firmware Verify APIs

API	Performance (Unit: cycle)		
	2 KB processing	4 KB processing	6 KB processing
R_TSIP_VerifyFirmwareMAC	12,000	24,000	35,000

Table 1.19 Performance of AES

API	Performance (Unit: cycle)		
	16-byte processing	48-byte processing	80-byte processing
R_TSIP_Aes128EcbEncryptInit	1,300	1,300	1,300
R_TSIP_Aes128EcbEncryptUpdate	560	750	920
R_TSIP_Aes128EcbEncryptFinal	520	510	510
R_TSIP_Aes128EcbDecryptInit	1,300	1,300	1,300
R_TSIP_Aes128EcbDecryptUpdate	680	860	1,100
R_TSIP_Aes128EcbDecryptFinal	520	520	520
R_TSIP_Aes256EcbEncryptInit	1,600	1,600	1,600
R_TSIP_Aes256EcbEncryptUpdate	610	850	1,100
R_TSIP_Aes256EcbEncryptFinal	520	510	510
R_TSIP_Aes256EcbDecryptInit	1,600	1,600	1,600
R_TSIP_Aes256EcbDecryptUpdate	750	1,000	1,300
R_TSIP_Aes256EcbDecryptFinal	530	520	520
R_TSIP_Aes128CbcEncryptInit	1,400	1,400	1,400
R_TSIP_Aes128CbcEncryptUpdate	630	810	980
R_TSIP_Aes128CbcEncryptFinal	540	530	530
R_TSIP_Aes128CbcDecryptInit	1,400	1,400	1,400
R_TSIP_Aes128CbcDecryptUpdate	730	910	1,100
R_TSIP_Aes128CbcDecryptFinal	540	540	540
R_TSIP_Aes256CbcEncryptInit	1,700	1,700	1,700
R_TSIP_Aes256CbcEncryptUpdate	660	910	1,200
R_TSIP_Aes256CbcEncryptFinal	540	540	540
R_TSIP_Aes256CbcDecryptInit	1,700	1,700	1,700
R_TSIP_Aes256CbcDecryptUpdate	800	1,100	1,300
R_TSIP_Aes256CbcDecryptFinal	550	550	550

Table 1.20 Performance of AES-GCM

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128GcmEncryptInit	5,200	5,200	5,200
R_TSIP_Aes128GcmEncryptUpdate	2,700	3,100	3,600
R_TSIP_Aes128GcmEncryptFinal	1,300	1,300	1,300
R_TSIP_Aes128GcmDecryptInit	5,200	5,200	5,200
R_TSIP_Aes128GcmDecryptUpdate	2,300	2,300	2,400
R_TSIP_Aes128GcmDecryptFinal	2,100	2,100	2,100
R_TSIP_Aes256GcmEncryptInit	5,900	5,900	5,900
R_TSIP_Aes256GcmEncryptUpdate	2,800	3,300	3,800
R_TSIP_Aes256GcmEncryptFinal	1,300	1,300	1,300
R_TSIP_Aes256GcmDecryptInit	5,900	5,900	5,900
R_TSIP_Aes256GcmDecryptUpdate	2,400	2,500	2,600
R_TSIP_Aes256GcmDecryptFinal	2,100	2,100	2,100

GCM performance was measured with parameters fixed as follows: ivec = 1,024 bits, AAD = 720 bits, and authentication tag = 128 bits.

Table 1.21 Performance of AES-CCM

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128CcmEncryptInit	2,500	2,500	2,500
R_TSIP_Aes128CcmEncryptUpdate	1,500	1,700	1,900
R_TSIP_Aes128CcmEncryptFinal	1,200	1,200	1,200
R_TSIP_Aes128CcmDecryptInit	2,300	2,300	2,300
R_TSIP_Aes128CcmDecryptUpdate	1,400	1,600	1,800
R_TSIP_Aes128CcmDecryptFinal	1,900	1,900	1,900
R_TSIP_Aes256CcmEncryptInit	2,900	2,900	2,900
R_TSIP_Aes256CcmEncryptUpdate	1,700	2,000	2,200
R_TSIP_Aes256CcmEncryptFinal	1,200	1,200	1,200
R_TSIP_Aes256CcmDecryptInit	2,900	2,900	2,900
R_TSIP_Aes256CcmDecryptUpdate	1,600	1,900	2,100
R_TSIP_Aes256CcmDecryptFinal	2,000	2,000	2,000

CCM performance was measured with parameters fixed as follows: nonce = 104 bits, AAD = 880 bits, and MAC = 128 bits.

Table 1.22 Performance of AES-CMAC

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128CmacGenerateInit	890	880	880
R_TSIP_Aes128CmacGenerateUpdate	730	810	900
R_TSIP_Aes128CmacGenerateFinal	1,100	1,100	1,100
R_TSIP_Aes128CmacVerifyInit	880	880	880
R_TSIP_Aes128CmacVerifyUpdate	720	810	900
R_TSIP_Aes128CmacVerifyFinal	1,800	1,800	1,800
R_TSIP_Aes256CmacGenerateInit	1,200	1,200	1,200
R_TSIP_Aes256CmacGenerateUpdate	800	930	1,100
R_TSIP_Aes256CmacGenerateFinal	1,200	1,200	1,200
R_TSIP_Aes256CmacVerifyInit	1,200	1,200	1,200
R_TSIP_Aes256CmacVerifyUpdate	800	920	1,100
R_TSIP_Aes256CmacVerifyFinal	1,800	1,800	1,800

Table 1.23 Performance of AES Key Wrap

API	Performance (Unit: cycle)	
	Wrap target key = AES-128	Wrap target key = AES-256
R_TSIP_Aes128KeyWrap	9,400	16,000
R_TSIP_Aes256KeyWrap	11,000	17,000
R_TSIP_Aes128KeyUnwrap	12,000	18,000
R_TSIP_Aes256KeyUnwrap	13,000	19,000

1.7.4 RX65N

Table 1.24 Performance of each APIs

API	Performance (Unit: cycle)
R_TSIP_Open	5,800,000
R_TSIP_Close	460
R_TSIP_GetVersion	30
R_TSIP_GenerateAes128KeyIndex	2,700
R_TSIP_GenerateAes256KeyIndex	2,800
R_TSIP_GenerateAes128RandomKeyIndex	1,500
R_TSIP_GenerateAes256RandomKeyIndex	2,100
R_TSIP_GenerateRandomNumber	670
R_TSIP_GenerateUpdateKeyRingKeyIndex	2,800
R_TSIP_UpdateAes128KeyIndex	2,300
R_TSIP_UpdateAes256KeyIndex	2,400

Table 1.25 Performance of Firmware Verify APIs

API	Performance (Unit: cycle)		
	8 KB processing	16 KB processing	24 KB processing
R_TSIP_VerifyFirmwareMAC	22,000	42,000	63,000

Table 1.26 Performance of AES

API	Performance (Unit: cycle)		
	16-byte processing	48-byte processing	80-byte processing
R_TSIP_Aes128EcbEncryptInit	1,700	1,700	1,700
R_TSIP_Aes128EcbEncryptUpdate	520	660	840
R_TSIP_Aes128EcbEncryptFinal	450	450	450
R_TSIP_Aes128EcbDecryptInit	1,700	1,700	1,700
R_TSIP_Aes128EcbDecryptUpdate	590	730	910
R_TSIP_Aes128EcbDecryptFinal	460	460	460
R_TSIP_Aes256EcbEncryptInit	1,800	1,800	1,800
R_TSIP_Aes256EcbEncryptUpdate	540	690	870
R_TSIP_Aes256EcbEncryptFinal	440	440	440
R_TSIP_Aes256EcbDecryptInit	1,800	1,800	1,800
R_TSIP_Aes256EcbDecryptUpdate	610	750	930
R_TSIP_Aes256EcbDecryptFinal	470	470	470
R_TSIP_Aes128CbcEncryptInit	1,700	1,700	1,700
R_TSIP_Aes128CbcEncryptUpdate	590	730	900
R_TSIP_Aes128CbcEncryptFinal	480	480	480
R_TSIP_Aes128CbcDecryptInit	1,700	1,700	1,700
R_TSIP_Aes128CbcDecryptUpdate	660	790	970
R_TSIP_Aes128CbcDecryptFinal	490	500	500
R_TSIP_Aes256CbcEncryptInit	1,900	1,900	1,900
R_TSIP_Aes256CbcEncryptUpdate	590	740	920
R_TSIP_Aes256CbcEncryptFinal	480	480	480
R_TSIP_Aes256CbcDecryptInit	1,900	1,900	1,900
R_TSIP_Aes256CbcDecryptUpdate	680	820	1,000
R_TSIP_Aes256CbcDecryptFinal	490	490	490

Table 1.27 Performance of GCM

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128GcmEncryptInit	5,600	5,600	5,600
R_TSIP_Aes128GcmEncryptUpdate	2,100	2,200	2,300
R_TSIP_Aes128GcmEncryptFinal	1,400	1,400	1,400
R_TSIP_Aes128GcmDecryptInit	5,500	5,500	5,500
R_TSIP_Aes128GcmDecryptUpdate	2,100	2,200	2,300
R_TSIP_Aes128GcmDecryptFinal	2,200	2,200	2,200
R_TSIP_Aes256GcmEncryptInit	5,500	5,500	5,500
R_TSIP_Aes256GcmEncryptUpdate	2,200	2,300	2,400
R_TSIP_Aes256GcmEncryptFinal	1,100	1,100	1,100
R_TSIP_Aes256GcmDecryptInit	5,500	5,500	5,500
R_TSIP_Aes256GcmDecryptUpdate	2,200	2,300	2,300
R_TSIP_Aes256GcmDecryptFinal	2,000	2,000	2,000

GCM performance was measured with parameters fixed as follows: ivec = 1,024 bits, AAD = 720 bits, and authentication tag = 128 bits.

Table 1.28 Performance of AES-CCM

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128CcmEncryptInit	3,100	3,100	3,100
R_TSIP_Aes128CcmEncryptUpdate	1,200	1,300	1,400
R_TSIP_Aes128CcmEncryptFinal	940	940	940
R_TSIP_Aes128CcmDecryptInit	3,200	3,200	3,200
R_TSIP_Aes128CcmDecryptUpdate	1,100	1,200	1,300
R_TSIP_Aes128CcmDecryptFinal	2,000	2,000	2,000
R_TSIP_Aes256CcmEncryptInit	2,400	2,400	2,400
R_TSIP_Aes256CcmEncryptUpdate	1,200	1,300	1,400
R_TSIP_Aes256CcmEncryptFinal	990	990	990
R_TSIP_Aes256CcmDecryptInit	2,400	2,400	2,400
R_TSIP_Aes256CcmDecryptUpdate	1,100	1,200	1,300
R_TSIP_Aes256CcmDecryptFinal	2,100	2,100	2,100

CCM performance was measured with parameters fixed as follows: nonce = 104 bits, AAD = 880 bits, and MAC = 128 bits.

Table 1.29 Performance of MAC (AES-CMAC)

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128CmacGenerateInit	1,200	1,200	1,200
R_TSIP_Aes128CmacGenerateUpdate	670	720	760
R_TSIP_Aes128CmacGenerateFinal	800	800	800
R_TSIP_Aes128CmacVerifyInit	1,200	1,200	1,200
R_TSIP_Aes128CmacVerifyUpdate	680	720	770
R_TSIP_Aes128CmacVerifyFinal	1,700	1,700	1,700
R_TSIP_Aes256CmacGenerateInit	1,300	1,300	1,300
R_TSIP_Aes256CmacGenerateUpdate	720	760	810
R_TSIP_Aes256CmacGenerateFinal	830	830	830
R_TSIP_Aes256CmacVerifyInit	1,300	1,300	1,300
R_TSIP_Aes256CmacVerifyUpdate	710	750	810
R_TSIP_Aes256CmacVerifyFinal	1,700	1,700	1,700

Table 1.30 Performance of AES Key Wrap

API	Performance (Unit: cycle)	
	Wrap target key = AES-128	Wrap target key = AES-256
R_TSIP_Aes128KeyWrap	8,300	13,000
R_TSIP_Aes256KeyWrap	8,400	14,000
R_TSIP_Aes128KeyUnwrap	9,300	14,000
R_TSIP_Aes256KeyUnwrap	9,500	15,000

Table 1.31 Performance of Common API (TDES Key Index Generation)

API	Performance (Unit: cycle)
R_TSIP_GenerateTdesKeyIndex	2,800
R_TSIP_GenerateTdesRandomKeyIndex	2,100
R_TSIP_UpdateTdesKeyIndex	2,400

Table 1.32 Performance of TDES

API	Performance (Unit: cycle)		
	16-byte processing	48-byte processing	80-byte processing
R_TSIP_TdesEcbEncryptInit	1,100	1,100	1,100
R_TSIP_TdesEcbEncryptUpdate	560	800	1,100
R_TSIP_TdesEcbEncryptFinal	450	450	450
R_TSIP_TdesEcbDecryptInit	1,100	1,100	1,100
R_TSIP_TdesEcbDecryptUpdate	590	830	1,100
R_TSIP_TdesEcbDecryptFinal	470	470	470
R_TSIP_TdesCbcEncryptInit	1,200	1,200	1,200
R_TSIP_TdesCbcEncryptUpdate	630	870	1,200
R_TSIP_TdesCbcEncryptFinal	480	480	480
R_TSIP_TdesCbcDecryptInit	1,200	1,200	1,200
R_TSIP_TdesCbcDecryptUpdate	650	900	1,200
R_TSIP_TdesCbcDecryptFinal	490	490	490

Table 1.33 Performance of Common API (ARC4 Key Index Generation)

API	Performance (Unit: cycle)
R_TSIP_GenerateArc4KeyIndex	4,600
R_TSIP_GenerateArc4RandomKeyIndex	11,000
R_TSIP_UpdateArc4KeyIndex	4,200

Table 1.34 Performance of ARC4

API	Performance (Unit: cycle)		
	Message size=16byte	Message size=48byte	Message size=80byte
R_TSIP_Arc4EncryptInit	2,100	2,100	2,100
R_TSIP_Arc4EncryptUpdate	490	630	810
R_TSIP_Arc4EncryptFinal	330	330	330
R_TSIP_Arc4DecryptInit	2,100	2,100	2,100
R_TSIP_Arc4DecryptUpdate	490	630	810
R_TSIP_Arc4DecryptFinal	320	330	330

Table 1.35 Performance of Common API (RSA Key Index Generation)

API	Performance (Unit: cycle)
R_TSIP_GenerateRsa1024PublicKeyIndex	38,000
R_TSIP_GenerateRsa1024PrivateKeyIndex	39,000
R_TSIP_GenerateRsa2048PublicKeyIndex	140,000
R_TSIP_GenerateRsa2048PrivateKeyIndex	140,000
R_TSIP_GenerateRsa1024RandomKeyIndex *1	75,000,000
R_TSIP_GenerateRsa2048RandomKeyIndex *1	540,000,000
R_TSIP_UpdateRsa1024PublicKeyIndex	38,000
R_TSIP_UpdateRsa1024PrivateKeyIndex	39,000
R_TSIP_UpdateRsa2048PublicKeyIndex	140,000
R_TSIP_UpdateRsa2048PrivateKeyIndex	140,000

Note 1. Average value at 10 runs.

Table 1.36 Performance of RSASSA-PKCS-v1_5 Signature Generation/Verification (HASH = SHA1)

API	Performance (Unit: cycle)		
	Message size=1byte	Message size=128byte	Message size=256byte
R_TSIP_RsassaPkcs1024SignatureGenerate	1,300,000	1,300,000	1,300,000
R_TSIP_RsassaPkcs1024SignatureVerification	18,000	19,000	20,000
R_TSIP_RsassaPkcs2048SignatureGenerate	27,000,000	27,000,000	27,000,000
R_TSIP_RsassaPkcs2048SignatureVerification	140,000	140,000	140,000

Table 1.37 Performance of RSASSA-PKCS-v1_5 Signature Generation/Verification (HASH = SHA256)

API	Performance (Unit: cycle)		
	Message size=1byte	Message size=128byte	Message size=256byte
R_TSIP_RsassaPkcs1024SignatureGenerate	1,300,000	1,300,000	1,300,000
R_TSIP_RsassaPkcs1024SignatureVerification	18,000	19,000	20,000
R_TSIP_RsassaPkcs2048SignatureGenerate	27,000,000	27,000,000	27,000,000
R_TSIP_RsassaPkcs2048SignatureVerification	140,000	140,000	140,000

Table 1.38 Performance of RSASSA-PKCS-v1_5 Signature Generation/Verification (HASH = MD5)

API	Performance (Unit: cycle)		
	Message size=1byte	Message size=128byte	Message size=256byte
R_TSIP_RsassaPkcs1024SignatureGenerate	1,300,000	1,300,000	1,300,000
R_TSIP_RsassaPkcs1024SignatureVerification	18,000	19,000	19,000
R_TSIP_RsassaPkcs2048SignatureGenerate	27,000,000	27,000,000	27,000,000
R_TSIP_RsassaPkcs2048SignatureVerification	140,000	140,000	140,000

Table 1.39 Performance of RSAES-PKCS1-v1_5 Encryption/Decryption with 1,024-Bit Key Size

API	Performance (Unit: cycle)	
	Message size=1byte	Message size=117byte
R_TSIP_RsaesPkcs1024Encrypt	23,000	17,000
R_TSIP_RsaesPkcs1024Decrypt	1,300,000	1,300,000

Table 1.40 Performance of RSAES-PKCS1-v1_5 Encryption/Decryption with 2,048-Bit Key Size

API	Performance (Unit: cycle)	
	Message size=1byte	Message size=245byte
R_TSIP_RsaesPkcs2048Encrypt	150,000	140,000
R_TSIP_RsaesPkcs2048Decrypt	27,000,000	27,000,000

Table 1.41 Performance of HASH (SHA1)

API	Performance (Unit: cycle)		
	128-byte processing	192-byte processing	256-byte processing
R_TSIP_Sha1Init	130	130	130
R_TSIP_Sha1Update	1,600	1,800	2,000
R_TSIP_Sha1Final	830	830	830

Table 1.42 Performance of HASH (SHA256)

API	Performance (Unit: cycle)		
	128-byte processing	192-byte processing	256-byte processing
R_TSIP_Sha256Init	140	140	140
R_TSIP_Sha256Update	1,600	1,800	2,000
R_TSIP_Sha256Final	840	840	840

Table 1.43 Performance of HASH (MD5)

API	Performance (Unit: cycle)		
	128-byte processing	192-byte processing	256-byte processing
R_TSIP_Md5Init	120	120	120
R_TSIP_Md5Update	1,500	1,700	1,900
R_TSIP_Md5Final	780	780	780

Table 1.44 Performance of Common API (HMAC Key Index Generation)

API	Performance (Unit: cycle)
R_TSIP_GenerateSha1HmacKeyIndex	3,000
R_TSIP_GenerateSha256HmacKeyIndex	3,000
R_TSIP_UpdateSha1HmacKeyIndex	2,700
R_TSIP_UpdateSha256HmacKeyIndex	2,700

Table 1.45 Performance of HMAC (SHA1)

API	Performance (Unit: cycle)		
	128-byte processing	192-byte processing	256-byte processing
R_TSIP_Sha1HmacGenerateInit	1,400	1,400	1,400
R_TSIP_Sha1HmacGenerateUpdate	980	1,300	1,500
R_TSIP_Sha1HmacGenerateFinal	2,000	2,000	2,000
R_TSIP_Sha1HmacVerifyInit	1,400	1,400	1,400
R_TSIP_Sha1HmacVerifyUpdate	980	1,300	1,500
R_TSIP_Sha1HmacVerifyFinal	3,700	3,700	3,700

Table 1.46 Performance of HMAC (SHA256)

API	Performance (Unit: cycle)		
	128-byte processing	192-byte processing	256-byte processing
R_TSIP_Sha256HmacGenerateInit	1,800	1,800	1,800
R_TSIP_Sha256HmacGenerateUpdate	920	1,200	1,400
R_TSIP_Sha256HmacGenerateFinal	2,000	2,000	2,000
R_TSIP_Sha256HmacVerifyInit	1,800	1,800	1,800
R_TSIP_Sha256HmacVerifyUpdate	920	1,200	1,400
R_TSIP_Sha256HmacVerifyFinal	3,700	3,700	3,700

Table 1.47 Performance of Common API (ECC Key Index Generation)

API	Performance (Unit: cycle)
R_TSIP_GenerateEccP192PublicKeyIndex	3,300
R_TSIP_GenerateEccP224PublicKeyIndex	3,300
R_TSIP_GenerateEccP256PublicKeyIndex	3,300
R_TSIP_GenerateEccP384PublicKeyIndex	3,400
R_TSIP_GenerateEccP192PrivateKeyIndex	3,000
R_TSIP_GenerateEccP224PrivateKeyIndex	3,000
R_TSIP_GenerateEccP256PrivateKeyIndex	3,000
R_TSIP_GenerateEccP384PrivateKeyIndex	2,900
R_TSIP_GenerateEccP192RandomKeyIndex *1	150,000
R_TSIP_GenerateEccP224RandomKeyIndex *1	160,000
R_TSIP_GenerateEccP256RandomKeyIndex *1	160,000
R_TSIP_GenerateEccP384RandomKeyIndex *1	1,100,000
R_TSIP_UpdateEccP192PublicKeyIndex	3,000
R_TSIP_UpdateEccP224PublicKeyIndex	3,000
R_TSIP_UpdateEccP256PublicKeyIndex	3,000
R_TSIP_UpdateEccP384PublicKeyIndex	3,100
R_TSIP_UpdateEccP192PrivateKeyIndex	2,700
R_TSIP_UpdateEccP224PrivateKeyIndex	2,700
R_TSIP_UpdateEccP256PrivateKeyIndex	2,700
R_TSIP_UpdateEccP384PrivateKeyIndex	2,600

Note 1. Average value at 10 runs.

Table 1.48 Performance of ECDSA Signature Generation/Verification

API	Performance (Unit: cycle)		
	Message size=1byte	**Message size=128byte**	**Message size=256byte**
R_TSIP_EcdsaP192SignatureGenerate	180,000	180,000	180,000
R_TSIP_EcdsaP224SignatureGenerate	180,000	180,000	180,000
R_TSIP_EcdsaP256SignatureGenerate	180,000	190,000	190,000
R_TSIP_EcdsaP384SignatureGenerate*1	1,200,000		
R_TSIP_EcdsaP192SignatureVerification	330,000	340,000	340,000
R_TSIP_EcdsaP224SignatureVerification	360,000	360,000	360,000
R_TSIP_EcdsaP256SignatureVerification	360,000	360,000	360,000
R_TSIP_EcdsaP384SignatureVerification*1	2,300,000		

Note 1. Not include SHA384 calculation.

Table 1.49 Performance of Key Exchange

API	Performance (Unit: cycle)
R_TSIP_EcdhP256Init	60
R_TSIP_EcdhP256ReadPublicKey	360,000
R_TSIP_EcdhP256MakePublicKey	340,000
R_TSIP_EcdhP256CalculateSharedSecretIndex	380,000
R_TSIP_EcdhP256KeyDerivation	3,800
R_TSIP_EcdheP512KeyAgreement	3,400,000
R_TSIP_Rsa2048DhKeyAgreement	53,000,000

Key exchange performance (without KeyAgreement) was measured with parameters fixed as follows: key exchange format = ECDHE and derived key type = AES-128.

1.7.5 RX671

Table 1.50 Performance of each APIs

API	Performance (Unit: cycle)
R_TSIP_Open	5,400,000
R_TSIP_Close	310
R_TSIP_GetVersion	22
R_TSIP_GenerateAes128KeyIndex	2,100
R_TSIP_GenerateAes256KeyIndex	2,200
R_TSIP_GenerateAes128RandomKeyIndex	1,200
R_TSIP_GenerateAes256RandomKeyIndex	1,700
R_TSIP_GenerateRandomNumber	540
R_TSIP_GenerateUpdateKeyRingKeyIndex	2,200
R_TSIP_UpdateAes128KeyIndex	1,800
R_TSIP_UpdateAes256KeyIndex	2,000

Table 1.51 Performance of Firmware Verify APIs

API	Performance (Unit: cycle)		
	8 KB processing	16 KB processing	24 KB processing
R_TSIP_VerifyFirmwareMAC	17,000	34,000	50,000

Table 1.52 Performance of AES

API	Performance (Unit: cycle)		
	16-byte processing	48-byte processing	80-byte processing
R_TSIP_Aes128EcbEncryptInit	1,300	1,200	1,200
R_TSIP_Aes128EcbEncryptUpdate	390	490	620
R_TSIP_Aes128EcbEncryptFinal	320	310	310
R_TSIP_Aes128EcbDecryptInit	1,300	1,300	1,300
R_TSIP_Aes128EcbDecryptUpdate	450	560	690
R_TSIP_Aes128EcbDecryptFinal	320	320	320
R_TSIP_Aes256EcbEncryptInit	1,400	1,400	1,400
R_TSIP_Aes256EcbEncryptUpdate	400	510	640
R_TSIP_Aes256EcbEncryptFinal	320	310	310
R_TSIP_Aes256EcbDecryptInit	1,400	1,400	1,400
R_TSIP_Aes256EcbDecryptUpdate	470	580	710
R_TSIP_Aes256EcbDecryptFinal	330	330	330
R_TSIP_Aes128CbcEncryptInit	1,300	1,300	1,300
R_TSIP_Aes128CbcEncryptUpdate	430	540	670
R_TSIP_Aes128CbcEncryptFinal	340	330	330
R_TSIP_Aes128CbcDecryptInit	1,300	1,300	1,300
R_TSIP_Aes128CbcDecryptUpdate	490	600	730
R_TSIP_Aes128CbcDecryptFinal	340	340	340
R_TSIP_Aes256CbcEncryptInit	1,400	1,400	1,400
R_TSIP_Aes256CbcEncryptUpdate	450	570	700
R_TSIP_Aes256CbcEncryptFinal	340	340	340
R_TSIP_Aes256CbcDecryptInit	1,400	1,400	1,400
R_TSIP_Aes256CbcDecryptUpdate	520	640	770
R_TSIP_Aes256CbcDecryptFinal	350	350	350

Table 1.53 Performance of GCM

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128GcmEncryptInit	4,100	4,100	4,100
R_TSIP_Aes128GcmEncryptUpdate	1,600	1,700	1,700
R_TSIP_Aes128GcmEncryptFinal	950	940	940
R_TSIP_Aes128GcmDecryptInit	4,100	4,100	4,100
R_TSIP_Aes128GcmDecryptUpdate	1,600	1,600	1,700
R_TSIP_Aes128GcmDecryptFinal	1,500	1,500	1,500
R_TSIP_Aes256GcmEncryptInit	4,200	4,100	4,100
R_TSIP_Aes256GcmEncryptUpdate	1,600	1,700	1,800
R_TSIP_Aes256GcmEncryptFinal	830	820	820
R_TSIP_Aes256GcmDecryptInit	4,100	4,100	4,100
R_TSIP_Aes256GcmDecryptUpdate	1,600	1,700	1,700
R_TSIP_Aes256GcmDecryptFinal	1,500	1,500	1,500

GCM performance was measured with parameters fixed as follows: ivec = 1,024 bits, AAD = 720 bits, and authentication tag = 128 bits.

Table 1.54 Performance of AES-CCM

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128CcmEncryptInit	2,300	2,300	2,300
R_TSIP_Aes128CcmEncryptUpdate	870	950	1,100
R_TSIP_Aes128CcmEncryptFinal	760	750	750
R_TSIP_Aes128CcmDecryptInit	2,400	2,400	2,400
R_TSIP_Aes128CcmDecryptUpdate	810	870	950
R_TSIP_Aes128CcmDecryptFinal	1,500	1,500	1,500
R_TSIP_Aes256CcmEncryptInit	1,900	1,900	1,900
R_TSIP_Aes256CcmEncryptUpdate	940	1,100	1,200
R_TSIP_Aes256CcmEncryptFinal	770	770	770
R_TSIP_Aes256CcmDecryptInit	1,900	1,900	1,900
R_TSIP_Aes256CcmDecryptUpdate	850	930	1,100
R_TSIP_Aes256CcmDecryptFinal	1,500	1,500	1,500

CCM performance was measured with parameters fixed as follows: nonce = 104 bits, AAD = 880 bits, and MAC = 128 bits.

Table 1.55 Performance of MAC (AES-CMAC)

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128CmacGenerateInit	880	870	870
R_TSIP_Aes128CmacGenerateUpdate	490	520	560
R_TSIP_Aes128CmacGenerateFinal	630	620	620
R_TSIP_Aes128CmacVerifyInit	870	870	870
R_TSIP_Aes128CmacVerifyUpdate	490	530	570
R_TSIP_Aes128CmacVerifyFinal	1,300	1,300	1,300
R_TSIP_Aes256CmacGenerateInit	980	980	980
R_TSIP_Aes256CmacGenerateUpdate	520	550	600
R_TSIP_Aes256CmacGenerateFinal	650	630	630
R_TSIP_Aes256CmacVerifyInit	970	970	970
R_TSIP_Aes256CmacVerifyUpdate	510	550	600
R_TSIP_Aes256CmacVerifyFinal	1,300	1,300	1,300

Table 1.56 Performance of AES Key Wrap

API	Performance (Unit: cycle)	
	Wrap target key = AES-128	Wrap target key = AES-256
R_TSIP_Aes128KeyWrap	6,400	10,000
R_TSIP_Aes256KeyWrap	6,600	11,000
R_TSIP_Aes128KeyUnwrap	7,200	11,000
R_TSIP_Aes256KeyUnwrap	7,400	12,000

Table 1.57 Performance of Common API (TDES Key Index Generation)

API	Performance (Unit: cycle)
R_TSIP_GenerateTdesKeyIndex	2,200
R_TSIP_GenerateTdesRandomKeyIndex	1,700
R_TSIP_UpdateTdesKeyIndex	2,000

Table 1.58 Performance of TDES

API	Performance (Unit: cycle)		
	16-byte processing	48-byte processing	80-byte processing
R_TSIP_TdesEcbEncryptInit	800	790	790
R_TSIP_TdesEcbEncryptUpdate	430	610	800
R_TSIP_TdesEcbEncryptFinal	320	300	300
R_TSIP_TdesEcbDecryptInit	800	800	800
R_TSIP_TdesEcbDecryptUpdate	450	640	830
R_TSIP_TdesEcbDecryptFinal	330	320	320
R_TSIP_TdesCbcEncryptInit	850	840	840
R_TSIP_TdesCbcEncryptUpdate	480	670	860
R_TSIP_TdesCbcEncryptFinal	320	320	320
R_TSIP_TdesCbcDecryptInit	850	850	850
R_TSIP_TdesCbcDecryptUpdate	500	700	890
R_TSIP_TdesCbcDecryptFinal	340	340	340

Table 1.59 Performance of Common API (ARC4 Key Index Generation)

API	Performance (Unit: cycle)
R_TSIP_GenerateArc4KeyIndex	3,900
R_TSIP_GenerateArc4RandomKeyIndex	8,600
R_TSIP_UpdateArc4KeyIndex	3,700

Table 1.60 Performance of ARC4

API	Performance (Unit: cycle)		
	Message size=16byte	Message size=48byte	Message size=80byte
R_TSIP_Arc4EncryptInit	1,800	1,800	1,800
R_TSIP_Arc4EncryptUpdate	360	480	610
R_TSIP_Arc4EncryptFinal	230	230	230
R_TSIP_Arc4DecryptInit	1,800	1,800	1,800
R_TSIP_Arc4DecryptUpdate	360	480	610
R_TSIP_Arc4DecryptFinal	230	230	230

Table 1.61 Performance of Common API (RSA Key Index Generation)

API	Performance (Unit: cycle)
R_TSIP_GenerateRsa1024PublicKeyIndex	37,000
R_TSIP_GenerateRsa1024PrivateKeyIndex	38,000
R_TSIP_GenerateRsa2048PublicKeyIndex	140,000
R_TSIP_GenerateRsa2048PrivateKeyIndex	140,000
R_TSIP_GenerateRsa1024RandomKeyIndex *1	67,000,000
R_TSIP_GenerateRsa2048RandomKeyIndex *1	360,000,000
R_TSIP_UpdateRsa1024PublicKeyIndex	37,000
R_TSIP_UpdateRsa1024PrivateKeyIndex	38,000
R_TSIP_UpdateRsa2048PublicKeyIndex	140,000
R_TSIP_UpdateRsa2048PrivateKeyIndex	140,000

Note 1. Average value at 10 runs.

Table 1.62 Performance of RSASSA-PKCS-v1_5 Signature Generation/Verification (HASH = SHA1)

API	Performance (Unit: cycle)		
	Message size=1byte	Message size=128byte	Message size=256byte
R_TSIP_RsassaPkcs1024SignatureGenerate	1,300,000	1,300,000	1,300,000
R_TSIP_RsassaPkcs1024SignatureVerification	16,000	18,000	18,000
R_TSIP_RsassaPkcs2048SignatureGenerate	27,000,000	27,000,000	27,000,000
R_TSIP_RsassaPkcs2048SignatureVerification	140,000	140,000	140,000

Table 1.63 Performance of RSASSA-PKCS-v1_5 Signature Generation/Verification (HASH = SHA256)

API	Performance (Unit: cycle)		
	Message size=1byte	Message size=128byte	Message size=256byte
R_TSIP_RsassaPkcs1024SignatureGenerate	1,300,000	1,300,000	1,300,000
R_TSIP_RsassaPkcs1024SignatureVerification	16,000	18,000	18,000
R_TSIP_RsassaPkcs2048SignatureGenerate	27,000,000	27,000,000	27,000,000
R_TSIP_RsassaPkcs2048SignatureVerification	140,000	140,000	140,000

Table 1.64 Performance of RSASSA-PKCS-v1_5 Signature Generation/Verification (HASH = MD5)

API	Performance (Unit: cycle)		
	Message size=1byte	Message size=128byte	Message size=256byte
R_TSIP_RsassaPkcs1024SignatureGenerate	1,300,000	1,300,000	1,300,000
R_TSIP_RsassaPkcs1024SignatureVerification	16,000	17,000	18,000
R_TSIP_RsassaPkcs2048SignatureGenerate	27,000,000	27,000,000	27,000,000
R_TSIP_RsassaPkcs2048SignatureVerification	140,000	140,000	140,000

Table 1.65 Performance of RSAES-PKCS1-v1_5 Encryption/Decryption with 1,024-Bit Key Size

API	Performance (Unit: cycle)	
	Message size=1byte	Message size=117byte
R_TSIP_RsaesPkcs1024Encrypt	20,000	16,000
R_TSIP_RsaesPkcs1024Decrypt	1,300,000	1,300,000

Table 1.66 Performance of RSAES-PKCS1-v1_5 Encryption/Decryption with 2,048-Bit Key Size

API	Performance (Unit: cycle)	
	Message size=1byte	Message size=245byte
R_TSIP_RsaesPkcs2048Encrypt	150,000	140,000
R_TSIP_RsaesPkcs2048Decrypt	27,000,000	27,000,000

Table 1.67 Performance of HASH (SHA1)

API	Performance (Unit: cycle)		
	128-byte processing	192-byte processing	256-byte processing
R_TSIP_Sha1Init	110	110	110
R_TSIP_Sha1Update	1,300	1,500	1,700
R_TSIP_Sha1Final	660	660	660

Table 1.68 Performance of HASH (SHA256)

API	Performance (Unit: cycle)		
	128-byte processing	192-byte processing	256-byte processing
R_TSIP_Sha256Init	120	120	120
R_TSIP_Sha256Update	1,300	1,500	1,600
R_TSIP_Sha256Final	670	670	670

Table 1.69 Performance of HASH (MD5)

API	Performance (Unit: cycle)		
	128-byte processing	192-byte processing	256-byte processing
R_TSIP_Md5Init	94	96	96
R_TSIP_Md5Update	1,200	1,300	1,500
R_TSIP_Md5Final	630	630	630

Table 1.70 Performance of Common API (HMAC Key Index Generation)

API	Performance (Unit: cycle)
R_TSIP_GenerateSha1HmacKeyIndex	2,300
R_TSIP_GenerateSha256HmacKeyIndex	2,300
R_TSIP_UpdateSha1HmacKeyIndex	2,100
R_TSIP_UpdateSha256HmacKeyIndex	2,000

Table 1.71 Performance of HMAC (SHA1)

API	Performance (Unit: cycle)		
	128-byte processing	192-byte processing	256-byte processing
R_TSIP_Sha1HmacGenerateInit	1,100	1,100	1,100
R_TSIP_Sha1HmacGenerateUpdate	810	1,100	1,300
R_TSIP_Sha1HmacGenerateFinal	1,600	1,600	1,600
R_TSIP_Sha1HmacVerifyInit	1,100	1,100	1,100
R_TSIP_Sha1HmacVerifyUpdate	800	1,100	1,300
R_TSIP_Sha1HmacVerifyFinal	2,800	2,800	2,800

Table 1.72 Performance of HMAC (SHA256)

API	Performance (Unit: cycle)		
	128-byte processing	192-byte processing	256-byte processing
R_TSIP_Sha256HmacGenerateInit	1,400	1,300	1,300
R_TSIP_Sha256HmacGenerateUpdate	740	910	1,100
R_TSIP_Sha256HmacGenerateFinal	1,600	1,600	1,600
R_TSIP_Sha256HmacVerifyInit	1,300	1,300	1,300
R_TSIP_Sha256HmacVerifyUpdate	730	910	1,100
R_TSIP_Sha256HmacVerifyFinal	2,700	2,700	2,700

Table 1.73 Performance of Common API (ECC Key Index Generation)

API	Performance (Unit: cycle)
R_TSIP_GenerateEccP192PublicKeyIndex	2,600
R_TSIP_GenerateEccP224PublicKeyIndex	2,600
R_TSIP_GenerateEccP256PublicKeyIndex	2,600
R_TSIP_GenerateEccP384PublicKeyIndex	2,800
R_TSIP_GenerateEccP192PrivateKeyIndex	2,300
R_TSIP_GenerateEccP224PrivateKeyIndex	2,300
R_TSIP_GenerateEccP256PrivateKeyIndex	2,300
R_TSIP_GenerateEccP384PrivateKeyIndex	2,300
R_TSIP_GenerateEccP192RandomKeyIndex *1	140,000
R_TSIP_GenerateEccP224RandomKeyIndex *1	150,000
R_TSIP_GenerateEccP256RandomKeyIndex *1	150,000
R_TSIP_GenerateEccP384RandomKeyIndex *1	1,100,000
R_TSIP_UpdateEccP192PublicKeyIndex	2,400
R_TSIP_UpdateEccP224PublicKeyIndex	2,300
R_TSIP_UpdateEccP256PublicKeyIndex	2,300
R_TSIP_UpdateEccP384PublicKeyIndex	2,500
R_TSIP_UpdateEccP192PrivateKeyIndex	2,100
R_TSIP_UpdateEccP224PrivateKeyIndex	2,000
R_TSIP_UpdateEccP256PrivateKeyIndex	2,000
R_TSIP_UpdateEccP384PrivateKeyIndex	2,100

Note 1. Average value at 10 runs.

Table 1.74 Performance of ECDSA Signature Generation/Verification

API	Performance (Unit: cycle)		
	Message size=1byte	Message size=128byte	Message size=256byte
R_TSIP_EcdsaP192SignatureGenerate	170,000	170,000	170,000
R_TSIP_EcdsaP224SignatureGenerate	170,000	170,000	170,000
R_TSIP_EcdsaP256SignatureGenerate	170,000	180,000	170,000
R_TSIP_EcdsaP384SignatureGenerate*1	1,200,000		
R_TSIP_EcdsaP192SignatureVerification	310,000	310,000	310,000
R_TSIP_EcdsaP224SignatureVerification	330,000	330,000	330,000
R_TSIP_EcdsaP256SignatureVerification	330,000	340,000	330,000
R_TSIP_EcdsaP384SignatureVerification*1	2,200,000		

Note 1. Not include SHA384 calculation.

Table 1.75 Performance of Key Exchange

API	Performance (Unit: cycle)
R_TSIP_EcdhP256Init	42
R_TSIP_EcdhP256ReadPublicKey	340,000
R_TSIP_EcdhP256MakePublicKey	310,000
R_TSIP_EcdhP256CalculateSharedSecretIndex	360,000
R_TSIP_EcdhP256KeyDerivation	3,000
R_TSIP_EcdheP512KeyAgreement	3,300,000
R_TSIP_Rsa2048DhKeyAgreement	53,000,000

Key exchange performance (without KeyAgreement) was measured with parameters fixed as follows: key exchange format = ECDHE and derived key type = AES-GCM-128.

1.7.6 RX72M and RX72N

Table 1.76 Performance of each APIs

API	Performance (Unit: cycle)
R_TSIP_Open	6,300,000
R_TSIP_Close	310
R_TSIP_GetVersion	20
R_TSIP_GenerateAes128KeyIndex	2,200
R_TSIP_GenerateAes256KeyIndex	2,300
R_TSIP_GenerateAes128RandomKeyIndex	1,300
R_TSIP_GenerateAes256RandomKeyIndex	1,800
R_TSIP_GenerateRandomNumber	560
R_TSIP_GenerateUpdateKeyRingKeyIndex	2,300
R_TSIP_UpdateAes128KeyIndex	1,900
R_TSIP_UpdateAes256KeyIndex	2,100

Table 1.77 Performance of Firmware Verify APIs

API	Performance (Unit: cycle)		
	8 KB processing	16 KB processing	24 KB processing
R_TSIP_VerifyFirmwareMAC	19,000	38,000	56,000

Table 1.78 Performance of AES

API	Performance (Unit: cycle)		
	16-byte processing	48-byte processing	80-byte processing
R_TSIP_Aes128EcbEncryptInit	1,300	1,300	1,300
R_TSIP_Aes128EcbEncryptUpdate	390	510	640
R_TSIP_Aes128EcbEncryptFinal	340	340	340
R_TSIP_Aes128EcbDecryptInit	1,300	1,300	1,300
R_TSIP_Aes128EcbDecryptUpdate	450	570	700
R_TSIP_Aes128EcbDecryptFinal	350	350	350
R_TSIP_Aes256EcbEncryptInit	1,400	1,400	1,400
R_TSIP_Aes256EcbEncryptUpdate	400	530	660
R_TSIP_Aes256EcbEncryptFinal	330	330	330
R_TSIP_Aes256EcbDecryptInit	1,400	1,400	1,400
R_TSIP_Aes256EcbDecryptUpdate	480	600	740
R_TSIP_Aes256EcbDecryptFinal	340	340	340
R_TSIP_Aes128CbcEncryptInit	1,400	1,400	1,400
R_TSIP_Aes128CbcEncryptUpdate	440	560	700
R_TSIP_Aes128CbcEncryptFinal	360	360	360
R_TSIP_Aes128CbcDecryptInit	1,400	1,400	1,400
R_TSIP_Aes128CbcDecryptUpdate	500	610	750
R_TSIP_Aes128CbcDecryptFinal	370	370	370
R_TSIP_Aes256CbcEncryptInit	1,500	1,500	1,500
R_TSIP_Aes256CbcEncryptUpdate	460	580	720
R_TSIP_Aes256CbcEncryptFinal	360	360	360
R_TSIP_Aes256CbcDecryptInit	1,500	1,500	1,500
R_TSIP_Aes256CbcDecryptUpdate	530	650	790
R_TSIP_Aes256CbcDecryptFinal	370	370	370

Table 1.79 Performance of AES-GCM

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128GcmEncryptInit	4,400	4,400	4,400
R_TSIP_Aes128GcmEncryptUpdate	1,600	1,700	1,800
R_TSIP_Aes128GcmEncryptFinal	1,100	1,100	1,100
R_TSIP_Aes128GcmDecryptInit	4,300	4,300	4,300
R_TSIP_Aes128GcmDecryptUpdate	1,600	1,700	1,800
R_TSIP_Aes128GcmDecryptFinal	1,700	1,700	1,700
R_TSIP_Aes256GcmEncryptInit	4,300	4,300	4,300
R_TSIP_Aes256GcmEncryptUpdate	1,600	1,700	1,800
R_TSIP_Aes256GcmEncryptFinal	860	860	860
R_TSIP_Aes256GcmDecryptInit	4,300	4,300	4,300
R_TSIP_Aes256GcmDecryptUpdate	1,700	1,700	1,800
R_TSIP_Aes256GcmDecryptFinal	1,500	1,500	1,500

GCM performance was measured with parameters fixed as follows: ivec = 1,024 bits, AAD = 720 bits, and authentication tag = 128 bits.

Table 1.80 Performance of AES-CCM

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128CcmEncryptInit	2,400	2,400	2,400
R_TSIP_Aes128CcmEncryptUpdate	900	970	1,100
R_TSIP_Aes128CcmEncryptFinal	750	750	750
R_TSIP_Aes128CcmDecryptInit	2,500	2,500	2,500
R_TSIP_Aes128CcmDecryptUpdate	820	900	980
R_TSIP_Aes128CcmDecryptFinal	1,500	1,500	1,500
R_TSIP_Aes256CcmEncryptInit	2,000	2,000	2,000
R_TSIP_Aes256CcmEncryptUpdate	960	1,100	1,200
R_TSIP_Aes256CcmEncryptFinal	800	800	800
R_TSIP_Aes256CcmDecryptInit	2,000	2,000	2,000
R_TSIP_Aes256CcmDecryptUpdate	860	950	1,100
R_TSIP_Aes256CcmDecryptFinal	1,600	1,600	1,600

CCM performance was measured with parameters fixed as follows: nonce = 104 bits, AAD = 880 bits, and MAC = 128 bits.

Table 1.81 Performance of AES-CMAC

API	Performance (Unit: cycle)		
	48-byte processing	64-byte processing	80-byte processing
R_TSIP_Aes128CmacGenerateInit	920	910	920
R_TSIP_Aes128CmacGenerateUpdate	490	530	570
R_TSIP_Aes128CmacGenerateFinal	630	620	620
R_TSIP_Aes128CmacVerifyInit	910	920	920
R_TSIP_Aes128CmacVerifyUpdate	490	530	570
R_TSIP_Aes128CmacVerifyFinal	1,300	1,300	1,300
R_TSIP_Aes256CmacGenerateInit	1,100	1,100	1,100
R_TSIP_Aes256CmacGenerateUpdate	520	560	600
R_TSIP_Aes256CmacGenerateFinal	660	660	660
R_TSIP_Aes256CmacVerifyInit	1,100	1,100	1,100
R_TSIP_Aes256CmacVerifyUpdate	530	570	610
R_TSIP_Aes256CmacVerifyFinal	1,300	1,300	1,300

Table 1.82 Performance of AES Key Wrap

API	Performance (Unit: cycle)	
	Wrap target key = AES-128	Wrap target key = AES-256
R_TSIP_Aes128KeyWrap	6,500	11,000
R_TSIP_Aes256KeyWrap	6,800	11,000
R_TSIP_Aes128KeyUnwrap	7,400	12,000
R_TSIP_Aes256KeyUnwrap	7,600	12,000

Table 1.83 Performance of Common API (TDES Key Index Generation)

API	Performance (Unit: cycle)
R_TSIP_GenerateTdesKeyIndex	2,300
R_TSIP_GenerateTdesRandomKeyIndex	1,800
R_TSIP_UpdateTdesKeyIndex	2,100

Table 1.84 Performance of TDES

API	Performance (Unit: cycle)		
	16-byte processing	48-byte processing	80-byte processing
R_TSIP_TdesEcbEncryptInit	820	820	820
R_TSIP_TdesEcbEncryptUpdate	440	640	840
R_TSIP_TdesEcbEncryptFinal	330	330	330
R_TSIP_TdesEcbDecryptInit	840	840	840
R_TSIP_TdesEcbDecryptUpdate	460	660	860
R_TSIP_TdesEcbDecryptFinal	340	340	340
R_TSIP_TdesCbcEncryptInit	880	880	880
R_TSIP_TdesCbcEncryptUpdate	490	690	890
R_TSIP_TdesCbcEncryptFinal	350	350	350
R_TSIP_TdesCbcDecryptInit	880	880	880
R_TSIP_TdesCbcDecryptUpdate	510	720	910
R_TSIP_TdesCbcDecryptFinal	370	370	370

Table 1.85 Performance of Common API (ARC4 Key Index Generation)

API	Performance (Unit: cycle)
R_TSIP_GenerateArc4KeyIndex	4,000
R_TSIP_GenerateArc4RandomKeyIndex	9,200
R_TSIP_UpdateArc4KeyIndex	3,800

Table 1.86 Performance of RSASSA-PKCS-v1_5 Signature Generation/Verification (HASH = SHA1)

API	Performance (Unit: cycle)		
	Message size=16byte	Message size=48byte	Message size=80byte
R_TSIP_Arc4EncryptInit	1,900	1,900	1,900
R_TSIP_Arc4EncryptUpdate	370	490	620
R_TSIP_Arc4EncryptFinal	240	240	240
R_TSIP_Arc4DecryptInit	1,900	1,900	1,900
R_TSIP_Arc4DecryptUpdate	370	490	620
R_TSIP_Arc4DecryptFinal	240	230	230

Table 1.87 Performance of Common API (RSA Key Index Generation)

API	Performance (Unit: cycle)
R_TSIP_GenerateRsa1024PublicKeyIndex	37,000
R_TSIP_GenerateRsa1024PrivateKeyIndex	38,000
R_TSIP_GenerateRsa2048PublicKeyIndex	140,000
R_TSIP_GenerateRsa2048PrivateKeyIndex	140,000
R_TSIP_GenerateRsa1024RandomKeyIndex *1	59,000,000
R_TSIP_GenerateRsa2048RandomKeyIndex *1	450,000,000
R_TSIP_UpdateRsa1024PublicKeyIndex	37,000
R_TSIP_UpdateRsa1024PrivateKeyIndex	38,000
R_TSIP_UpdateRsa2048PublicKeyIndex	140,000
R_TSIP_UpdateRsa2048PrivateKeyIndex	140,000

Note 1. Average value at 10 runs.

Table 1.88 Performance of RSASSA-PKCS-v1_5 Signature Generation/Verification (HASH = SHA1)

API	Performance (Unit: cycle)		
	Message size=1byte	Message size=128byte	Message size=256byte
R_TSIP_RsassaPkcs1024SignatureGenerate	1,300,000	1,300,000	1,300,000
R_TSIP_RsassaPkcs1024SignatureVerification	17,000	18,000	18,000
R_TSIP_RsassaPkcs2048SignatureGenerate	27,000,000	27,000,000	27,000,000
R_TSIP_RsassaPkcs2048SignatureVerification	140,000	140,000	140,000

Table 1.89 Performance of RSASSA-PKCS-v1_5 Signature Generation/Verification (HASH = SHA256)

API	Performance (Unit: cycle)		
	Message size=1byte	Message size=128byte	Message size=256byte
R_TSIP_RsassaPkcs1024SignatureGenerate	1,300,000	1,300,000	1,300,000
R_TSIP_RsassaPkcs1024SignatureVerification	17,000	18,000	18,000
R_TSIP_RsassaPkcs2048SignatureGenerate	27,000,000	27,000,000	27,000,000
R_TSIP_RsassaPkcs2048SignatureVerification	140,000	140,000	140,000

Table 1.90 Performance of RSASSA-PKCS-v1_5 Signature Generation/Verification (HASH = MD5)

API	Performance (Unit: cycle)		
	Message size=1byte	Message size=128byte	Message size=256byte
R_TSIP_RsassaPkcs1024SignatureGenerate	1,300,000	1,300,000	1,300,000
R_TSIP_RsassaPkcs1024SignatureVerification	17,000	18,000	18,000
R_TSIP_RsassaPkcs2048SignatureGenerate	27,000,000	27,000,000	27,000,000
R_TSIP_RsassaPkcs2048SignatureVerification	140,000	140,000	140,000

Table 1.91 Performance of RSAES-PKCS1-v1_5 Encryption/Decryption with 1,024-Bit Key Size

API	Performance (Unit: cycle)	
	Message size=1byte	Message size=117byte
R_TSIP_RsaesPkcs1024Encrypt	21,000	16,000
R_TSIP_RsaesPkcs1024Decrypt	1,300,000	1,300,000

Table 1.92 Performance of RSAES-PKCS1-v1_5 Encryption/Decryption with 2,048-Bit Key Size

API	Performance (Unit: cycle)	
	Message size=1byte	Message size=245byte
R_TSIP_RsaesPkcs2048Encrypt	150,000	140,000
R_TSIP_RsaesPkcs2048Decrypt	27,000,000	27,000,000

Table 1.93 Performance of HASH (SHA1)

API	Performance (Unit: cycle)		
	128-byte processing	192-byte processing	256-byte processing
R_TSIP_Sha1Init	100	100	100
R_TSIP_Sha1Update	1,300	1,500	1,700
R_TSIP_Sha1Final	670	670	670

Table 1.94 Performance of HASH (SHA256)

API	Performance (Unit: cycle)		
	128-byte processing	192-byte processing	256-byte processing
R_TSIP_Sha256Init	110	110	110
R_TSIP_Sha256Update	1,300	1,500	1,700
R_TSIP_Sha256Final	640	640	640

Table 1.95 Performance of HASH (MD5)

API	Performance (Unit: cycle)		
	128-byte processing	192-byte processing	256-byte processing
R_TSIP_Md5Init	94	94	94
R_TSIP_Md5Update	1,200	1,400	1,500
R_TSIP_Md5Final	630	630	630

Table 1.96 Performance of Common API (HMAC Key Index Generation)

API	Performance (Unit: cycle)
R_TSIP_GenerateSha1HmacKeyIndex	2,400
R_TSIP_GenerateSha256HmacKeyIndex	2,400
R_TSIP_UpdateSha1HmacKeyIndex	2,200
R_TSIP_UpdateSha256HmacKeyIndex	2,200

Table 1.97 Performance of HMAC (SHA1)

API	Performance (Unit: cycle)		
	128-byte processing	192-byte processing	256-byte processing
R_TSIP_Sha1HmacGenerateInit	1,100	1,100	1,100
R_TSIP_Sha1HmacGenerateUpdate	800	1,100	1,300
R_TSIP_Sha1HmacGenerateFinal	1,700	1,700	1,700
R_TSIP_Sha1HmacVerifyInit	1,100	1,100	1,100
R_TSIP_Sha1HmacVerifyUpdate	810	1,100	1,300
R_TSIP_Sha1HmacVerifyFinal	2,800	2,800	2,800

Table 1.98 Performance of HMAC (SHA256)

API	Performance (Unit: cycle)		
	128-byte processing	192-byte processing	256-byte processing
R_TSIP_Sha256HmacGenerateInit	1,400	1,400	1,400
R_TSIP_Sha256HmacGenerateUpdate	740	910	1,100
R_TSIP_Sha256HmacGenerateFinal	1,600	1,600	1,600
R_TSIP_Sha256HmacVerifyInit	1,400	1,400	1,400
R_TSIP_Sha256HmacVerifyUpdate	730	910	1,100
R_TSIP_Sha256HmacVerifyFinal	2,800	2,800	2,800

Table 1.99 Performance of Common API (ECC Key Index Generation)

API	Performance (Unit: cycle)
R_TSIP_GenerateEccP192PublicKeyIndex	2,700
R_TSIP_GenerateEccP224PublicKeyIndex	2,700
R_TSIP_GenerateEccP256PublicKeyIndex	2,700
R_TSIP_GenerateEccP384PublicKeyIndex	2,900
R_TSIP_GenerateEccP192PrivateKeyIndex	2,400
R_TSIP_GenerateEccP224PrivateKeyIndex	2,400
R_TSIP_GenerateEccP256PrivateKeyIndex	2,400
R_TSIP_GenerateEccP384PrivateKeyIndex	2,400
R_TSIP_GenerateEccP192RandomKeyIndex *1	140,000
R_TSIP_GenerateEccP224RandomKeyIndex *1	150,000
R_TSIP_GenerateEccP256RandomKeyIndex *1	150,000
R_TSIP_GenerateEccP384RandomKeyIndex *1	1,100,000
R_TSIP_UpdateEccP192PublicKeyIndex	2,500
R_TSIP_UpdateEccP224PublicKeyIndex	2,400
R_TSIP_UpdateEccP256PublicKeyIndex	2,500
R_TSIP_UpdateEccP384PublicKeyIndex	2,600
R_TSIP_UpdateEccP192PrivateKeyIndex	2,100
R_TSIP_UpdateEccP224PrivateKeyIndex	2,200
R_TSIP_UpdateEccP256PrivateKeyIndex	2,100
R_TSIP_UpdateEccP384PrivateKeyIndex	2,200

Note 1. Average value at 10 runs.

Table 1.100 Performance of ECDSA Signature Generation/Verification

API	Performance (Unit: cycle)		
	Message size=1byte	Message size=128byte	Message size=256byte
R_TSIP_EcdsaP192SignatureGenerate	170,000	170,000	170,000
R_TSIP_EcdsaP224SignatureGenerate	170,000	170,000	170,000
R_TSIP_EcdsaP256SignatureGenerate	170,000	170,000	170,000
R_TSIP_EcdsaP384SignatureGenerate*1	1,200,000		
R_TSIP_EcdsaP192SignatureVerification	310,000	310,000	310,000
R_TSIP_EcdsaP224SignatureVerification	330,000	330,000	340,000
R_TSIP_EcdsaP256SignatureVerification	330,000	330,000	340,000
R_TSIP_EcdsaP384SignatureVerification*1	2,100,000		

Note 1. Not include SHA384 calculation.

Table 1.101 Performance of Key Exchange

API	Performance (Unit: cycle)
R_TSIP_EcdhP256Init	42
R_TSIP_EcdhP256ReadPublicKey	340,000
R_TSIP_EcdhP256MakePublicKey	310,000
R_TSIP_EcdhP256CalculateSharedSecretIndex	360,000
R_TSIP_EcdhP256KeyDerivation	3,200
R_TSIP_EcdheP512KeyAgreement	3,300,000
R_TSIP_Rsa2048DhKeyAgreement	53,000,000

Key exchange performance (without KeyAgreement) was measured with parameters fixed as follows: key exchange format = ECDHE and derived key type = AES-128.

2. API Information

2.1 Hardware Requirements

The TSIP driver depends upon the TSIP capabilities provided on the MCU. Use a model name provided built-in TSIP.

2.2 Software Requirements

The TSIP driver is dependent on the following module:

r_bsp Use rev7.10 or later.

- When using the RX231 or RX23W (On the RX231, a portion of the comment below following “= Chip” differs.)

Change the following macro value to 0xB, or 0xD(Only RX23W) of the file r_bsp_config.h in the r_config folder.

```
/* Chip version.
   Character(s) = Value for macro =
   A   = 0xA     = Chip version A
                 = Security function not included.
   B   = 0xB     = Chip version B
                 = Security function included.
   C   = 0xC     = Chip version C
                 = Security function not included.
   D   = 0xD     = Chip version D
                 = Security function included.
*/
#define BSP_CFG_MCU_PART_VERSION        (0xB)
```

- When using the RX66T or RX72T (On the RX72T, a portion of the comment below following “= PGA” differs.)

Change the value of the following macro in r_bsp_config.h in the r_config folder to 0xE, 0xF, or 0x10.

```
/* Whether PGA differential input, Encryption and USB are included or not.
   Character(s) = Value for macro = Description
   A = 0xA  = PGA differential input included, Encryption module not included,
              USB module not included
   B = 0xB  = PGA differential input not included, Encryption module not included,
              USB module not included
   C = 0xC  = PGA differential input included, Encryption module not included,
              USB module included
   E = 0xE  = PGA differential input included, Encryption module included,
              USB module not included
   F = 0xF  = PGA differential input not included, Encryption module included,
              USB module not included
   G = 0x10 = PGA differential input included, Encryption module included,
              USB module included
*/
#define BSP_CFG_MCU_PART_FUNCTION   (0xE)
```

- If using RX66N, RX671, RX72M, or RX72N

Change the value of the following macro of r_bsp_config.h in the r_config folder to 0x11

```
/* Whether Encryption is included or not.
   Character(s) = Value for macro = Description
   D            = 0xD             = Encryption module not included
   H            = 0x11            = Encryption module included
*/
#define BSP_CFG_MCU_PART_FUNCTION       (0x11)
```

- If using RX65N

Change the value of the following macro of r_bsp_config.h in the r_config folder to true.

```
/* Whether Encryption and SDHI/SDSI are included or not.
   Character(s) = Value for macro = Description
   A  = false = Encryption module not included, SDHI/SDSI module not included
   B  = false = Encryption module not included, SDHI/SDSI module included
   D  = false = Encryption module not included, SDHI/SDSI module included
   E  = true  = Encryption module included, SDHI/SDSI module not included
   F  = true  = Encryption module included, SDHI/SDSI module included
   H  = true  = Encryption module included, SDHI/SDSI module included
*/
#define BSP_CFG_MCU_PART_ENCRYPTION_INCLUDED   (true)
```

2.3 Supported Toolchain

The operation of the TSIP driver with the following toolchain has been confirmed.

RX Family C/C++ Compiler Package V3.04.00

The TSIP driver has been confirmed to work with the tool chain shown in "5.1 Confirmed Operation Environment".

2.4 Header File

All API calls and their supported interface definitions are contained in r_tsip_rx_if.h.

2.5 Integer Types

This project uses stdint.h defined ANSI C99.

2.6 API Data Structure

For the data structures used in the TSIP driver, refer to r_tsip_rx_if.h.

2.7 Return Values

This shows the different values API functions can return. This enum is found in r_tsip_rx_if.h along with the API function declarations.

```
typedef enum e_tsip_err
{
   TSIP_SUCCESS=0,
    TSIP_ERR_FAIL,           // Self-check failed to terminate normally, or
                             // Detected illegal MAC by using
                             // R_TSIP_VerifyFirmwareMAC.
                             // or each R_TSIP_ function internal error.
   TSIP_ERR_RESOURCE_CONFLICT,// A resource conflict occurred because
                                // a resource required by the processing
                                // routine was in use by another
                                // processing routine.
   TSIP_ERR_RETRY,              // Indicates that self-check terminated
                                //with an error. Run the function again.
   TSIP_ERR_KEY_SET,    // An error occuerd when setting the invalid key index.
   TSIP_ERR_AUTHENTICATION     // Authentication failed
   TSIP_ERR_CALLBACK_UNREGIST,// Callback function is not registered.
   TSIP_ERR_PARAMETER,         // Input date is illegal.
   TSIP_ERR_PROHIBIT_FUNCTION,         // An invalid function call occurred.
   TSIP_RESUME_FIRMWARE_GENERATE_MAC,  // There is additional processing.
                                       //It is necessary to call the API again.
   TSIP_ERR_VERIFICATION_FAIL,      // Verification of TLS1.3 handshake failed.
  }e_tsip_err_t
```

2.8 Adding the FIT Module to Your Project

This module must be added to each project in which it is used. Renesas recommends using “Smart Configurator” described in (1) or (3). However, “Smart Configurator” only supports some RX devices. Please use the methods of (2) or (4) for unsupported RX devices.

(1) Adding the FIT module to your project using “Smart Configurator” in e^2 studio
By using the “Smart Configurator” in e^2 studio, the FIT module is automatically added to your project. Refer to “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)” for details.

(2) Adding the FIT module to your project using “FIT Configurator” in e^2 studio
By using the “FIT Configurator” in e^2 studio, the FIT module is automatically added to your project. Refer to “Adding Firmware Integration Technology Modules to Projects (R01AN1723)” for details.

(3) Adding the FIT module to your project using “Smart Configurator” on CS+
By using the “Smart Configurator Standalone version” in CS+, the FIT module is automatically added to your project. Refer to “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)” for details.

(4) Adding the FIT module to your project in CS+
In CS+, please manually add the FIT module to your project. Refer to “Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)” for details.

3. How to Use TSIP Driver

The TSIP driver for the RX family provides the following functions:

- Random number generation
- Secure key management
- Unauthorized access monitoring
- Acceleration of cryptographic operations
- Acceleration of TLS processing

The keys handled by the TSIP driver (input and output keys) are opaque keys wrapped in a device-specific key called Hardware Unique Key(HUK), which is accessible only by TSIP. In the RX TSIP driver, this opaque key is called the Key Index. Secure key management in the TSIP driver is achieved by wrapping the key with a Hardware Unique Key, which provides key confidentiality and tamper detection outside of TSIP.

Unauthorized access monitoring by TSIP covers all cryptographic operations provided by this driver and is always enabled during cryptographic operations. If tampering of cryptographic operations is detected while this driver is in use, this driver will stop operation.

There are two types of APIs provided by the TSIP driver for accelerating cryptographic operations: those that provide cryptographic operations with a single API and those that provide them with multiple APIs. In this document, the former is referred to as single-part operations and the latter as multi-part operations.

Symmetric ciphers and hashes provide APIs for multi-part operations split into Init-Update-Final, while other ciphers provide APIs for single-part operations.

3.1 How to Recover from Unauthorized Access Detection

Unauthorized access monitoring by TSIP is always enabled during execution of all cryptographic APIs. If tampering of cryptographic operations is detected while this driver is in use, this driver will stop operation in an infinite loop.

Whether or not the TSIP driver is stopped in an infinite loop due to unauthorized access must be detected on the user application side using a watchdog timer or other means.

If unauthorized access is detected by the user application, take appropriate measures to satisfy the system security policy, such as logging and system restart.

To recover from unauthorized access detection, close the TSIP driver once with R_TSIP_Close() and restart TSIP again with R_TSIP_Open() or reset the device.

3.2 How to Avoid TSIP Access Conflicts

All RX Family products with TSIP can use only one channel of TSIP, and the TSIP driver, like drivers for many peripheral IPs, occupies the hardware resources of TSIP during the execution of the driver API.

Among the APIs that provide multipart operations, the symmetric key cipher and HMAC functions continue to occupy TSIP hardware resources until a series of multipart operations are completed.

Therefore, when using the TSIP driver in a user application program, please note the following two points to avoid access conflicts to TSIP.

1) It is not allowed to execute other TSIP driver APIs while the TSIP driver API is being executed.

2) Symmetric key ciphers and HMAC functions cannot execute other TSIP driver APIs until a series of operations (Init/Update/Final) currently being processed are completed.

Note that the message digest generation function can execute other TSIP driver APIs during a series of multi-part operations.

If a TSIP driver API causes a TSIP hardware resource access conflict, the API returns TSIP_ERR_RESOURCE_CONFLICT or TSIP_ERR_PROHIBIT_FUNCTION.

When using the TSIP driver, please use one of the following method to avoid TSIP access conflicts.

- Use the APIs in an order that does not cause TSIP access conflicts.

3.3 BSP FIT Module Integration

The TSIP driver uses the BSP FIT module internally as described in section 2.2. When using the TSIP driver, please link the following API. For details, refer to the Board Support Package Module Firmware Integration Technology Application Note (R01AN1685xJxxxxxx).

- R_BSP_RegisterProtectEnable()
- R_BSP_RegisterProtectDisable()
- R_BSP_InterruptsEnable()
- R_BSP_InterruptsDisable()

It is also assumed that BSP startup has been completed before these APIs are called; if BSP startup is not used, call R_BSP_StartupOpen() beforehand. Initialize internal variables used within the above API.

3.4 Single-part and Multi-part Operations

There are two types of APIs provided by the TSIP driver for accelerating cryptographic operations: those that provide cryptographic operations in a single API and those that provide them in multiple APIs. In this document, the former is referred to as single-part operations and the latter as multi-part operations.

Symmetric key ciphers and hashes (message digest generation function and HMAC function) provide APIs for multi-part operations, while other ciphers provide APIs for single-part operations.

Multi-part operations are APIs which split a single cryptographic operation into a sequence of separate steps (Init-Update-Final). This enables fine control over the configuration of the cryptographic operation, and allows the message data to be processed in fragments instead of all at once.

All multi-part operations follow the same pattern of use:

Init: Initialize and start the operation.

On success, the operation active. On failure, the operation will enter an error state.

Update: Update the operation. The update function can provide additional parameters, supply data for processing or generate outputs.

On success, the operation remains active. On failure, the operation will enter an error state.

Final: To end the operation, call the applicable finalizing function. This will take any final inputs, produce any final outputs, and then release any resources associated with the operation.

On success, the operation returns to the inactive state. On failure, the operation will enter an error state.

3.5 Open and Close

This driver provides APIs for the following types of driver management operation:

No.	API	Description
1	R_TSIP_Open	Open TSIP driver Initializes the TSIP, self-test of the TSIP fault detection and random number generation circuitry.
2	R_TSIP_Close	Close TSIP driver
3	R_TSIP_SoftwareReset	Reset TSIP driver
4	R_TSIP_GetVersion	Get TSIP driver version

Applications using this driver must call R_TSIP_Open() to initialize the driver before using other functions. Also, when terminating the use of this driver, R_TSIP_Close() must be called.

If some problems occur while using the driver and you want to reset the driver and its control target, TSIP, you need to call R_TSIP_SoftwareReset() or R_TSIP_Open() after calling R_TSIP_Close(). driver, call R_TSIP_SoftwareReset() if you do not want to resume TSIP driver processing, or call R_TSIP_Open() if you want to resume TSIP driver processing.

In R_TSIP_Open(), a self-test is performed to detect hardware failure of TSIP and to confirm that there is no abnormality in the random number generation circuit. The self-test of the random number generator circuitry performs the health test described in NIST SP800-90B on the data generated by the physical random number generator, evaluates the entropy using and then generates a random number of seed.

3.6 Random Number Generation

This driver provides an API for the random number generation:

No.	API	Description
1	R_TSIP_GenerateRandomNumber	Random numbers are generated using the CTR-DRBG method described in NIST SP800-90A.

3.7 Key Management

This driver provides APIs for the following types of key management operation:

No.	API	Description
1	R_TSIP_GenerateUpdateKeyRingKeyIndex R_TSIP_GenerateAes*XXX*KeyIndex R_TSIP_GenerateTdesKeyIndex R_TSIP_GenerateArc4KeyIndex R_TSIP_GenerateSha*XXX*HmacKeyIndex R_TSIP_GenerateRsa*XXX*PublicKeyIndex R_TSIP_GenerateRsa*XXX*PrivateKeyIndex R_TSIP_GenerateEccP*XXX*KeyIndex R_TSIP_GenerateTlsRsaPublicKeyIndex	Key injection API that uses the Renesas Key Wrap service to convert a user key into a Key Index wrapped in a HUK. It can be used for factory key injection. [Aes] XXX = 128, 256 [Hmac] XXX = 1, 256 [Rsa] XXX = 1024, 2048, 3072 [Note], 4096 [Note] [Ecc] XXX = 196, 224, 256, 384
2	R_TSIP_UpdateAes*XXX*KeyIndex R_TSIP_UpdateTdesKeyIndex R_TSIP_UpdateArc4KeyIndex R_TSIP_UpdateRsa*XXX*PublicKeyIndex R_TSIP_UpdateRsa*XXX*PrivateKeyIndex R_TSIP_UpdateEccP*XXX*KeyIndex	Key update API that uses an Update Key Ring to convert a user key into a Key Index wrapped in a HUK. It can be used to update keys in the field. [Aes] XXX = 128, 256 [Hmac] XXX = 1, 256 [Rsa] XXX = 1024, 2048, 3072 [Note], 4096 [Note] [Ecc] XXX = 196, 224, 256, 384
3	R_TSIP_GenerateAes*XXX*RandomKeyIndex R_TSIP_GenerateTdesRandomKeyIndex R_TSIP_GenerateArc4RandomKeyIndex R_TSIP_GenerateRsa*XXX*RandomKeyIndex R_TSIP_GenerateEccP*XXX*RandomKeyIndex	Generates a random key and converts it to a Key Index. [Aes] XXX = 128, 256 [Rsa] XXX = 1024, 2048 [Ecc] XXX = 196, 224, 256, 384

Note These key lengths provide only public keys.

3.7.1 Key Injection and Update

Key Injection and Key Update provide a mechanism to enable secure delivery of user keys, converting them into a Key Index wrapped with a HUK.

Wrapping a private key with Hardware Unique Key involves encryption and adding MAC, while wrapping a public key involves only adding MAC. Since the Key Index of the public key is not encrypted, the plaintext public key can be extracted from the Key Index of the public key.

The user key is encrypted in AES-128 CBC mode using the first 128 bits of the Provisioning Key or Update Key Ring used for wrapping. Calculate the MAC of the user key in AES-128 CBC-MAC using the trailing 128 bits of the Provisioning Key or Update Key Ring used for wrapping as the key. Concatenate the encrypted user key with the MAC of the user key to generate an Encrypted User Key.

This application note explains the provisioning key and encrypted provisioning key using the key attached to the sample program. These key for mass production needs to be newly generated. An application note with these key details is available.

We will provide the product to customers who will be adopting or plan to adopt a Renesas microcontroller. Please contact your local Renesas Electronics sales office or distributor.

https://www.renesas.com/contact/

3.8 Symmetric Cryptography

This driver provides APIs for the following types of symmetric cryptographic operation:

No.	API	Description
1	R_TSIP_Aes*XXX*[*Mode*]Encrypt* R_TSIP_Aes*XXX*[*Mode*]Decrypt* R_TSIP_Aes*XXX*Ctr* R_TSIP_Tdes[*Mode*]Encrypt* R_TSIP_Tdes[*Mode*]Decrypt* R_TSIP_Arc4Encrypt* R_TSIP_Arc4Decrypt*	Symmetric ciphers AES 128/256bit: ECB, CBC, CTR encryption and decryption TDES: ECB, CBC encryption and decryption ARC4 *XXX*=128, 256 *Mode*=Ecb, Cbc
2	R_TSIP_Aes*XXX*GcmEncrypt* R_TSIP_Aes*XXX*GcmDecrypt* R_TSIP_Aes*XXX*CcmEncrypt* R_TSIP_Aes*XXX*CcmDecrypt*	Authenticated encryption with associated data (AEAD) AES-GCM, AES-CCM 128/256bit encryption and decryption *XXX*=128, 256
3	R_TSIP_Aes*XXX*CmacGenerate* R_TSIP_Aes*XXX*CmacVerify*	Message authentication codes (MAC) AES-CMAC 128/256bit MAC operation *XXX*=128, 256
4	R_TSIP_AES*XXX*KeyWrap R_TSIP_AES*XXX*KeyUnwrap	AES Key Wrap/ Unwrap *XXX*=128, 256

*= Init, Update, Final

For each type of symmetric cryptographic operation, the API provides a set of functions that allow multi-part operations. For details on multi-part operations, refer to 3.4 Single-part and Multi-part Operations.

3.8.1 Symmetric ciphers

The encryption operation for each AES mode is used as follows:

Call R_TSIP_AesXXX[Mode]EncryptInit() to specify the required key and initial vector.

Call the R_TSIP_AesXXX[Mode]EncryptUpdate() function on successive chunks of the plaintext message.

To complete an encryption operation, call R_TSIP_AesXXX[Mode]EncryptFinal().

The decryption operation for each AES mode is used as follows:

Call R_TSIP_AesXXX[Mode]DecryptInit() to specify the required key and initial vector.

Call the R_TSIP_AesXXX[Mode]DecryptUpdate() function on successive chunks of the ciphertext message.

To complete an decryption operation, call R_TSIP_AesXXX[Mode]DecryptFinal().

The TDES and ARC4 cryptographic APIs are used in the same way as AES.

3.8.2 Authenticated encryption with associated data (AEAD)

The AES-GCM encryption operation is used as follows:

Call R_TSIP_AesXXXGcmEncryptInit() to specify the required key and initial vector.

Call the R_TSIP_AesXXXGcmEncryptUpdate() function on successive chunks of the plaintext message and additional data.

To complete an encryption operation and to compute authentication tag, call R_TSIP_AesXXXGcmEncryptFinal().

The AES-GCM decryption operation is used as follows:

Call R_TSIP_AesXXXGCMDecryptInit() to specify the required key and initial vector.

Call the R_TSIP_AesXXXGCMDecryptUpdate() function on successive chunks of the ciphertext message and additional data.

To complete an decryption operation and to compute authentication tag and verify it against a reference value, call R_TSIP_AesXXXGcmDecryptFinal().

The AES-CCM cryptographic APIs are used in the same way as AES-GCM.

3.8.3 Message authentication codes (MAC)

The AES-CMAC signing operation is used as follows:

Call R_TSIP_AesXXXCmacGenerateInit() to specify the required key.

Call the R_TSIP_AesXXXCmacGenerateUpdate() function on successive chunks of the message.

To complete the MAC signing of a message, call R_TSIP_AesXXXCmacGenerateFinal().

The AES-CMAC verification operation is used as follows:

Call R_TSIP_AesXXXCmacVerifyInit() to specify the required key.

Call the R_TSIP_AesXXXCmacVerifyUpdate() function on successive chunks of the message.

To verify the MAC of a message, call R_TSIP_AesXXXCmacVerifyFinal() and specify the required MAC for verification.

3.9 Asymmetric Cryptography

This driver provides APIs for the following types of asymmetric cryptographic operation:

No.	API	Description
1	R_TSIP_RsaesPkcsXXXEncrypt R_TSIP_RsaesPkcsXXXDecrypt	[RSAES-PKCS1-V1_5 encrypt] *XXX* = 1024, 2048, 3072, 4096 [RSAES-PKCS1-V1_5 decrypt] *XXX* = 1024, 2048
2	R_TSIP_RsassaPkcsXXXSignatureGenerate R_TSIP_RsassaPkcsXXXSignatureVerification R_TSIP_RsassaPssXXXSignatureGenerate R_TSIP_RsassaPssXXXSignatureVerification R_TSIP_EcdsaPXXXSignatureGenerate R_TSIP_EcdsaPXXXSignatureVerification	[RSASSA-PKCS1-V1_5 Sign] *XXX* = 1024, 2048 [RSASSA-PKCS1-V1_5 verify] *XXX* = 1024, 2048, 3072, 4096 [RSASSA-PSS sign/verify] *XXX* = 1024, 2048 [ECDSA sign/verify] *XXX* = 192, 224, 256, 384

For asymmetric encryption and signature, the API provides single-part functions.

3.10 Hash Functions

This driver provides APIs for the following types of hash operation:

No.	API	Description
1	R_TSIP_ShaXXX* R_TSIP_Md5* R_TSIP_GetCurrentHashDigestValue	Message digests (hash functions) SHA-1, SHA-256 *XXX* = 1, 256
2	R_TSIP_ShaXXXHmacGenerate* R_TSIP_ShaXXXHmacVerify*	Message authentication codes (MAC) HMAC: HMAC-SHA1, HMAC-SHA256 *XXX* = 1, 256

*= Init, Update, Final

For each type of hash operation, the API provides a set of functions that allow multi-part operations. For details on multi-part operations, refer to 3.4 Single-part and Multi-part Operations.

3.10.1 Message digests (hash functions)

A hash operation is used as follows:

Call R_TSIP_ShaXXXInit() to specify the required hash algorithm and the newly allocated object for the operation.

Call the R_TSIP_ShaXXXUpdate() function on successive chunks of the message.

To calculate the digest of a message, call R_TSIP_ShaXXXFinal().

After R_TSIP_ShaXXXUpdate(), you can call R_TSIP_GetCurrentHashDigestValue() to retrieve data in progress of the hash operation.

The MD5 APIs are used in the same way as SHA.

3.10.2 Message authentication codes (HMAC)

The HMAC signing operation is used as follows:

Call R_TSIP_ShaXXXHmacGenerateInit() to specify the required key and the newly allocated object for the operation.

Call the R_TSIP_ShaXXXHmacGenerateUpdate() function on successive chunks of the message.

To complete the MAC signing of a message, call R_TSIP_ShaXXXHmacGenerateFinal().

The HMAC verification operation is used as follows:

Call R_TSIP_ShaXXXHmacVerifyInit() to specify the required key and the newly allocated object for the operation.

Call the R_TSIP_ShaXXXHmacVerifyUpdate() function on successive chunks of the message.

To verify the MAC of a message, call R_TSIP_ShaXXXHmacVerifyFinal() and specify the required MAC for verification.

3.11 Firmware Update

The TSIP driver supports a firmware update function to decrypt encrypted programs and perform MAC verification.

This driver provides APIs for the following firmware updates

No.	API	Discription
1	R_TSIP_StartUpdateFirmware	Transition TSIP to a state where the firmware update function is available.
2	R_TSIP_GenerateFirmwareMAC	Decrypts the encrypted program, verify with MAC, and generates a MAC.
3	R_TSIP_VerifyFirmwareMac	Verifies with the MAC generated by R_TSIP_GenerateFirmwareMAC for the specified area.

The sample program to execute firmware update and an application note with details of this function is available.

We will provide the product to customers who will be adopting or plan to adopt a Renesas microcontroller. Please contact your local Renesas Electronics sales office or distributor.

https://www.renesas.com/contact/

4. API Functions

4.1 List of API Functions

The TSIP driver implements the following API functions

(1) TSIP system function API
(2) Random number generate API
(3) AES Encryption/Decryption API
(4) DES Encryption/Decryption API
(5) ARC4 Encryption/Decryption API
(6) RSA Operation API
(7) ECC Signature Generate/Verify API
(8) HASH Caluculation API
(9) SHA-HMAC Generate/Verify API
(10) DH Caluculation API
(11) ECDH Key Exchange API
(12) Key Wrap API
(13) TLS function API
(14) Firmware Update API

Table 4.1 System function API

API	Description	TSIP-Lite	TSIP
R_TSIP_Open	Enables TSIP functionality.	✔	✔
R_TSIP_Close	Disables TSIP functionality.	✔	✔
R_TSIP_SoftwareReset	Resets the TSIP module.	✔	✔
R_TSIP_GetVersion	Outputs the TSIP driver version.	✔	✔
R_TSIP_GenerateUpdateKeyRingKeyIndex	Generates a Key Index for key updating.	✔	✔

Table 4.2 Random number generate API

API	Description	TSIP-Lite	TSIP
R_TSIP_GenrateRandomNumber	Generate random number.	✔	✔

Table 4.3 AES Encryption/Decryption API

API	Description	TSIP-Lite	TSIP
R_TSIP_GenerateAes*XXX*KeyIndex	Generates AES Key Index.	✔	✔
R_TSIP_UpdateAes*XXX*KeyIndex	Generates the updating AES Key Index.	✔	✔
R_TSIP_GenerateAes*XXX*RandomKeyIndex	The AES key is generated from random numbers and output with the Key Index.	✔	✔
R_TSIP_Aes*XXX*EcbEncryptInit R_TSIP_Aes*XXX*EcbEncryptUpdate R_TSIP_Aes*XXX*EcbEncryptFinal	AES-ECB mode encryption is performed using AES Key Index.	✔	✔
R_TSIP_Aes*XXX*EcbDecryptInit R_TSIP_Aes*XXX*EcbDecryptUpdate R_TSIP_Aes*XXX*EcbDecryptFinal	AES-ECB mode decryption is performed using AES Key Index.	✔	✔
R_TSIP_Aes*XXX*CbcEncryptInit R_TSIP_Aes*XXX*CbcEncryptUpdate R_TSIP_Aes*XXX*CbcEncryptFinal	AES-CBC mode encryption is performed using AES Key Index.	✔	✔
R_TSIP_Aes*XXX*CbcDecryptInit R_TSIP_Aes*XXX*CbcDecryptUpdate R_TSIP_Aes*XXX*CbcDecryptFinal	AES-CBC mode decryption is performed using AES Key Index.	✔	✔
R_TSIP_Aes*XXX*CtrInit R_TSIP_Aes*XXX*CtrUpdate R_TSIP_Aes*XXX*CtrFinal	AES-CTR mode cryptographic is performed using AES Key Index.	✔	✔
R_TSIP_Aes*XXX*GcmEncryptInit R_TSIP_Aes*XXX*GcmEncryptUpdate R_TSIP_Aes*XXX*GcmEncryptFinal	AES-GCM mode encryption is performed using AES Key Index.	✔	✔
R_TSIP_Aes*XXX*GcmDecryptInit R_TSIP_Aes*XXX*GcmDecryptUpdate R_TSIP_Aes*XXX*GcmDecryptFinal	AES-GCM mode decryption is performed using AES Key Index.	✔	✔
R_TSIP_Aes*XXX*CcmEncryptInit R_TSIP_Aes*XXX*CcmEncryptUpdate R_TSIP_Aes*XXX*CcmEncryptFinal	AES-CCM mode encryption is performed using AES Key Index.	✔	✔
R_TSIP_Aes*XXX*CcmDecryptInit R_TSIP_Aes*XXX*CcmDecryptUpdate R_TSIP_Aes*XXX*CcmDecryptFinal	AES-CCMmode decryption is performed using AES Key Index.	✔	✔
R_TSIP_Aes*XXX*CmacGenerateInit R_TSIP_Aes*XXX*CmacGenerateUpdate R_TSIP_Aes*XXX*CmacGenerateFinal	AES-CMAC mode MAC generation is performed using AES Key Index.	✔	✔
R_TSIP_Aes*XXX*CmacVerifyInit R_TSIP_Aes*XXX*CmacVerifyUpdate R_TSIP_Aes*XXX*CmacVerifyFinal	Verifies MACs generated in AES-CMAC mode using AES Key Index.	✔	✔

Table 4.4 DES Encryption/Decryption API

API	Description	TSIP-Lite	TSIP
R_TSIP_GenerateTdesKeyIndex	Generates TDES Key Index.	-	✔
R_TSIP_UpdateTdesKeyIndex	Generates the updating TDES Key Index.	-	✔
R_TSIP_GenerateTdesRandomKeyIndex	The TDES key is generated from random numbers and output with the Key Index.	-	✔
R_TSIP_TdesEcbEncryptInit R_TSIP_TdesEcbEncryptUpdate R_TSIP_TdesEcbEncryptFinal	TDES-ECB mode encryption.	-	✔
R_TSIP_TdesEcbDecryptInit R_TSIP_TdesEcbDecryptUpdate R_TSIP_TdesEcbDecryptFinal	TDES-ECB mode decryption.	-	✔
R_TSIP_TdesCbcEncryptInit R_TSIP_TdesCbcEncryptUpdate R_TSIP_TdesCbcEncryptFinal	TDES-CBC mode encryption.	-	✔
R_TSIP_TdesCbcDecryptInit R_TSIP_TdesCbcDecryptUpdate R_TSIP_TdesCbcDecryptFinal	TDES-CBC mode decryption.	-	✔

Table 4.5 ARC4 Encryption/Decryption API

API	Description	TSIP-Lite	TSIP
R_TSIP_GenerateArc4KeyIndex	Generates ARC4 Key Index.	-	✔
R_TSIP_UpdateArc4KeyIndex	Generates the updating ARC4 Key Index.	-	✔
R_TSIP_GenerateArc4RandomKeyIndex	The ARC4 key is generated from random numbers and output with the Key Index.	-	✔
R_TSIP_Arc4EncryptInit R_TSIP_Arc4EncryptUpdate R_TSIP_Arc4EncryptFinal	ARC4 encryption.	-	✔
R_TSIP_Arc4DecryptInit R_TSIP_Arc4DecryptUpdate R_TSIP_Arc4DecryptFinal	ARC4 decryption.	-	✔

Table 4.6 RSA Operation API

API	Description	TSIP-Lite	TSIP
R_TSIP_GenerateRsa*XXX*PrivateKeyIndex	Generates RSA Private key Key Index.	-	✔
R_TSIP_GenerateRsa*XXX*PublicKeyIndex	Generates RSA Public key Key Index.	-	✔
R_TSIP_UpdateRsa*XXX*PrivateKeyIndex	Generates the updating RSA Private key Key Index.	-	✔
R_TSIP_UpdateRsa*XXX*PublicKeyIndex	Generates the updating RSA Public key Key Index.	-	✔
R_TSIP_GenerateRsa*XXX*RandomKeyIndex	The RSA key is generated from random numbers and output with the Key Index. Exponent is fixed at 0x10001.	-	✔
R_TSIP_RsaesPkcs*XXX*Encrypt	RSA encryption by RSAES-PKCS1-V1_5.	-	✔
R_TSIP_RsaesPkcs*XXX*Decrypt	RSA decryption by RSAES-PKCS1-V1_5.	-	✔
R_TSIP_RsassaPkcs*XXX*SignatureGenerate	Generate signatures by RSASSA-PKCS1-V1_5.	-	✔
R_TSIP_RsassaPkcs*XXX*SignatureVerification	Verify signatures by RSASSA-PKCS1-V1_5.	-	✔
R_TSIP_RsassaPss*XXX*SignatureGenerate	Generate signatures by RSASSA-PSS.	-	✔
R_TSIP_RsassaPss*XXX*SignatureVerification	Verify signatures by RSASSA-PSS.	-	✔

Table 4.7 ECC signature generation/verification API

API	Description	TSIP-Lite	TSIP
R_TSIP_GenerateEccP*XXX*PublicKeyIndex	Generates ECC Private key Key Index.	-	✔
R_TSIP_GenerateEccP*XXX*PrivateKeyIndex	Generates ECC Public key Key Index.	-	✔
R_TSIP_UpdateEccP*XXX*PublicKeyIndex	Generates the updating ECC Private key Key Index.	-	✔
R_TSIP_UpdateEccP*XXX*PrivateKeyIndex	Generates the updating ECC Public key Key Index.	-	✔
R_TSIP_GenerateEccP*XXX*RandomKeyIndex	Generates Key Index for ECC private keys and the corresponding public key.	-	✔
R_TSIP_EcdsaP*XXX*SignatureGenerate	Generate signatures by ECDSA.	-	✔
R_TSIP_EcdsaP*XXX*SignatureVerification	Verify signatures by ECDSA.	-	✔

Table 4.8 HASH calucuration API

API	Description	TSIP-Lite	TSIP
R_TSIP_Sha*XXX*Init R_TSIP_Sha*XXX*Update R_TSIP_Sha*XXX*Final	Perform hash value operations with SHA.	-	✔
R_TSIP_Md5Init R_TSIP_Md5Update R_TSIP_Md5Final	Perform hash value operations with MD5.	-	✔
R_TSIP_GetCurrentHashDigestValue	Output hash value calculation in progress.	-	✔

Table 4.9 SHA-HMAC generate/verify API

API	Description	TSIP-Lite	TSIP
R_TSIP_GenerateSha*XXX*HmacKeyIndex	Generates a Key Index for SHA-HMAC computation.	-	✔
R_TSIP_UpdateSha*XXX*HmacKeyIndex	Updates a Key Index for SHA-HMAC computation.	-	✔
R_TSIP_Sha*XXX*HmacGenerateInit R_TSIP_Sha*XXX*HmacGenerateUpdate R_TSIP_Sha*XXX*HmacGenerateFinal	Perform SHA-HMAC calculation.	-	✔
R_TSIP_Sha*XXX*HmacVerifyInit R_TSIP_Sha*XXX*HmacVerifyUpdate R_TSIP_Sha*XXX*HmacVerifyFinal	Verifies SHA-HMAC calculation.	-	✔

Table 4.10 DH caluculation API

API	Description	TSIP-Lite	TSIP
R_TSIP_Rsa2048DhKeyAgreement	Calculate DH key agreement using RSA-2048	-	✔

Table 4.11 ECDH key exchange API

API	Description	TSIP-Lite	TSIP
R_TSIP_EcdhP256Init	Prepares to perform ECDH P-256 key exchange computation.	-	✔
R_TSIP_EcdhP256ReadPublicKey	Verifies the ECC P-256 public key signature of the other key exchange party.	-	✔
R_TSIP_EcdhP256MakePublicKey	Signs the ECC P-256 private key.	-	✔
R_TSIP_EcdhP256CalculateSharedSecretIndex	Computes the shared secret Z from the public key of the other key exchange party and your own public key.	-	✔
R_TSIP_EcdhP256KeyDerivation	Derives Z from the shared key.	-	✔
R_TSIP_EcdheP512KeyAgreement	Calculate ECDHE key agreement using Brainpool P512r1	-	✔

Table 4.12 Key Wrap API

API	Description	TSIP-Lite	TSIP
R_TSIP_Aes*XXX*KeyWrap	Wraps a key with an AES key.	✔	✔
R_TSIP_Aes*XXX*KeyUnwrap	Unwraps a key wrapped with an AES key.	✔	✔

Table 4.13 TLS function API

API	Description	TSIP -Lite	TSIP
R_TSIP_GenerateTlsRsaPublicKeyIndex	Generates an RSA 2048-bit public key Key Index used in TLS cooperation.	-	✔
R_TSIP_UpdateTlsRsaPublicKeyIndex	Update an RSA 2048-bit public key Key Index used in TLS cooperation.	-	✔
R_TSIP_TlsRootCertificateVerification	Verifies the root CA certificate bundle.	-	✔
R_TSIP_TlsCertificateVerification	Verifies a signature in the server certificate and intermediate certificate.	-	✔
R_TSIP_TlsCertificateVerificationExtension	Verifies a signature in the server certificate and intermediate certificate.	-	✔
R_TSIP_TlsGeneratePreMasterSecret	Generates the encrypted PreMasterSecret.	-	✔
R_TSIP_TlsEncryptPreMasterSecretWithRsa2048PublicKey	Encrypts the PreMasterSecret using RSA-2048.	-	✔
R_TSIP_TlsGenerateMasterSecret	Generates the encrypted MasterSecret.	-	✔
R_TSIP_TlsGenerateSessionKey	Outputs TLS communication keys.	-	✔
R_TSIP_TlsGenerateVerifyData	Generates VerifyData.	-	✔
R_TSIP_TlsServersEphemeralEcdhPublicKeyRetrieves	Verifies a ServerKeyExchange signature.		✔
R_TSIP_GenerateTlsP256EccKeyIndex	Generates a key pair from a random number used by the TLS cooperation function for elliptic curve cryptography over a 256-bit prime field.		✔
R_TSIP_TlsGeneratePreMasterSecretWithEccP256Key	Generates an ECC encrypted PreMasterSecret.		✔
R_TSIP_TlsGenerateExtendedMasterSecret	Generates the encrypted ExtendedMasterSecret.		✔
R_TSIP_GenerateTls13P256EccKeyIndex	Generates a key pair from a random number used by the TLS1.3 cooperation function for elliptic curve cryptography over a 256-bit prime field.		✔
R_TSIP_Tls13GenerateEcdheSharedSecret	Generates a SharedSecret Key Index.		✔
R_TSIP_Tls13GenerateHandshakeSecret	Generates a HandshakeSecret Key Index.		✔
R_TSIP_Tls13GenerateServerHandshakeTrafficKey	Generates a ServerWriteKey Key Index and a ServerFinishedKey Key Index.		✔
R_TSIP_Tls13GenerateServerHandshakeVerification	Verifys Finished provided by the server.		✔
R_TSIP_Tls13GenerateClientHandshakeTrafficKey	Generates a ClientWriteKey Key Index and a ClientFinishedKey Key Index.		✔
R_TSIP_Tls13GenerateMasterSecret	Generates a MasterSecret Key Index.		✔

R_TSIP_Tls13GenerateApplicationTrafficKey	Generates ApplicationTrafficSecret Key Indexes and AppicatonTrafficKey Key Indexes.		✔
R_TSIP_Tls13UpdateApplicationTrafficKey	Updates an ApplicationTrafficSecret Key Index and an AppicatonTrafficKey Key Index.		✔
R_TSIP_Tls13GenerateResumptionMasterSecret	Generates a ResumptionMasterSecret Key Index.		✔
R_TSIP_Tls13GeneratePreSharedKey	Generates a PreSharedKey Key Index.		✔
R_TSIP_Tls13GeneratePskBinderKey	Generates a Binder Key Index.		✔
R_TSIP_Tls13GenerateResumptionHandshakeSecret	Generates a HandshakeSecret Key Index for Resumption.		✔
R_TSIP_Tls13Generate0RttApplicationWriteKey	Generates a ClientWriteKey Key Index for 0-RTT.		✔
R_TSIP_Tls13CertificateVerifyGenerate	Generates CertificateVerify used by the TLS1.3 cooperation function.		✔
R_TSIP_Tls13CertificateVerifyVerification	Verifies CertificateVerify used by the TLS1.3 cooperation function.		✔
R_TSIP_GenerateTls13SVP256EccKeyIndex	Generates a key pair from a random number used by the TLS1.3 cooperation function for elliptic curve cryptography over a 256-bit prime field.		✔
R_TSIP_Tls13SVGenerateEcdheSharedSecret	Generates a SharedSecret Key Index.		✔
R_TSIP_Tls13SVGenerateHandshakeSecret	Generates a HandshakeSecret Key Index.		✔
R_TSIP_Tls13SVGenerateServerHandshakeTrafficKey	Generates a ServerWriteKey Key Index and a ServerFinishedKey Key Index.		✔
R_TSIP_Tls13SVGenerateClientHandshakeTrafficKey	Generates a ClientWriteKey key index and a ClientFinishedKey key index.		✔
R_TSIP_Tls13SVGenerateClientHandshakeVerification	Verifys Finished provided by the client.		✔
R_TSIP_Tls13SVGenerateMasterSecret	Generates a MasterSecret Key Index.		✔
R_TSIP_Tls13SVGenerateApplicationTrafficKey	Generates ApplicationTrafficSecret Key Indexes and AppicatonTrafficKey Key Indexes.		✔
R_TSIP_Tls13SVUpdateApplicationTrafficKey	Updates an ApplicationTrafficSecret Key Index and an AppicatonTrafficKey Key Index.		✔
R_TSIP_Tls13SVGenerateResumptionMasterSecret	Generates a ResumptionMasterSecret Key Index.		✔
R_TSIP_Tls13SVGeneratePreSharedKey	Generates a PreSharedKey Key Index.		✔
R_TSIP_Tls13SVGeneratePskBinderKey	Generates a Binder Key Index.		✔
R_TSIP_Tls13SVGenerateResumptionHandshakeSecret	Generates a HandshakeSecret Key Index for Resumption.		✔
R_TSIP_Tls13SVGenerate0RttApplicationWriteKey	Generates a ClientWriteKey Key Index for 0-RTT.		✔
R_TSIP_Tls13SVCertificateVerifyGenerate	Generates CertificateVerify used by the TLS1.3 cooperation function.		✔

R_TSIP_Tls13SVCertificateVerifyVerification	Verifies CertificateVerify used by the TLS1.3 cooperation function.		✔
R_TSIP_Tls13EncryptInit	Prepares to encrypt data used by the TLS1.3 cooperation function.		✔
R_TSIP_Tls13EncryptUpdate	Encrypts data used by the TLS1.3 cooperation function.		✔
R_TSIP_Tls13EncryptFinal	Prepares final processing for encryption used by the TLS1.3 cooperation function.		✔
R_TSIP_Tls13DecryptInit	Prepares to decrypt data used by the TLS1.3 cooperation function.		✔
R_TSIP_Tls13DecryptUpdate	Decrypts data used by the TLS1.3 cooperation function.		✔
R_TSIP_Tls13DecryptFinal	Prepares final processing for decryption used by the TLS1.3 cooperation function.		✔

Table 4.14 Firmware Update API

API	Description	TSIP-Lite	TSIP
R_TSIP_StartUpdateFirmware	Transitions to firmware update mode.	✔	✔
R_TSIP_GenerateFirmwareMAC	Decrypts and generates the MAC for the encrypted firmware.	✔	✔
R_TSIP_VerifyFirmwareMAC	Performs a MAC check on the firmware.	✔	✔

4.2 Detailed Description of API Functions

4.2.1 System API

4.2.1.1 R_TSIP_Open

Format

```
#include “r_tsip_rx_if.h”
e_tsip_err_t R_TSIP_Open (
        tsip_tls_ca_certification_public_key_index_t *key_index_1,
        tsip_update_key_ring_t *key_index_2
)
```

Parameters

key_index_1	Input	TLS cooperation RSA public keyring Key Index
key_index_2	Input	Key update keyring Key Index

Return Values

TSIP_SUCCESS	Normal termination
TSIP_ERR_FAIL	The error-detection self-test failed to terminate normally.
TSIP_ERR_RESOURCE_CONFLICT	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_RETRY	Indicates that an entropy evaluation failure occurred. Run the function again.

Description

Enables use of TSIP functionality.

For key_index_1, input the “Key Index of TLS cooperation RSA public key” generated by R_TSIP_GenerateTlsRsaPublicKeyIndex() or R_TSIP_UpdateTlsRsaPublicKeyIndex(). If the TLS cooperation function is not used, input a null pointer.

For key_index_2, input the “keyring Key Index for key update” generated by R_TSIP_GenerateUpdateKeyRingKeyIndex(). If the key update cooperation function is not used, input a null pointer.

Reentrant

Not supported

4.2.1.2 R_TSIP_Close

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Close(void);
```

Parameters

None

Return Values

TSIP_SUCCESS	Normal termination

Description

Stops supply of power to the TSIP.

Reentrant

Not supported

4.2.1.3 R_TSIP_SoftwareReset

Format

#include "r_tsip_rx_if.h"

void R_TSIP_SoftwareReset (void);

Parameters

None.

Return Values

None.

Description

Reverts the state to the TSIP initial state.

Reentrant

Not supported

4.2.1.4 R_TSIP_GetVersion

Format

```
#include "r_tsip_rx_if.h"
uint32_t R_TSIP_GetVersion(void);
```

Parameters

None

Return Values

Upper 2 bytes :	Major version (decimal notation)
Lower 2 bytes :	Minor version (decimal notation)

Description

This function can get the TSIP driver version.

Reentrant

Not supported

4.2.1.5 R_TSIP_GenerateUpdateKeyRingKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateUpdateKeyRingKeyIndex
        (uint8_t *encrypted_provisioning_key, uint8_t *iv, uint8_t *encrypted_key,
        tsip_update_key_ring_t  *key_index);
```

Parameters

encrypted_provisioning_key	Input	Provisioning key wrapped by the DLM server
iv	Input	Initialization vector when generating encrypted_key
encrypted_key	Input	User key encrypted and MAC appended
key_index	Output	Update Key Ring Key Index

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This API outputs a Key Index for the Update Key Ring.

For encrypted_key, enter data encrypted with Provisioning Key as shown in 5.3.7Update Key Ring.

See 3.7.1Key Injection and Updatefor how to generate encrypted_provisioning_key, iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.2 Random number generation

4.2.2.1 R_TSIP_GenerateRandomNumber

Format

#include “r_tsip_rx_if.h”

e_tsip_err_t R_TSIP_GenerateRandomNumber(uint32_t *random);

Parameters

random	Output	Stores 4words (16 bytes) random data.

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.

Description

This API can generate NIST SP800-90A compliant word of 4 random number.

Reentrant

Not supported

4.2.3 AES

4.2.3.1 R_TSIP_GenerateAes*XXX*KeyIndex

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateAes128KeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_update_key_ring_t *key_index
        )
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateAes256KeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_update_key_ring_t *key_index
        )
```

Parameters

encrypted_provisioning_key	Input	Provisioning key wrapped by the DLM server
iv	Input	Initialization vector when generating encrypted_key
encrypted_key	Input	User key encryptedand MAC appended
key_index	Output	Key Index

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.

Description

R_TSIP_GenerateAes128KeyIndex outputs 128-bit AES Key Index.

R_TSIP_GenerateAes256KeyIndex outputs 256-bit AES Key Index.

For encrypted_key, enter data encrypted with Provisioning Key as shown in 5.3.1AES.

Refer 3.7.1 Key Injection and Update for how to generate encrypted_provisioning_key, iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.3.2 R_TSIP_UpdateAes*XXX*KeyIndex

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateAes128KeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_aes_key_index_t *key_index
        )
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateAes256KeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_aes_key_index_t *key_index
        )
```

Parameters

iv	Input	Initialization vector when generating encrypted_key
encrypted_key	Input	User key encrypted with Update Key Ring with MAC appended
key_index	Output	Key Index

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

R_TSIP_UpdateAes128KeyIndex updates the Key Index of an AES 128 key.

R_TSIP_UpdateAes256KeyIndex updates the Key Index of an AES 256 key.

For encrypted_key, enter the data shown in 5.3.1 AES, encrypted with the Update Key Ring.

Refer 3.7.1 Key Injection and Update for how to generate iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.3.3 R_TSIP_GenerateAes*XXX*RandomKeyIndex

Format

(1) #include “r_tsip_rx_if.h”
e_tsip_err_t R_TSIP_GenerateAes128RandomKeyIndex(
    tsip_aes_key_index_t *key_index
)

(2) #include “r_tsip_rx_if.h”
e_tsip_err_t R_TSIP_GenerateAes256RandomKeyIndex(
    tsip_aes_key_index_t *key_index
)

Parameters

key_index	Output	(1) 128-bit AES Key Index (2) 256-bit AES Key Index

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.

Description

R_TSIP_GenerateAes128RandomKeyIndex outputs 128-bit AES Key Index.

R_TSIP_GenerateAes256RandomKeyIndex outputs 256-bit AES Key Index.

This API generates a user key from a random number in the TSIP. Accordingly, user key input is unnecessary. By encrypting data using the Key Index that is output by this API, dead copying of data can be prevented.

Refer 3.7.1 Key Injection and Update for how to use key_index.

Reentrant

Not supported

4.2.3.4 R_TSIP_Aes*XXX*EcbEncryptInit

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128EcbEncryptInit(
        tsip_aes_handle_t *handle,
        tsip_aes_key_index_t *key_index
        )
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256EcbEncryptInit(
        tsip_aes_handle_t *handle,
        tsip_aes_key_index_t *key_index
        )
```

Parameters

handle	Output	AES handler (work area)
key_index	Input	Key Index area

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET:	Input illegal Key Index.

Description

The R_TSIP_Aes*XXX*EcbEncryptInit() function performs preparations for the execution of an AES calculation, and writes the result to the first argument, handle. The value of handle is used as an argument in the subsequent R_TSIP_Aes*XXX*EcbEncryptUpdate() function and R_TSIP_Aes*XXX*EcbEncryptFinal() function.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.3.5 R_TSIP_Aes*XXX*EcbEncryptUpdate

Format

(1) #include "r_tsip_rx_if.h"

```
e_tsip_err_t R_TSIP_Aes128EcbEncryptUpdate(
        tsip_aes_handle_t *handle,
        uint8_t *plain,
        uint8_t *cipher,
        uint32_t plain_length
        )
```

(2) #include "r_tsip_rx_if.h"

```
e_tsip_err_t R_TSIP_Aes128EcbEncryptUpdate(
        tsip_aes_handle_t *handle,
        uint8_t *plain,
        uint8_t *cipher,
        uint32_t plain_length
        )
```

Parameters

handle	Input/Output	AES handler (work area)
plain	Input	plaintext data area
cipher	Output	ciphertext data area
plain_length	Input	byte length of plaintext data (must be a multiple of 16)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	Input data is illegal.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Aes*XXX*EcbEncryptUpdate() function encrypts the second argument, plain, utilizing the Key Index specified by the Init function, and writes the encryption result to the third argument, cipher. After plaintext input is completed, call R_TSIP_Aes*XXX*EcbEncryptFinal().

Specify areas for plain and cipher not not overlap excluding that both addresses are same.

Reentrant

Not supported

4.2.3.6 R_TSIP_Aes*XXX*EcbEncryptFinal

Format

(1) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_Aes128EcbEncryptFinal(
        tsip_aes_handle_t *handle,
        uint8_t *cipher,
        uint32_t *cipher_length
        )
```

(2) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_Aes256EcbEncryptFinal(
        tsip_aes_handle_t *handle,
        uint8_t *cipher,
        uint32_t *cipher_length
        )
```

Parameters

handle	Input	AES handler (work area)
cipher	Output	ciphertext data area (nothing ever written here)
cipher_length	Output	ciphertext data length (0 always written here)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	Input data is illegal.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

Using the handle specified in the first argument, handle, the R_TSIP_Aes*XXX*EcbEncryptFinal() function writes the calculation result to the second argument, cipher, and writes the length of the calculation result to the third argument, cipher_length. The original intent was for a portion of the encryption result that was not a multiple of 16 bytes to be written to the second argument. However, as a result of the restriction that only multiples of 16 can be input to the Update function, nothing is ever written to cipher, and 0 is always written to cipher_length. The arguments cipher and cipher_length are provided for compatibility in anticipation of the time when this restriction is lifted.

Reentrant

Not supported

4.2.3.7 R_TSIP_Aes*XXX*EcbDecryptInit

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128EcbDecryptInit(
        tsip_aes_handle_t *handle,
        tsip_aes_key_index_t *key_index
)
```
(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256EcbDecryptInit(
        tsip_aes_handle_t *handle,
        tsip_aes_key_index_t *key_index
)
```

Parameters

handle	Output	AES handler (work area)
key_index	Input	Key Index area

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET:	Input illegal Key Index.

Description

The R_TSIP_Aes*XXX*EcbDecryptInit() function performs preparations for the execution of an AES calculation, and writes the result to the first argument, handle. The value of handle is used as an argument in the subsequent R_TSIP_Aes*XXX*EcbDecryptUpdate() function and R_TSIP_Aes*XXX*EcbDecryptFinal() function.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.3.8 R_TSIP_Aes*XXX*EcbDecryptUpdate

Format

(1) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_Aes128EcbDecryptUpdate(
        tsip_aes_handle_t *handle,
        uint8_t *cipher,
        uint8_t *plain,
        uint32_t cipher_length
        )
```

(2) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_Aes256EcbDecryptUpdate(
        tsip_aes_handle_t *handle,
        uint8_t *cipher,
        uint8_t *plain,
        uint32_t cipher_length
        )
```

Parameters

handle	Input/Output	AES handler (work area)
cipher	Input	ciphertext data area
plain	Output	plaintext data area
cipher_length	Input	byte length of ciphertext data (must be a multiple of 16)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	Input data is illegal.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Aes*XXX*EcbDecryptUpdate() function decrypts the second argument, cipher, utilizing the Key Index specified by the Init function, and writes the decryption result to the third argument, plain. After ciphertext input is completed, call R_TSIP_Aes*XXX*EcbDecryptFinal().

Specify areas for plain and cipher not to overlap excluding that both addresses are same.

Reentrant

Not supported

4.2.3.9 R_TSIP_Aes*XXX*EcbDecryptFinal

Format

(1) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_Aes128EcbDecryptFinal(
        tsip_aes_handle_t *handle,
        uint8_t *plain,
        uint32_t *plain_length
        )
```

(2) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_Aes256EcbDecryptFinal(
        tsip_aes_handle_t *handle,
        uint8_t *plain,
        uint32_t *plain_length
        )
```

Parameters

handle	Input	AES handler (work area)
plain	Output	plaintext data area (nothing ever written here)
plain_length	Output	plaintext data length (0 always written here)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	Input data is illegal.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

Using the handle specified in the first argument, handle, the R_TSIP_Aes*XXX*EcbDecryptFinal() function writes the calculation result to the second argument, plain, and writes the length of the calculation result to the third argument, plain_length. The original intent was for a portion of the decryption result that was not a multiple of 16 bytes to be written to the second argument. However, as a result of the restriction that only multiples of 16 can be input to the Update function, nothing is ever written to plain, and 0 is always written to plain_length. The arguments plain and plain_length are provided for compatibility in anticipation of the time when this restriction is lifted.

Reentrant

Not supported

4.2.3.10 R_TSIP_Aes*XXX*CbcEncryptInit

Format

(1) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_Aes128CbcEncryptInit(
        tsip_aes_handle_t *handle,
        tsip_aes_key_index_t *key_index,
        uint8_t *ivec
)
```
(2) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_Aes256CbcEncryptInit(
        tsip_aes_handle_t *handle,
        tsip_aes_key_index_t *key_index,
        uint8_t *ivec
)
```

Parameters

handle	Output	AES handler (work area)
key_index	Input	Key Index area
ivec	Input	initial vector area(16byte)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET:	Input illegal Key Index.

Description

The R_TSIP_Aes*XXX*CbcEncryptInit() function performs preparations for the execution of an AES calculation, and writes the result to the first argument, handle. The value of handle is used as an argument in the subsequent R_TSIP_Aes*XXX*CbcEncryptUpdate() function and R_TSIP_Aes*XXX*CbcEncryptFinal() function.

When using the TLS cooperation function, input client_crypto_key_index or server_crypto_key_index, generated by R_TSIP_TlsGenerateSessionKeyR_TSIP_TlsGenerateSessionKey(), as key_index.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.3.11 R_TSIP_Aes*XXX*CbcEncryptUpdate

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CbcEncryptUpdate(
        tsip_aes_handle_t *handle,
        uint8_t *plain,
        uint8_t *cipher,
        uint32_t plain_length
        )
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CbcEncryptUpdate(
        tsip_aes_handle_t *handle,
        uint8_t *plain,
        uint8_t *cipher,
        uint32_t plain_length
        )
```

Parameters

handle	Input/Output	AES handler (work area)
plain	Input	plaintext data area
cipher	Output	ciphertext data area
plain_length	Input	byte length of plaintext data (must be a multiple of 16)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	Input data is illegal.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Aes*XXX*CbcEncryptUpdate() function encrypts the second argument, plain, utilizing the Key Index specified by Init function, and writes the encryption result to the third argument, cipher. After plaintext input is completed, call R_TSIP_Aes*XXX*CbcEncryptFinal().

Specify areas for plain and cipher not to overlap excluding that both addresses are same.

Reentrant

Not supported

4.2.3.12 R_TSIP_Aes*XXX*CbcEncryptFinal

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CbcEncryptFinal(
        tsip_aes_handle_t *handle,
        uint8_t *cipher,
        uint32_t *cipher_length
        )
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CbcEncryptFinal(
        tsip_aes_handle_t *handle,
        uint8_t *cipher,
        uint32_t *cipher_length
        )
```

Parameters

handle	Input	AES handler (work area)
cipher	Output	ciphertext data area (nothing ever written here)
cipher_length	Output	ciphertext data length (0 always written here)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	Input data is illegal.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

Using the handle specified in the first argument, handle, the R_TSIP_Aes*XXX*CbcEncryptFinal() function writes the calculation result to the second argument, cipher, and writes the length of the calculation result to the third argument, cipher_length. The original intent was for a portion of the encryption result that was not a multiple of 16 bytes to be written to the second argument. However, as a result of the restriction that only multiples of 16 can be input to the Update function, nothing is ever written to cipher, and 0 is always written to cipher_length. The arguments cipher and cipher_length are provided for compatibility in anticipation of the time when this restriction is lifted.

Reentrant

Not supported

4.2.3.13 R_TSIP_Aes*XXX*CbcDecryptInit

Format

(1) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_Aes128CbcDecryptInit(
        tsip_aes_handle_t *handle,
        tsip_aes_key_index_t *key_index,
        uint8_t *ivec
)
```
(2) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_Aes256CbcDecryptInit(
        tsip_aes_handle_t *handle,
        tsip_aes_key_index_t *key_index,
        uint8_t *ivec
)
```

Parameters

handle	Output	AES handler (work area)
key_index	Input	Key Index area
ivec	Input	initial vector area(16byte)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET:	Input illegal Key Index.

Description

The R_TSIP_Aes*XXX*CbcDecryptInit() function performs preparations for the execution of an AES calculation, and writes the result to the first argument, handle. The value of handle is used as an argument in the subsequent R_TSIP_Aes*XXX*CbcDecryptUpdate() function and R_TSIP_Aes*XXX*CbcDecryptFinal() function.

When using the TLS cooperation function, input client_crypto_key_index or server_crypto_key_index, generated by R_TSIP_TlsGenerateSessionKeyR_TSIP_TlsGenerateSessionKey(), as key_index.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.3.14 R_TSIP_Aes*XXX*CbcDecryptUpdate

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CbcDecryptUpdate(
        tsip_aes_handle_t *handle,
        uint8_t *cipher,
        uint8_t *plain,
        uint32_t cipher_length
        )
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CbcDecryptUpdate(
        tsip_aes_handle_t *handle,
        uint8_t *cipher,
        uint8_t *plain,
        uint32_t cipher_length
        )
```

Parameters

handle	Input/Output	AES handler (work area)
cipher	Input	ciphertext data area
plain	Output	plaintext data area
cipher_length	Input	byte length of ciphertext data (must be a multiple of 16)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	Input data is illegal.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Aes*XXX*CbcDecryptUpdate() function decrypts the second argument, cipher, utilizing the Key Index specified by the Init function, and writes the decryption result to the third argument, plain. After ciphertext input is completed, call R_TSIP_Aes*XXX*CbcDecryptFinal().

Specify areas for plain and cipher not to overlap excluding that both addresses are same.

Reentrant

Not supported

4.2.3.15 R_TSIP_Aes*XXX*CbcDecryptFinal

Format

(1) #include "r_tsip_rx_if.h"

```
e_tsip_err_t R_TSIP_Aes128CbcDecryptFinal(
        tsip_aes_handle_t *handle,
        uint8_t *plain,
        uint32_t *plain_length
        )
```

(2) #include "r_tsip_rx_if.h"

```
e_tsip_err_t R_TSIP_Aes256CbcDecryptFinal(
        tsip_aes_handle_t *handle,
        uint8_t *plain,
        uint32_t *plain_length
        )
```

Parameters

handle	Input	AES handler (work area)
plain	Output	plaintext data area (nothing ever written here)
plain_length	Output	plaintext data length (0 always written here)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	Input data is illegal.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

Using the handle specified in the first argument, handle, the R_TSIP_Aes*XXX*CbcDecryptFinal() function writes the calculation result to the second argument, plain, and writes the length of the calculation result to the third argument, plain_length. The original intent was for a portion of the decryption result that was not a multiple of 16 bytes to be written to the second argument. However, as a result of the restriction that only multiples of 16 can be input to the Update function, nothing is ever written to plain, and 0 is always written to plain_length. The arguments plain and plain_length are provided for compatibility in anticipation of the time when this restriction is lifted.

Reentrant

Not supported

4.2.3.16 R_TSIP_Aes*XXX*CtrInit

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CtrInit(
        tsip_aes_handle_t *handle,
        tsip_aes_key_index_t *key_index,
        uint8_t *ictr
)
```
(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CtrInit(
        tsip_aes_handle_t *handle,
        tsip_aes_key_index_t *key_index,
        uint8_t *ictr
)
```

Parameters

handle	Output	AES handler (work area)
key_index	Input	Key Index area
ictr	Input	initial counter (16byte)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET:	Invalid Key Index was input.

Description

The R_TSIP_Aes*XXX*CtrInit() function performs preparations for the execution of an AES calculation, and writes the result to the first argument, handle. The value of handle is used as an argument in the subsequent R_TSIP_Aes*XXX*CtrUpdate() function and R_TSIP_Aes*XXX*CtrFinal() function.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.3.17 R_TSIP_Aes*XXX*CtrUpdate

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CtrUpdate(
        tsip_aes_handle_t *handle,
        uint8_t *itext,
        uint8_t *otext,
        uint32_t itext_length
        )
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CtrUpdate(
        tsip_aes_handle_t *handle,
        uint8_t *itext,
        uint8_t *otext,
        uint32_t itext_length
        )
```

Parameters

handle	Input/Output	AES handler (work area)
itext	Input	input data (plain or cipher) area
otext	Output	output data (cipher or plain) area
itext_length	Input	byte length of input data (must be a multiple of 16)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	After the data from plain was input, an invalid handle was input from aad.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Aes*XXX*CtrUpdate() function encrypts the second argument, itext, utilizing the Key Index specified by the Init finction, and writes the result to the third argument, otext. After plaintext input is completed, call R_TSIP_Aes*XXX*CtrFinal().

When the length of the last block is 1~127 bit, allocate area which unit is 16 byte for itext and otext. And set arbitrary value to the fraction area of itext, and ignore the stored value in the fraction area of otext.

Specify areas for itext and otext not to overlap excluding that both addresses are same.

Reentrant

Not supported

4.2.3.18 R_TSIP_Aes*XXX*CtrFinal

Format

(1) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_Aes128CtrFinal(
        tsip_aes_handle_t *handle
        )
```

(2) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_Aes256CtrFinal(
        tsip_aes_handle_t *handle
        )
```

Parameters

handle	Input	AES handler (work area)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

Using the handle specified in the first argument, handle, the R_TSIP_Aes*XXX*CtrFinal() function finish the calculation.

Reentrant

Not supported

4.2.3.19 R_TSIP_Aes*XXX*GcmEncryptInit

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128GcmEncryptInit(
        tsip_gcm_handle_t *handle,
        tsip_aes_key_index_t *key_index,
        uint8_t *ivec,
        uint32_t ivec_len
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256GcmEncryptInit(
        tsip_gcm_handle_t *handle,
        tsip_aes_key_index_t *key_index,
        uint8_t *ivec,
        uint32_t ivec_len
)
```

Parameters

handle	Output	AES-GCM handler (work area)
key_index	Input	Key Index area
ivec	Input	initialization vector area (iv_len byte) [note]
ivec_len	Input	initialization vector length (1 or more bytes)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET:	Invalid Key Index was input.
TSIP_ERR_PARAMETER:	Input data is illegal.

Description

The R_TSIP_AesXXXGcmEncryptInit() function performs preparations for the execution of an GCM calculation, and writes the result to the first argument, handle. The value of handle is used as an argument in the subsequent R_TSIP_AesXXXGcmEncryptUpdate() function and R_TSIP_AesXXXGcmEncryptFinal() function. Moreover, please set 4-byte aligned RAM address to ivec.

[note]

When key_index->type is TSIP_KEY_INDEX_TYPE_AES128_FOR_TLS

The key_index value generated by the R_TSIP_TlsGenerateSessionKey() function when 6 or 7 is specified for select_cipher includes a 96-bit IV. Input a null pointer as the third argument, ivec. Specify 0 as the fourth argument, ivec_len.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.3.20 R_TSIP_Aes*XXX*GcmEncryptUpdate

Format

(1) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_Aes128GcmEncryptUpdate(
        tsip_gcm_handle_t *handle,
        uint8_t *plain,
        uint8_t *cipher,
        uint32_t plain_data_len,
        uint8_t *aad,
        uint32_t aad_len
)
```
(2) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_Aes256GcmEncryptUpdate(
        tsip_gcm_handle_t *handle,
        uint8_t *plain,
        uint8_t *cipher,
        uint32_t plain_data_len,
        uint8_t *aad,
        uint32_t aad_len
)
```

Parameters

handle	Input/Output	AES-GCM handler (work area)
plain	Input	plaintext data area
cipher	Output	ciphertext data area
plain_data_len	Input	plaintext data length (0 or more bytes)
aad	Input	additional authentication data (aad_len byte)
aad_len	Input	additional authentication data length (0 or more bytes)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	After the data from plain was input, an invalid handle was input from aad.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Aes*XXX*GcmEncryptUpdate() function encrypts the plaintext specified in the second argument, plain, in GCM mode using the values specified for key_index and ivec in R_TSIP_Aes*XXX*GcmEncryptInit(), along with the additional authentication data specified in the fifth argument, aad. Inside this function, the data that is input by the user is buffered until the input values of aad and plain exceed 16 bytes. After the input data from plain reaches 16 bytes or more, the encryption result is output to the ciphertext data area specified in the third argument, cipher. The lengths of the plain and aad data to input are respectively specified in the fourth argument, plain_data_len, and the sixth argument, aad_len. For these, specify not the total byte count for the aad and plain input data, but rather the data length to input when the user calls this function. If the input values plain and aad are not divisible by 16 bytes, they will be padded inside the function. First process the data that is input from aad, and then process the data that is input from plain. If aad data is input after starting to input plain data, an error will occur. If aad data and plain data are input to this function at the same time, the aad data will be processed, and then the function will transition to the plain data input state. Specify areas for plain and cipher not to overlap. For plain, cipher, and aad, specify RAM addresses that are multiples of 4.

Reentrant

Not supported

4.2.3.21 R_TSIP_Aes*XXX*GcmEncryptFinal

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128GcmEncryptFinal(
        tsip_gcm_handle_t *handle,
        uint8_t *cipher,
        uint32_t *cipher_data_len,
        uint8_t *atag
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256GcmEncryptFinal(
        tsip_gcm_handle_t *handle,
        uint8_t *cipher,
        uint32_t *cipher_data_len,
        uint8_t *atag
)
```

Parameters

handle	Input	AES-GCM handler (work area)
cipher	Output	ciphertext data area (data_len byte)
cipher_data_len	Output	ciphertext data length (0 or more bytes)
atag	Output	authentication tag area

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

If there is 16-byte fractional data indicated by the total data length of the value of plain that was input by R_TSIP_Aes*XXX*GcmEncryptUpdate (), the R_TSIP_Aes*XXX*GcmEncryptFinal() function will output the result of encrypting that fractional data to the ciphertext data area specified in the second argument, cipher. Here, the portion that does not reach 16 bytes will be padded with zeros. The authentication tag is output to the fourth argument, atag. For cipher and atag, specify RAM addresses that are multiples of 4.

Reentrant

Not supported

4.2.3.22 R_TSIP_Aes*XXX*GcmDecryptInit

Format

(1) #include “r_tsip_rx_if.h”

```
e_tsip_err_t R_TSIP_Aes128GcmDecryptInit(
        tsip_gcm_handle_t *handle,
        tsip_aes_key_index_t *key_index,
        uint8_t *ivec,
        uint32_t ivec_len
)
```

(2) #include “r_tsip_rx_if.h”

```
e_tsip_err_t R_TSIP_Aes256GcmDecryptInit(
        tsip_gcm_handle_t *handle,
        tsip_aes_key_index_t *key_index,
        uint8_t *ivec,
        uint32_t ivec_len
)
```

Parameters

handle	Output	AES-GCM handler (work area)
key_index	Input	Key Index area
ivec	Input	initialization vector area (iv_len byte) [note]
ivec_len	Input	initialization vector length (1 or more bytes)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET:	Invalid Key Index was input.
TSIP_ERR_PARAMETER:	Input data is illegal.

Description

The R_TSIP_Aes*XXX*GcmDecryptInit() function performs preparations for the execution of an GCM calculation, and writes the result to the first argument, handle. The value of handle is used as an argument in the subsequent R_TSIP_Aes*XXX*GcmDecryptUpdate() function and R_TSIP_Aes*XXX*GcmDecryptFinal() function. Moreover, please set 4-byte aligned RAM address to ivec.

[note]

When key_index->type is TSIP_KEY_INDEX_TYPE_AES128_FOR_TLS.

The key_index value generated by the R_TSIP_TlsGenerateSessionKey() function when 6 or 7 is specified for select_cipher includes a 96-bit IV. Input a null pointer as the third argument, ivec. Specify 0 as the fourth argument, ivec_len.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.3.23 R_TSIP_Aes*XXX*GcmDecryptUpdate

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128GcmDecryptUpdate(
        tsip_gcm_handle_t *handle,
        uint8_t *cipher,
        uint8_t *plain,
        uint32_t cipher_data_len,
        uint8_t *aad,
        uint32_t aad_len
        )
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256GcmDecryptUpdate(
        tsip_gcm_handle_t *handle,
        uint8_t *cipher,
        uint8_t *plain,
        uint32_t cipher_data_len,
        uint8_t *aad,
        uint32_t aad_len
        )
```

Parameters

handle	Input/Output	AES-GCM handler (work area)
cipher	Input	ciphertext data area
plain	Output	plaintext data area
cipher_data_len	Input	ciphertext data length (0 or more bytes)
aad	Input	additional authentication data (aad_len byte)
aad_len	Input	additional authentication data length (0 or more bytes)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	After the data from plain was input, an invalid handle was input from aad.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Aes*XXX*GcmDecryptUpdate() function decrypts the ciphertext specified in the second argument, cipher, in GCM mode using the values specified for key_index and ivec in R_TSIP_Aes*XXX*GcmDecryptInit(), along with the additional authentication data specified in the fifth argument, aad. Inside this function, the data that is input by the user is buffered until the input values of aad and plain exceed 16 bytes. After the input data from cipher reaches 16 bytes or more, the decryption result is output to the plaintext data area specified in the third argument, plain. The lengths of the cipher and aad data to input are respectively specified in the fourth argument, cipher_data_len, and the sixth argument, aad_len. For these, specify not the total byte count for the aad and cipher input data, but rather the data length to input when the user calls this function. If the input values cipher and aad are not divisible by 16 bytes, they will be padded inside the function. First process the data that is input from aad, and then process the data that is input from cipher. If aad data is input after starting to input cipher data, an error will occur. If aad data and cipher data are input to this function at the same time, the aad data will be processed, and then the function will transition to the cipher data input state. Specify areas for plain and cipher not to overlap. For plain, cipher, and aad, specify RAM addresses that are multiples of 4.

Reentrant

Not supported

4.2.3.24 R_TSIP_Aes*XXX*GcmDecryptFinal

Format

(1) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_Aes128GcmDecryptFinal(
        tsip_gcm_handle_t *handle,
        uint8_t *plain,
        uint32_t *plain_data_len,
        uint8_t *atag,
        uint32_t atag_len
        )
```

(2) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_Aes256GcmDecryptFinal(
        tsip_gcm_handle_t *handle,
        uint8_t *plain,
        uint32_t *plain_data_len,
        uint8_t *atag,
        uint32_t atag_len
        )
```

Parameters

handle	Input	AES-GCM handler (work area)
plain	Output	plaintext data area (data_len byte)
plain_data_len	Output	plaintext data length (0 or more bytes)
atag	Output	authentication tag area (atag_len byte)
atag_len	Input	authentication tag length (4,8,12,13,14,15,16byte)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_AUTHENTICATION:	Authentication failed
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.
TSIP_ERR_PARAMETER:	Input data is illegal

Description

The R_TSIP_Aes*XXX*GcmDecryptFinal() function decrypts, in GCM mode, the fractional ciphertext specified by R_TSIP_Aes*XXX*GcmDecryptUpdate() that does not reach 16 bytes, and ends GCM decryption. The encryption data and authentication tag are respectively output to the plaintext data area specified in the second argument, plain, and the authentication tag area specified in the fourth argument, atag. The decoded data length is output to the third argument, plain_data_len. If authentication fails, the return value will be TSIP_ERR_AUTHENTICATION. For the fourth argument, atag, input 16 bytes or less. If it is less than 16 bytes, it will be padded with zeros inside the function. For plain and atag, specify RAM addresses that are multiples of 4.

Reentrant

Not supported

4.2.3.25 R_TSIP_Aes*XXX*CcmEncryptInit

Format

(1) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_Aes128CcmEncryptInit(
        tsip_ccm_handle_t *handle,
        tsip_aes_key_index_t *key_index,
        uint8_t *nonce,
        uint32_t nonce_len,
        uint8_t *adata,
        uint8_t a_len,
        uint32_t payload_len,
        uint32_t mac_len
)
```
(2) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_Aes256CcmEncryptInit(
        tsip_ccm_handle_t *handle,
        tsip_aes_key_index_t *key_index,
        uint8_t *nonce,
        uint32_t nonce_len,
        uint8_t *adata,
        uint8_t a_len,
        uint32_t payload_len,
        uint32_t mac_len
)
```

Parameters

handle	Output	AES-CCM handler (work area)
key_index	Input	Key Index area
nonce	Input	Nonce
nonce_len	Input	Nonce data length (7 to 13 bytes)
adata	Input	additional authentication data
a_len	Input	additional authentication data length (0 to 110 bytes)
payload_len	Input	Payload length (any number of bytes)
mac_len	Input	MAC length (4, 6, 8, 10, 12, 14, or 16 bytes)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET:	Invalid Key Index was input.

Description

The R_TSIP_Aes*XXX*CcmEncryptInit() function prepares to perform CCM computation and writes the result to the first argument, handle. The succeeding functions R_TSIP_Aes*XXX*CcmEncryptUpdate() and R_TSIP_Aes*XXX*CcmEncryptFinal() use handle as an argument.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.3.26 R_TSIP_Aes*XXX*CcmEncryptUpdate

Format

(1) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_Aes128CcmEncryptUpdate(
        tsip_ccm_handle_t *handle,
        uint8_t *plain,
        uint8_t *cipher,
        uint32_t plain_length
)
```
(2) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_Aes256CcmEncryptUpdate(
        tsip_ccm_handle_t *handle,
        uint8_t *plain,
        uint8_t *cipher,
        uint32_t plain_length
)
```

Parameters

handle	Input/Output	AES-CCM handler (work area)
plain	Input	plaintext data area
cipher	Output	ciphertext data area
plain_length	Input	plaintext data length

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	An invalid function was called.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid handle was input.

Description

The R_TSIP_Aes*XXX*CcmEncryptUpdate() function encrypts the plaintext specified in the second argument, plain, in CCM mode using the values specified by key_index, nonce, and adata in R_TSIP_Aes*XXX*CcmEncryptInit(). This function buffers internally the data input by the user until the input value of plain exceeds 16 bytes. Once the amount of plain input data is 16 bytes or greater, the encrypted result is output to cipher, which is specified in the third argument. Use payload_len in R_TSIP_Aes*XXX*CcmEncryptInit() to specify the total data length of plain that will be input. Use plain_length in this function to specify the data length to be input when the user calls this function. If the input value of plain is less than 16 bytes, the function performs padding internally.

Ensure that the areas allocated to plain and cipher do not overlap. Also, specify RAM addresses that are multiples of 4 for plain and cipher.

Reentrant

Not supported

4.2.3.27 R_TSIP_Aes*XXX*CcmEncryptFinal

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CcmEncryptFinal(
        tsip_ccm_handle_t *handle,
        uint8_t *cipher,
        uint32_t *cipher_length,
        uint8_t *mac,
        uint32_t mac_length
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CcmEncryptFinal(
        tsip_ccm_handle_t *handle,
        uint8_t *cipher,
        uint32_t *cipher_length,
        uint8_t *mac,
        uint32_t mac_length
)
```

Parameters

handle	Input	AES-CCM handler (work area)
cipher	Output	ciphertext data area
cipher_data_len	Output	ciphertext data length
mac	Output	MAC area
mac_length	Input	MAC length (4, 6, 8, 10, 12, 14, or 16 bytes)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	Input data is illegal
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

If the data length of plain input in R_TSIP_Aes*XXX*CcmEncryptUpdate() results in leftover data after 16 bytes, the R_TSIP_Aes*XXX*CcmEncryptFinal() function outputs the leftover encrypted data to cipher, which is specified in the second argument. The MAC value is output to the fourth argument, mac. Set the fifth argument, mac_length, to the same value as that specified for the argument mac_len in Aes*XXX*CcmEncryptInit(). Also, specify RAM addresses that are multiples of 4 for cipher and mac.

Reentrant

Not supported

4.2.3.28 R_TSIP_Aes*XXX*CcmDecryptInit

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CcmDecryptInit(
        tsip_ccm_handle_t *handle,
        tsip_aes_key_index_t *key_index,
        uint8_t *nonce,
        uint32_t nonce_len,
        uint8_t *adata,
        uint8_t a_len,
        uint32_t payload_len,
        uint32_t mac_len
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CcmDecryptInit(
        tsip_ccm_handle_t *handle,
        tsip_aes_key_index_t *key_index,
        uint8_t *nonce,
        uint32_t nonce_len,
        uint8_t *adata,
        uint8_t a_len,
        uint32_t payload_len,
        uint32_t mac_len
)
```

Parameters

handle	Input	AES-CCM handler (work area)
key_index	Input	Key Index area
nonce	Input	Nonce
nonce_len	Input	Nonce data length (7 to 13 bytes)
adata	Input	additional authentication data
a_len	Input	additional authentication data length (0 to 110 bytes)
payload_len	Input	Payload length (any number of bytes)
mac_len	Input	MAC length (4, 6, 8, 10, 12, 14, or 16 bytes)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET:	Invalid Key Index was input.

Description

The R_TSIP_Aes*XXX*CcmDecryptInit() function prepares to perform CCM computation and writes the result to the first argument, handle. The succeeding functions R_TSIP_Aes*XXX*CcmDecryptUpdate() and R_TSIP_Aes*XXX*CcmDecryptFinal() use handle as an argument.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.3.29 R_TSIP_Aes*XXX*CcmDecryptUpdate

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CcmDecryptUpdate(
        tsip_ccm_handle_t *handle,
        uint8_t *cipher,
        uint8_t *plain,
        uint32_t cipher_length
)
```
(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CcmDecryptUpdate(
        tsip_ccm_handle_t *handle,
        uint8_t *cipher,
        uint8_t *plain,
        uint32_t cipher_length
)
```

Parameters

handle	Input/Output	AES-CCM handler (work area)
cipher	Input	plaintext data area
plain	Output	ciphertext data area
cipher_length	Input	ciphertext data length

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Aes*XXX*CcmDecryptUpdate() function decrypts the ciphertext specified by the second argument, cipher, in CCM mode using the values specified by key_index, nonce, and adata in in R_TSIP_Aes*XXX*CcmDecryptInit(). This function buffers internally the data input by the user until the input value of cipher exceeds 16 bytes. Once the amount of cipher input data is 16 bytes or greater, the decrypted result is output to plain, which is specified in the third argument. Use payload_len in R_TSIP_Aes*XXX*CcmDecryptInit() to specify the total data length of cipher that will be input. Use cipher_length in this function to specify the data length to be input when the user calls this function. If the input value of cipher is less than 16 bytes, the function performs padding internally.

Ensure that the areas allocated to cipher and plain do not overlap. Also, specify RAM addresses that are multiples of 4 for cipher and plain.

Reentrant

Not supported

4.2.3.30 R_TSIP_Aes*XXX*CcmDecryptFinal

Format

(1) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_Aes128CcmDecryptFinal(
        tsip_ccm_handle_t *handle,
        uint8_t *plain,
        uint32_t *plain_length,
        uint8_t *mac,
        uint32_t mac_length
        )
```

(2) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_Aes256CcmDecryptFinal(
        tsip_ccm_handle_t *handle,
        uint8_t *plain,
        uint32_t *plain_length,
        uint8_t *mac,
        uint32_t mac_length
        )
```

Parameters

handle	Input	AES-CCM handler (work area)
plain	Output	plaintext data area
plain_length	Output	plaintext data length
mac	Output	MAC area
mac_length	Input	MAC length (4, 6, 8, 10, 12, 14, or 16 bytes)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	Internal error, or authentication failed.
TSIP_ERR_PARAMETER:	Input data is illegal
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

If the data length of cipher input in R_TSIP_Aes*XXX*CcmDecryptUpdate() results in leftover data after 16 bytes, the R_TSIP_Aes*XXX*CcmDecryptFinal() function outputs the leftover decrypted data to cipher, which is specified in the second argument. In addition, the function verifies the fourth argument, mac. Set the fifth argument, mac_length, to the same value as that specified for the argument mac_len in Aes*XXX*CcmDecryptInit().

Reentrant

Not supported

4.2.3.31 R_TSIP_Aes*XXX*CmacGenerateInit

Format

(1) #include "r_tsip_rx_if.h"

```
e_tsip_err_t R_TSIP_Aes128CmacGenerateInit(
        tsip_cmac_handle_t *handle,
        tsip_aes_key_index_t *key_index
)
```

(2) #include "r_tsip_rx_if.h"

```
e_tsip_err_t R_TSIP_Aes256CmacGenerateInit(
        tsip_cmac_handle_t *handle,
        tsip_aes_key_index_t *key_index
)
```

Parameters

handle	Output	AES-CMAC handler (work area)
key_index	Input	Key Index area

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET:	Invalid Key Index was input.

Description

The R_TSIP_Aes*XXX*CmacGenerateInit() function performs preparations for the execution of an CMAC calculation, and writes the result to the first argument, handle. The value of handle is used as an argument in the subsequent R_TSIP_Aes*XXX*CmacGenerateUpdate() function and R_TSIP_Aes*XXX*CmacGenerateFinal() function.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.3.32 R_TSIP_Aes*XXX*CmacGenerateUpdate

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CmacGenerateUpdate(
        tsip_cmac_handle_t *handle,
        uint8_t *message,
        uint32_t message_length
)
```
(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CmacGenerateUpdate(
        tsip_cmac_handle_t *handle,
        uint8_t *message,
        uint32_t message_length
)
```

Parameters

handle	Input/Output	AES-CMAC handler (work area)
message	Input	message data area (message_length byte)
message_length	Input	message data length (0 or more bytes)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Aes*XXX*CmacGenerateUpdate() function performs MAC value generation based on the message specified in the second argument, message, using the value specified for key_index in R_TSIP_Aes*XXX*CmacGenerateInit(). Inside this function, the data that is input by the user is buffered until the input value of message exceeds 16 bytes. The length of the message data to input is specified in the third argument, message_len. For these, input not the total byte count for message input data, but rather the message data length to input when the user calls this function. If the input value, message, is not a multiple of 16 bytes, it will be padded within the function. For message, specify a RAM address that are multiples of 4.

Reentrant

Not supported

4.2.3.33 R_TSIP_Aes*XXX*CmacGenerateFinal

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CmacGenerateFinal(
        tsip_cmac_handle_t *handle,
        uint8_t *mac
)
```
(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CmacGenerateFinal(
        tsip_cmac_handle_t *handle,
        uint8_t *mac
)
```

Parameters

handle	Input	AES-CMAC handler (work area)
mac	Output	MAC data area (16byte)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Aes*XXX*CmacGenerateFinal() function outputs the MAC value to the MAC data area specified in the second argument, mac, and ends CMAC mode.

Reentrant

Not supported

4.2.3.34 R_TSIP_Aes*XXX*CmacVerifyInit

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CmacVerifyInit(
        tsip_cmac_handle_t *handle,
        tsip_aes_key_index_t *key_index
)
```
(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CmacVerifyInit(
        tsip_cmac_handle_t *handle,
        tsip_aes_key_index_t *key_index
)
```

Parameters

handle	Output	AES-CMAC handler (work area)
key_index	Input	Key Index area

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET:	Invalid Key Index was input.

Description

The R_TSIP_Aes*XXX*CmacVerifyInit() function performs preparations for the execution of a CMAC calculation, and writes the result to the first argument, handle. The value of handle is used as an argument in the subsequent R_TSIP_Aes*XXX*CmacVerifyUpdate() function and R_TSIP_Aes*XXX*CmacVerifyFinal() function.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.3.35 R_TSIP_Aes*XXX*CmacVerifyUpdate

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CmacVerifyUpdate(
        tsip_cmac_handle_t *handle,
        uint8_t *message,
        uint32_t message_length
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CmacVerifyUpdate(
        tsip_cmac_handle_t *handle,
        uint8_t *message,
        uint32_t message_length
)
```

Parameters

handle	Input/Output	AES-CMAC handler (work area)
message	Input	message data area (message_length byte)
message_length	Input	message data length (0 or more bytes)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Aes*XXX*CmacVerifyUpdate() function performs MAC value generation based on the message specified in the second argument, message, using the value specified for key_index in R_TSIP_Aes*XXX*CmacGenerateInit(). Inside this function, the data that is input by the user is buffered until the input value of message exceeds 16 bytes. The length of the message data to input is specified in the third argument, message_len. For these, input not the total byte count for message input data, but rather the message data length to input when the user calls this function. If the input value, message, is not a multiple of 16 bytes, it will be padded within the function. For message, specify a RAM address that are multiples of 4.

Reentrant

Not supported

4.2.3.36 R_TSIP_Aes*XXX*CmacVerifyFinal

Format

(1) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_Aes128CmacVerifyFinal(
        tsip_cmac_handle_t *handle,
        uint8_t *mac,
        uint32_t mac_length
)
```
(2) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_Aes256macVerifyFinal(
        tsip_cmac_handle_t *handle,
        uint8_t *mac,
        uint32_t mac_length
)
```

Parameters

handle	Input	AES-CMAC handler (work area)
mac	Input	MAC data area (mac_length byte)
mac_length	Input	MAC data length (2 to 16 bytes)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_AUTHENTICATION:	Authentication failed
TSIP_ERR_PARAMETER:	Input data is illegal .
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Aes*XXX*CmacVerifyFinal() function inputs the MAC value in the MAC data area specified in the second argument, mac, and verifies the MAC value. If authentication fails, the return value will be TSIP_ERR_AUTHENTICATION. If the MAC value is less than 16 bytes, it will be padded with zeros inside the function.

Reentrant

Not supported

4.2.4 DES

4.2.4.1 R_TSIP_GenerateTdesKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateTdesKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_tdes_key_index_t *key_index
)
```

Parameters

encrypted_provisioning_key	Input	Provisioning key wrapped by the DLM server
iv	Input	Initial vector when generating encrypted_key
encrypted_key	Input	Encrypted Triple-DES user key with MAC appended
key_index	Output	Triple-DES Key Index

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required for processing is in use by another processing routine.
TSIP_ERR_FAIL:	An internal error occurred.

Description

This API outputs Triple-DES Key Index.

For encrypted_key, enter data encrypted with Provisioning Key as shown in 5.3.2 DES.

Refer 3.7.1 Key Injection and Update for how to generate encrypted_provisioning_key, iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.4.2 R_TSIP_UpdateTdesKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateTdesKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_tdes_key_index_t *key_index
)
```

Parameters

iv	Input	Initialization vector when generating encrypted_key
encrypted_key	Input	User key encrypted with Update Key Ring with MAC
key_index	Output	Triple-DES Key Index

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_FAIL:	An internal error occurred.

Description

This API updates the Triple-DES Key Index.

For encrypted_key, enter the data shown in 5.3.2 DES, encrypted with the Update Key Ring.

Refer 3.7.1 Key Injection and Update for how to generate iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.4.3 R_TSIP_GenerateTdesRandomKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateTdesRandomKeyIndex(tsip_tdes_key_index_t *key_index);
```

Parameters

key_index	Output	Triple-DES Key Index (13 words)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required for processing is in use by another processing routine.

Description

This API outputs Triple-DES Key Index.

This API is used to generate a user key from a random number internally in the TSIP. Consequentially, there is no need to input a user key. The Key Index output by this API can be used to encrypt data and thereby prevent dead copying.

Refer 3.7.1 Key Injection and Update for how to use key_index.

Reentrant

Not supported

4.2.4.4 R_TSIP_TdesEcbEncryptInit

Format

```
#include “r_tsip_rx_if.h”
e_tsip_err_t R_TSIP_TdesEcbEncryptInit(
        tsip_tdes_handle_t *handle,
        tsip_tdes_key_index_t *key_index
)
```

Parameters

handle	Output	TDES handler (work area)
key_index	Input	Key Index area

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required for processing is in use by another processing routine.
TSIP_ERR_KEY_SET:	Incorrect Key Index was input.

Description

The R_TSIP_TdesEcbEncryptInit() function prepares to perform DES calculation and writes the result to the first argument, handle. The subsequent functions R_TSIP_TdesEcbEncryptUpdate() function and R_TSIP_TdesEcbEncryptFinal() also use handle as an argument.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.4.5 R_TSIP_TdesEcbEncryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesEcbEncryptUpdate(
        tsip_tdes_handle_t *handle,
        uint8_t *plain,
        uint8_t *cipher,
        uint32_t plain_length
)
```

Parameters

handle	Input / Output	TDES handler (work area)
plain	Input	plaintext data area
cipher	Output	ciphertext data area
plain_length	Input	byte length of plaintext data (Must be a multiple of 8.)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	An illegal handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An illegal function was called.

Description

The R_TSIP_TdesEcbEncryptUpdate() function uses the handle specified by the first argument, handle, and encrypts the contents of the second argument, plain, using the key_index specified in the Init function, and writes the encrypted result to the third argument, cipher. After plaintext input finishes, call R_TSIP_TdesEcbEncryptFinal().

Ensure that plain and cipher are not assigned to overlapping areas. Also, specify RAM addresses for plain and cipher that are multiples of 4.

Reentrant

Not supported

4.2.4.6 R_TSIP_TdesEcbEncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesEcbEncryptFinal(
        tsip_tdes_handle_t *handle,
        uint8_t *cipher,
        uint32_t *cipher_length
)
```

Parameters

handle	Input	TDES handler (work area)
cipher	Output	ciphertext data area (Nothing is ever written to this area.)
cipher_length	Outpu	ciphertext data length (Zero is always written to this area.)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	An illegal handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An illegal function was called.

Description

The R_TSIP_TdesEcbEncryptFinal() function writes the calculation result to the second argument, cipher, and the length of the calculation result to the third argument, cipher_length, using the handle specified by the first argument, handle. The leftover amount less than a multiple of 8 bytes was originally supposed to be encrypted and the result written to the second argument, but the Update function has a restriction that only allows it to handle values that are multiples of 8 bytes. Therefore, this function never actually writes anything to cipher and it always writes 0 to cipher_length. The arguments cipher and cipher_length are provided to ensure compatibility in case this restriction is removed in future.

Reentrant

Not supported

4.2.4.7 R_TSIP_TdesEcbDecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesEcbDecryptInit(
        tsip_tdes_handle_t *handle,
        tsip_tdes_key_index_t *key_index
)
```

Parameters

handle	Output	TDES handler (work area)
key_index	Input	Key Index area

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_KEY_SET:	Incorrect Key Index was input.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required for processing is in use by another processing routine.

Description

The R_TSIP_TdesEcbDecryptInit() function prepares to perform DES calculation and writes the result to the first argument, handle. The subsequent functions R_TSIP_TdesEcbDecryptUpdate() function and R_TSIP_TdesEcbDecryptFinal() also use handle as an argument.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.4.8 R_TSIP_TdesEcbDecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesEcbDecryptUpdate(
        tsip_tdes_handle_t *handle,
        uint8_t *cipher,
        uint8_t *plain,
        uint32_t cipher_length
)
```

Parameters

handle	Input/Output	TDES handler (work area)
cipher	Input	ciphertext data area
plain	Output	plaintext data area
cipher_length	Input	byte length of ciphertext data (Must be a multiple of 8.)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	An illegal handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An illegal function was called.

Description

The R_TSIP_TdesEcbDecryptUpdate() function uses the handle specified by the first argument, handle, and decrypts the contents of the second argument, cipher, using the key_index specified in the Init function, and writes the encrypted result to the third argument, plain. After ciphertext input finishes, call R_TSIP_TdesEcbDecryptFinal().

Ensure that plain and cipher are not assigned to overlapping areas. Also, specify RAM addresses for plain and cipher that are multiples of 4.

Reentrant

Not supported

4.2.4.9 R_TSIP_TdesEcbDecryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesEcbDecryptFinal(
        tsip_tdes_handle_t *handle,
        uint8_t *plain,
        uint32_t *plain_length
)
```

Parameters

handle	Input	TDES handler (work area)
plain	Output	plaintext data area (Nothing is ever written to this area.)
plain_length	Output	plaintext data length (Zero is always written to this area.)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	An illegal handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An illegal function was called.

Description

The R_TSIP_TdesEcbDecryptFinal() function writes the calculation result to the second argument, plain, and the length of the calculation result to the third argument, plain_length, using the handle specified by the first argument, handle. The leftover amount less than a multiple of 8 bytes was originally supposed to be encrypted and the result written to the second argument, but the Update function has a restriction that only allows it to handle values that are multiples of 8 bytes. Therefore, this function never actually writes anything to plain and it always writes 0 to plain_length. The arguments plain and plain_length are provided to ensure compatibility in case this restriction is removed in future.

Reentrant

Not supported

4.2.4.10 R_TSIP_TdesCbcEncryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesCbcEncryptInit(
        tsip_tdes_handle_t *handle,
        tsip_tdes_key_index_t *key_index,
        uint8_t *ivec
)
```

Parameters

handle	Output	TDES handler (work area)
key_index	Input	Key Index area
ivec	Input	initialization vector(8byte)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_KEY_SET:	Incorrect Key Index was input.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required for processing is in use by another processing routine.

Description

The R_TSIP_TdesCbcEncryptInit() function prepares to perform DES calculation and writes the result to the first argument, handle. The subsequent functions R_TSIP_TdesCbcEncryptUpdate() function and R_TSIP_TdesCbcEncryptFinal() also use handle as an argument.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.4.11 R_TSIP_TdesCbcEncryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesCbcEncryptUpdate(
        tsip_tdes_handle_t *handle,
        uint8_t *plain,
        uint8_t *cipher,
        uint32_t plain_length
)
```

Parameters

handle	Input/Output	TDES handler (work area)
plain	Input	plaintext data area
cipher	Output	ciphertext data area
plain_length	Input	byte length of plaintext data (Must be a multiple of 8.)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	An illegal handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An illegal function was called.

Description

The R_TSIP_TdesCbcEncryptUpdate() function uses the handle specified by the first argument, handle, and encrypts the contents of the second argument, plain, using the key_index specified in the Init function, and writes the encrypted result to the third argument, cipher. After plaintext input finishes, call R_TSIP_TdesCbcEncryptFinal().

Ensure that plain and cipher are not assigned to overlapping areas. Also, specify RAM addresses for plain and cipher that are multiples of 4.

Reentrant

Not supported

4.2.4.12 R_TSIP_TdesCbcEncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesCbcEncryptFinal(
        tsip_tdes_handle_t *handle,
        uint8_t *cipher,
        uint32_t *cipher_length
)
```

Parameters

handle	Input	TDES handler (work area)
cipher	Output	ciphertext data area (Nothing is ever written to this area.)
cipher_length	Output	ciphertext data length (Zero is always written to this area.)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	An illegal handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An illegal function was called.

Description

The R_TSIP_TdesCbcEncryptFinal() function writes the calculation result to the second argument, cipher, and the length of the calculation result to the third argument, cipher_length, using the handle specified by the first argument, handle. The leftover amount less than a multiple of 8 bytes was originally supposed to be encrypted and the result written to the second argument, but the Update function has a restriction that only allows it to handle values that are multiples of 8 bytes. Therefore, this function never actually writes anything to cipher and it always writes 0 to cipher_length. The arguments cipher and cipher_length are provided to ensure compatibility in case this restriction is removed in future.

Reentrant

Not supported

4.2.4.13 R_TSIP_TdesCbcDecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesCbcDecryptInit(
        tsip_tdes_handle_t *handle,
        tsip_tdes_key_index_t *key_index,
        uint8_t *ivec
)
```

Parameters

handle	Output	TDES handler (work area)
key_index	Input	Key Index area
ivec	Input	initialization vector(16byte)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_KEY_SET:	Incorrect Key Index was input.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required for processing is in use by another processing routine.

Description

The R_TSIP_TdesCbcDecryptInit() function prepares to perform DES calculation and writes the result to the first argument, handle. The subsequent functions R_TSIP_TdesCbcDecryptUpdate() function and R_TSIP_TdesCbcDecryptFinal() also use handle as an argument.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.4.14 R_TSIP_TdesCbcDecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesCbcDecryptUpdate(
        tsip_tdes_handle_t *handle,
        uint8_t *cipher,
        uint8_t *plain,
        uint32_t cipher_length
)
```

Parameters

handle	Input/Output	TDES handler (work area)
cipher	Input	ciphertext data area
plain	Output	plaintext data area
cipher_length	Input	byte length of ciphertext data (Must be a multiple of 16.)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	An illegal handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An illegal function was called.

Description

The R_TSIP_TdesCbcDecryptUpdate() function uses the handle specified by the first argument, handle, and decrypts the contents of the second argument, cipher, using the key_index specified in the Init function, and writes the encrypted result to the third argument, plain. After ciphertext input finishes, call R_TSIP_TdesCbcDecryptFinal().

Ensure that plain and cipher are not assigned to overlapping areas. Also, specify RAM addresses for plain and cipher that are multiples of 4.

Reentrant

Not supported

4.2.4.15 R_TSIP_TdesCbcDecryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesCbcDecryptFinal(
        tsip_tdes_handle_t *handle,
        uint8_t *plain,
        uint32_t *plain_length
)
```

Parameters

handle	Input	TDES handler (work area)
plain	Output	plaintext data area (Nothing is ever written to this area.)
plain_length	Output	plaintext data length (Zero is always written to this area.)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	An illegal handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An illegal function was called.

Description

The R_TSIP_TdesCbcDecryptFinal() function writes the calculation result to the second argument, plain, and the length of the calculation result to the third argument, plain_length, using the handle specified by the first argument, handle. The leftover amount less than a multiple of 8 bytes was originally supposed to be encrypted and the result written to the second argument, but the Update function has a restriction that only allows it to handle values that are multiples of 8 bytes. Therefore, this function never actually writes anything to plain and it always writes 0 to plain_length. The arguments plain and plain_length are provided to ensure compatibility in case this restriction is removed in future.

Reentrant

Not supported

4.2.5 ARC4

4.2.5.1 R_TSIP_GenerateArc4KeyIndex

Format

```
 #include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateArc4KeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
         tsip_arc4_key_index_t *key_index
)
```

Parameters

encrypted_provisioning_key	Input	Provisioning key wrapped by the DLM server
iv	Input	Initialization vector used when generating encrypted_key
encrypted_key	Input	ARC4 user key with encrypted MAC appended
key_index	Output	ARC4 Key Index

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This API outputs an ARC4 Key Index. For encrypted_key, enter data encrypted with Provisioning Key as shown in 5.3.3 ARC4.

Refer 3.7.1 Key Injection and Update for how to generate encrypted_provisioning_key, iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.5.2 R_TSIP_UpdateArc4KeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateArc4KeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_arc4_key_index_t *key_index
)
```

Parameters

iv	Input	Initialization vector when generating encrypted_key
encrypted_key	Input	User key with MAC encrypted with Update Key Ring appended
key_index	Output	ARC4 Key Index

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This API updates the Key Index of an ARC4 key.

For encrypted_key, enter the data shown in 5.3.3 ARC4 encrypted with the Update Key Ring.

Refer 3.7.1 Key Injection and Update for how to generate iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.5.3 R_TSIP_GenerateArc4RandomKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateArc4RandomKeyIndex(
        tsip_arc4_key_index_t *key_index
)
```

Parameters

key_index	Output	ARC4 Key Index

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This API outputs an ARC4 Key Index.

This API generates a user key from a random number internally in the TSIP. Accordingly, user key input is unnecessary. By encrypting data using the Key Index that is output by this API, dead copying of data can be prevented.

Refer 3.7.1 Key Injection and Update for how to use key_index.

Reentrant

Not supported

4.2.5.4 R_TSIP_Arc4EncryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Arc4EncryptInit(
        tsip_arc4_handle_t *handle,
        tsip_arc4_key_index_t *key_index
)
```

Parameters

handle	Output	ARC4 handler (work area)
key_index	Input	ARC4 Key Index area

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET:	An invalid Key Index was input.

Description

The R_TSIP_Arc4EncryptInit() function performs preparations for the execution of an ARC4 calculation, and writes the result to the first argument, handle. The value of handle is used as an argument in the subsequent R_TSIP_Arc4EncryptUpdate() function and R_TSIP_Arc4EncryptFinal() function.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.5.5 R_TSIP_Arc4EncryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Arc4EncryptUpdate(
        tsip_arc4_handle_t *handle,
        uint8_t *plain,
        uint8_t *cipher,
        uint32_t plain_length
)
```

Parameters

handle	Input/Output	ARC4 handler (work area)
plain	Input	Plaintext data area
cipher	Output	Ciphertext data area
plain_length	Input	Byte length of plaintext data (must be a multiple of 16)

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Arc4EncryptUpdate() function encrypts the second argument, plain, utilizing the Key Index specified in the Init function, and writes the encryption result to the third argument, cipher. After plaintext input is completed, call R_TSIP_Arc4EncryptFinal().

Specify areas for plain and cipher not to overlap. For plain and cipher, specify RAM addresses that are multiples of 4.

Reentrant

Not supported

4.2.5.6 R_TSIP_Arc4EncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Arc4EncryptFinal(
        tsip_arc4_handle_t *handle,
        uint8_t *cipher,
        uint32_t *cipher_length
)
```

Parameters

handle	Input	ARC4 handler (work area)
cipher	Output	Ciphertext data area (nothing ever written here)
cipher_length	Output	Ciphertext data length (0 always written here)

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

Using the handle specified in the first argument, handle, the R_TSIP_Arc4EncryptFinal() function writes the calculation result to the second argument, cipher, and writes the length of the calculation result to the third argument, cipher_length. The original intent was for the portion of the encryption result that was not a multiple of 16 bytes to be written to the second argument. However, as a result of the restriction that only multiples of 16 can be input to the Update function, nothing is ever written to cipher, and 0 is always written to cipher_length. The arguments cipher and cipher_length are provided for compatibility in anticipation of the time when this restriction is lifted.

Reentrant

Not supported

4.2.5.7 R_TSIP_Arc4DecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Arc4DecryptInit(
        tsip_arc4_handle_t *handle,
        tsip_arc4_key_index_t *key_index
)
```

Parameters

handle	Output	ARC4 handler (work area)
key_index	Input	ARC4 Key Index area

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET:	An invalid Key Index was input.

Description

The R_TSIP_Arc4DecryptInit() function performs preparations for the execution of an ARC4 calculation, and writes the result to the first argument, handle. The value of handle is used as an argument in the subsequent R_TSIP_Arc4DecryptUpdate() function and R_TSIP_Arc4DecryptFinal() function.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.5.8 R_TSIP_Arc4DecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Arc4DecryptUpdate(
        tsip_arc4_handle_t *handle,
        uint8_t *cipher,
        uint8_t *plain,
        uint32_t cipher_length
)
```

Parameters

handle	Input / Output	ARC4 handler (work area)
cipher	Input	Ciphertext data area
plain	Output	Plaintext data area
cipher_length	Input	Byte length of Ciphertext data (must be a multiple of 16)

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Arc4DecryptUpdate() function decrypts the second argument, cipher, utilizing the Key Index specified in the Init function, and writes the decryption result to the third argument, plain. After ciphertext input is completed, call R_TSIP_Arc4DecryptFinal().

Specify areas for plain and cipher not to overlap. For plain and cipher, specify RAM addresses that are multiples of 4.

Reentrant

Not supported

4.2.5.9 R_TSIP_Arc4DecryptFinal

Format

```
#include “r_tsip_rx_if.h”
e_tsip_err_t R_TSIP_Aes128EcbDecryptFinal(
        tsip_aes_handle_t *handle,
        uint8_t *plain,
        uint32_t *plain_length
        )
```

Parameters

handle	Input	ARC4 handler (work area)
plain	Output	Plaintext data area (nothing ever written here)
plain_length	Output	Plaintext data length (0 always written here)

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

Using the handle specified in the first argument, handle, the R_TSIP_Arc4DecryptFinal() function writes the calculation result to the second argument, plain, and writes the length of the calculation result to the third argument, plain_length. The original intent was for the portion of the decryption result that was not a multiple of 16 bytes to be written to the second argument. However, as a result of the restriction that only multiples of 16 can be input to the Update function, nothing is ever written to plain, and 0 is always written to plain_length. The arguments plain and plain_length are provided for compatibility in anticipation of the time when this restriction is lifted.

Reentrant

Not supported

4.2.6 RSA

4.2.6.1 R_TSIP_GenerateRsa*XXX*PublicKeyIndex

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateRsa1024PublicKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_rsa1024_public_key_index_t *key_index
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateRsa2048PublicKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_rsa2048_public_key_index_t *key_index
)
```

(3)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateRsa3072PublicKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_rsa3072_public_key_index_t *key_index
)
```

(4)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateRsa4096PublicKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_rsa4096_public_key_index_t *key_index
)
```

Parameters

encrypted_provisioning_key	Input	Provisioning key wrapped by the DLM server
iv	Input	Initial vector used when generating encrypted_key
encrypted_key	Input	Encrypted RSA public key with MAC apended
key_index	Output	RSA public key Key Index
value		Public key
key_management_info1		Key management information
key_n		Modulus n (Plain text) (1) RSA 1024bit (2) RSA 2048bit (3) RSA 3072bit (4) RSA 4096bit
key_e		Exponent e(Plain text) (1) RSA 1024bit (2) RSA 2048bit (3) RSA 3072bit (4) RSA 4096bit
dummy		Reserved
key_management_info2		Key management information

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_FAIL	An internal error occurred.

Description

This API outputs a 1024-bit, 2048-bit, 3972-bit and 4096-bit RSA public key Key Index.

Ensure that the areas allocated for encrypted_key and key_index do not overlap.

For encrypted_key, enter data encrypted with Provisioning Key as shown in 5.3.4 RSA.

Refer 3.7.1 Key Injection and Update for how to generate encrypted_provisioning_key, iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.6.2 R_TSIP_GenerateRsa*XXX*PrivateKeyIndex

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateRsa1024PrivateKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_rsa1024_private_key_index_t *key_index
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateRsa2048PrivateKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_rsa2048_private_key_index_t *key_index
)
```

Parameters

encrypted_provisioning_key	Input	Provisioning key wrapped by the DLM server
iv	Input	Initial vector used when generating encrypted_key
encrypted_key	Input	Encrypted RSA private key with MAC
key_index	Output	RSA private key Key Index (1) RSA 1024bit (2) RSA 2048bit

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_FAIL	An internal error occurred.

Description

This API outputs a 1024-bit 2048-bit and RSA private user key Key Index.

Ensure that the areas allocated for encrypted_key and key_index do not overlap.

For encrypted_key, enter data encrypted with Provisioning Key as shown in 5.3.4 RSA.

Refer 3.7.1 Key Injection and Update for how to generate encrypted_provisioning_key, iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.6.3 R_TSIP_GenerateRsa*XXX*RandomKeyIndex

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateRsa1024RandomKeyIndex(
        tsip_rsa1024_key_pair_index_t *key_pair_index
)
```
(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateRsa2048RandomKeyIndex(
        tsip_rsa2048_key_pair_index_t *key_pair_index
)
```

Parameters

key_pair_index	Output	Key Index for RSA public key and private key pair
public		RSA public key Key Index
value		Public key
key_management_info1		Key management information
key_n		Modulus n (Plain text) (1) RSA 1024bit (2) RSA 2048bit
key_e		Exponent e (Plain text) (1) RSA 1024bit (2) RSA 2048bit
dummy		Reserved
key_management_info2		Key management information
private		RSA private key Key Index

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_FAIL	An internal error occurred. Key generation failed.

Description

This API outputs a Key Index for a 1024-bit and 2048-bit RSA public key and private key pair. The API generates a user key from a random value produced internally by the TSIP. Consequently, there is no need to input a user key. Dead copying of data can be prevented by encrypting the data using the Key Index output by this API. A public key Key Index is generated by key_pair_index->public, and a private key Key Index is generated by key_pair_index->private. As the public key exponent, only 0x00010001 is generated.

Refer 3.7.1 Key Injection and Update for how to use key_index.

key_pair_index->public is the same operation as the public key Key Index output from R_TSIP_GenerateRsa*XXX*PublicKeyIndex(), and Key_pair_index->private is the same operation as the private key Key Index output from R_TSIP_GenerateRsa*XXX*PrivateKeyIndex().

Reentrant

Not supported

4.2.6.4 R_TSIP_UpdateRsa*XXX*PublicKeyIndex

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateRsa1024PublicKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_rsa1024_public_key_index_t *key_index
)
```
(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateRsa2048PublicKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_rsa2048_public_key_index_t *key_index
)
```
(3)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateRsa3072PublicKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_rsa3072_public_key_index_t *key_index
)
```
(4)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateRsa4096PublicKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_rsa4096_public_key_index_t *key_index
)
```

Parameters

iv	Input	Initialization vector when generating encrypted_key
encrypted_key	Input	Public key encrypted with Update Key Ring with MAC
key_index	Output	RSA public key Key Index
value		Public key
key_management_info1		Key management information
key_n		Modulus n (Plain text) (1) RSA 1024bit (2) RSA 2048bit (3) RSA 3072bit (4) RSA 4096bit
key_e		Exponent e(Plain text) (1) RSA 1024bit (2) RSA 2048bit (3) RSA 3072bit (4) RSA 4096bit
dummy		Reserved
key_management_info2		Key management information

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_FAIL	An internal error occurred.

Description

This API updates an RSA 1024-bit, 2048-bit, 3072-bit and 4096-bit public key Key Index.

For encrypted_key, enter the data shown in 5.3.4 RSA encrypted with the Update Key Ring.

Refer 3.7.1 Key Injection and Update for how to generate iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.6.5 R_TSIP_UpdateRsa*XXX*PrivateKeyIndex

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateRsa1024PrivateKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_rsa1024_private_key_index_t *key_index
)
```
(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateRsa2048PrivateKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_rsa2048_private_key_index_t *key_index
)
```

Parameters

iv	Input	Initialization vector when generating encrypted_key
encrypted_key	Input	Private key encrypted with Update Key Ring with MAC appended
key_index	Output	RSA private key Key Index (1) RSA 1024bit (2) RSA 2048bit

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_FAIL	An internal error occurred.

Description

This API updates an RSA 1024-bit and 2048-bit private key Key Index.

For encrypted_key, enter the data shown in 5.3.4 RSA encrypted with the Update Key Ring.

Refer 3.7.1 Key Injection and Update for how to generate iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.6.6 R_TSIP_RsaesPkcs*XXX*Encrypt

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsaesPkcs1024Encrypt(
        tsip_rsa_byte_data_t *plain,
        tsip_rsa_byte_data_t *cipher,
        tsip_rsa1024_public_key_index_t *key_index
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsaesPkcs2048Encrypt(
        tsip_rsa_byte_data_t *plain,
        tsip_rsa_byte_data_t *cipher,
        tsip_rsa2048_public_key_index_t *key_index
)
```

(3)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsaesPkcs3072Encrypt(
        tsip_rsa_byte_data_t *plain,
        tsip_rsa_byte_data_t *cipher,
        tsip_rsa3072_public_key_index_t *key_index
)
```

(4)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsaesPkcs4096Encrypt(
        tsip_rsa_byte_data_t *plain,
        tsip_rsa_byte_data_t *cipher,
        tsip_rsa4096_public_key_index_t *key_index
)
```

Parameters

plain	Input	plaintext
pdata		Specifies pointer to array containing plaintext.
data_length		Specifies valid data length of plaintext array. data size ≤ public key n size – 11
cipher	Output	ciphertext
pdata		Specifies pointer to array containing ciphertext.
data_length		Inputs ciphertext buffer size. Outputs valid data length after encryption (public key n size).
key_index	Input	Inputs the RSA public key Key Index. (1) RSA 1024bit (2) RSA 2048bit (3) RSA 3072bit (4) RSA 4096bit

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required for processing is in use by another processing routine.

TSIP_ERR_KEY_SET	Incorrect Key Index was input.
TSIP_ERR_PARAMETER	Input data is illegal

Description

The R_TSIP_RsaesPkcs*XXX*Encrypt() function RSA-encrypts the plaintext input to the first argument, plain, according to RSAES-PKCS1-V1_5. It writes the encryption result to the second argument, cipher.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.6.7 R_TSIP_RsaesPkcs*XXX*Decrypt

Format

(1) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_RsaesPkcs1024Decrypt(
        tsip_rsa_byte_data_t *cipher,
        tsip_rsa_byte_data_t *plain,
        tsip_rsa1024_private_key_index_t *key_index
)
```
(2) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_RsaesPkcs2048Decrypt(
        tsip_rsa_byte_data_t *cipher,
        tsip_rsa_byte_data_t *plain,
        tsip_rsa2048_private_key_index_t *key_index
)
```

Parameters

cipher	Input	ciphertext
pdata		Specifies pointer to array containing ciphertext.
data_length		Specifies valid data length of ciphertext array. (public key n size)
plain	Output	plaintext
pdata		Specifies pointer to array containing plaintext.
data_length		Inputs plaintext buffer size. The following size is required. Plaintext buffer size >= public key n size -11
key_index	Input	Inputs the RSA private key Key Index. (1) RSA 1024bit (2) RSA 2048bit

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required for processing is in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.
TSIP_ERR_PARAMETER	Input data is illegal.

Description

The R_TSIP_RsaesPkcs*XXX*Decrypt() function RSA-decrypts the ciphertext input to the first argument, cipher, according to RSAES-PKCS1-V1_5. It writes the decryption result to the second argument, plain.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.6.8 R_TSIP_RsassaPkcs*XXX*SignatureGenerate

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsassaPkcs1024SignatureGenerate(
        tsip_rsa_byte_data_t *message_hash,
        tsip_rsa_byte_data_t *signature,
        tsip_rsa1024_private_key_index_t *key_index,
        uint8_t hash_type
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsassaPkcs2048SignatureGenerate(
        tsip_rsa_byte_data_t *message_hash,
        tsip_rsa_byte_data_t *signature,
        tsip_rsa2048_private_key_index_t *key_index,
        uint8_t hash_type
)
```

Parameters

message_hash	Input	Message or hash value to which to attach signature
pdata		Specifies pointer to array storing the message or hash value
data_length		Specifies effective data length of the array (Specify only when Message is selected)
data_type		Selects the data type of message_hash Message: 0 Hash value : 1
signature	Output	Signature text storage destination information
pdata		Specifies pointer to array storing the signature text
data_length		data length
key_index	Input	Inputs the RSA private key Key Index. (1) RSA 1024bit (2) RSA 2048bit
hash_type	Input	Hash type R_TSIP_RSA_HASH_MD5 R_TSIP_RSA_HASH_SHA1 R_TSIP_RSA_HASH_SHA256

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Invalid Key Index was input.
TSIP_ERR_PARAMETER	Input data is invalid.

Description

The R_TSIP_RsassaPkcs*XXX*SignatureGenerate() function generates, in accordance with RSASSA-PKCS1-V1_5, a signature from the message text or hash value that is input in the first argument, message_hash, using the private key Key Index input to the third argument, key_index, and writes the signature text to the second argument, signature. When a message is specified in the first argument, message_hash->data_type, a hash value is calculated for the message as specified by the fourth argument, hash_type. When specifying a hash value in the first argument, message_hash->data_type, a hash value calculated with a hash algorithm as specified by the fourth argument, hash_type, must be input to message_hash->pdata.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.6.9 R_TSIP_RsassaPkcs*XXX*SignatureVerification

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsassaPkcs1024SignatureVerification(
        tsip_rsa_byte_data_t *signature,
        tsip_rsa_byte_data_t *message_hash,
        tsip_rsa1024_public_key_index_t *key_index,
        uint8_t hash_type
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsassaPkcs2048SignatureVerification(
        tsip_rsa_byte_data_t *signature,
        tsip_rsa_byte_data_t *message_hash,
        tsip_rsa2048_public_key_index_t *key_index,
        uint8_t hash_type
)
```

(3)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsassaPkcs3072SignatureVerification(
        tsip_rsa_byte_data_t *signature,
        tsip_rsa_byte_data_t *message_hash,
        tsip_rsa3072_public_key_index_t *key_index,
        uint8_t hash_type
)
```

(4)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsassaPkcs4096SignatureVerification(
        tsip_rsa_byte_data_t *signature,
        tsip_rsa_byte_data_t *message_hash,
        tsip_rsa4096_public_key_index_t *key_index,
        uint8_t hash_type
)
```

Parameters

signature	Input	Signature text information to verify
pdata		Specifies pointer to array storing the signature text
message_hash	Input	Message text or hash value to verify
pdata		Specifies pointer to array storing the message or hash value
data_length		Specifies effective data length of the array (Specify only when Message is selected)
data_type		Selects the data type of message_hash Message : 0 Hash value : 1
key_index	Input	Inputs the RSA public key Key Index. (1) RSA 1024bit (2) RSA 2048bit (3) RSA 3072bit (4) RSA 4096bit

hash_type	Input	Hash type R_TSIP_RSA_HASH_MD5 R_TSIP_RSA_HASH_SHA1 R_TSIP_RSA_HASH_SHA256

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Invalid Key Index was input.
TSIP_ERR_AUTHENTICATION	Authentication failed
TSIP_ERR_PARAMETER	Input data is invalid.

Description

R_TSIP_RsassaPkcs*XXX*SignatureVerification() function verifies, in accordance with RSASSA-PKCS1-V1_5, the signature text input to the first argument signature, and the message text or hash value input to the second argument, message_hash, using the public key Key Index input to the third argument, key_index. When a message is specified in the second argument, message_hash->data_type, a hash value is calculated using the public key Key Index input to the third argument, key_index, and as specified by the fourth argument, hash_type. When specifying a hash value in the second argument, message_hash->data_type, a hash value calculated with a hash algorithm as specified by the fourth argument, hash_type, must be input to message_hash->pdata.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.7 ECC

4.2.7.1 R_TSIP_GenerateEccP*XXX*PublicKeyIndex

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP192PublicKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_public_key_index_t *key_index
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP224PublicKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_public_key_index_t *key_index
)
```

(3)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP256PublicKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_public_key_index_t *key_index
)
```

(4)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP386PublicKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_public_key_index_t *key_index
)
```

Parameters

encrypted_provisioning_key	Input	Provisioning key wrapped by the DLM server
iv	Input	Initial vector used when generating encrypted_key
encrypted_key	Input	Encrypted ECC public key with MAC value dded
key_index	Output	ECC public key Key Index
value		Public key
key_management_info		Key management information
key_q		(1) ECC P-192 Public key Q(Plain text) (2) ECC P-224 Public key Q(Plain text) (3) ECC P-256 Public key Q(Plain text) (4) ECC P-386 Public key Q(Plain text)

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required by the processing is in use by other processing.
TSIP_ERR_FAIL	An internal error occurred.

Description

This is an API for outputting an ECC P-192, P-224, P-256 and P-384 public key Key Index.

Ensure that the areas for the encrypted_key and key_index do not overlap.

For encrypted_key, enter data encrypted with Provisioning Key as shown in 5.3.5 ECC.

For the format of the public key plaintext data output by key_index-> value.key_q, see 5.4.2 ECC.

Refer 3.7.1 Key Injection and Update for how to generate encrypted_provisioning_key, iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.7.2 R_TSIP_GenerateEccP*XXX*PrivateKeyIndex

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP192PrivateKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_private_key_index_t *key_index
)
```
(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP224PrivateKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_private_key_index_t *key_index
)
```
(3)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP256PrivateKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_private_key_index_t *key_index
)
```
(4)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP386PrivateKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_private_key_index_t *key_index
)
```

Parameters

encrypted_provisioning_key	Input	Provisioning key wrapped by the DLM server
iv	Input	Initial vector used when generating encrypted_key
encrypted_key	Input	Encrypted ECC P-192 private key with MAC value added
key_index	Output	ECC private key Key Index (1) ECC P-192 private key Key Index (2) ECC P-224 private key Key Index (3) ECC P-256 private key Key Index (4) ECC P-386 private key Key Index

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required by the processing is in use by other processing.
TSIP_ERR_FAIL	An internal error occurred.

Description

This is an API for outputting an ECC P-192, P-224, P-256 and P-384 private key Key Index.

Ensure that the areas for the encrypted_key and key_index do not overlap.

For encrypted_key, enter data encrypted with Provisioning Key as shown in 5.3.5 ECC.

Refer 3.7.1 Key Injection and Update for how to generate encrypted_provisioning_key, iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.7.3 R_TSIP_GenerateEccP*XXX*RandomKeyIndex

Format

(1) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_GenerateEccP192RandomKeyIndex(
        tsip_ecc_key_pair_index_t *key_pair_index
)
```
(2) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_GenerateEccP224RandomKeyIndex(
        tsip_ecc_key_pair_index_t *key_pair_index
)
```
(3) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_GenerateEccP256RandomKeyIndex(
        tsip_ecc_key_pair_index_t *key_pair_index
)
```
(4) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_GenerateEccP386RandomKeyIndex(
        tsip_ecc_key_pair_index_t *key_pair_index
)
```

Parameters

key_pair_index	Output	Key Index es for ECC P-192 public key and private key pair (1) ECC P-192 (2) ECC P-224 (3) ECC P-256 (4) ECC P-384
->public		ECC public key Key Index
value		Public key
key_management_info		Key management information
key_q		Public key(Qx || Qy) (Plain text)
->private		Key management information

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required by the processing is in use by other processing.
TSIP_ERR_FAIL	An internal error occurred.

Description

This is an API for outputting Key Indexes for an ECC P-192, P-224, P-256 and P-384 public key and private key pair. This API generates a user key from a random number value internally within the TSIP. There is therefore no need to input a user key. It is possible to prevent dead copying of data by using the Key Index output by this API to encrypt the data. The public Key Index is generated in key_pair_index->public, and the private key Key Index is generated in key_pair_index->private.For the format of the public key plaintext data output by key_index-> value.key_q, see 5.4.2 ECC.

key_pair_index->public is the same operation as the public key Key Index output from R_TSIP_GenerateEccP*XXX*PublicKeyIndex(), and Key_pair_index->private is the same operation as the private key Key Index output from R_TSIP_GenerateEccP*XXX*PrivateKeyIndex().

Reentrant

Not supported

4.2.7.4 R_TSIP_UpdateEccP*XXX*PublicKeyIndex

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP192PublicKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_public_key_index_t *key_index
)
```
(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP224PublicKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_public_key_index_t *key_index
)
```
(3)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP256PublicKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_public_key_index_t *key_index
)
```
(4)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP384PublicKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_public_key_index_t *key_index
)
```

Parameters

iv	Input	Initialization vector when generating encrypted_key
encrypted_key	Input	Public key encrypted with Update Key Ring with MAC value added
key_index	Output	ECC public key Key Index
value		Public key
key_management_info		Key management information
key_q		Public key(Qx || Qy) (Plain text) (1) ECC P-192 (2) ECC P-224 (3) ECC P-256 (4) ECC P-384

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required by the processing is in use by other processing.
TSIP_ERR_FAIL	An internal error occurred.

Description

This is an API for updating the Key Index of an ECC P-192, P-224, P-256 and P-384 public key.

For encrypted_key, enter the data shown in 5.3.5 ECC encrypted with the Update Key Ring.

Refer 3.7.1 Key Injection and Update for how to generate iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.7.5 R_TSIP_UpdateEccP*XXX*PrivateKeyIndex

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP192PrivateKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_private_key_index_t *key_index
)
```
(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP224PrivateKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_private_key_index_t *key_index
)
```
(3)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP256PrivateKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_private_key_index_t *key_index
)
```
(4)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP384PrivateKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_ecc_private_key_index_t *key_index
)
```

Parameters

iv	Input	Initialization vector when generating encrypted_key
encrypted_key	Input	Private key encrypted with Update Key Ring with MAC value added
key_index	Output	ECC private key Key Index (1) ECC P-192 (2) ECC P-224 (3) ECC P-256 (4) ECC P-384

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required by the processing is in use by other processing.
TSIP_ERR_FAIL	An internal error occurred.

Description

This is an API for updating the Key Index of an ECC P-192 private key.

For encrypted_key, enter the data shown in 5.3.5 ECC encrypted with the Update Key Ring.

Refer 3.7.1 Key Injection and Update for how to generate iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.7.6 R_TSIP_EcdsaP*XXX*SignatureGenerate

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdsaP192SignatureGenerate(
        tsip_ecdsa_byte_data_t *message_hash,
        tsip_ecdsa_byte_data_t *signature,
        tsip_ecc_private_key_index_t *key_index
)
```
(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdsaP224SignatureGenerate(
        tsip_ecdsa_byte_data_t *message_hash,
        tsip_ecdsa_byte_data_t *signature,
        tsip_ecc_private_key_index_t *key_index
)
```
(3)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdsaP256SignatureGenerate(
        tsip_ecdsa_byte_data_t *message_hash,
        tsip_ecdsa_byte_data_t *signature,
        tsip_ecc_private_key_index_t *key_index
)
```
(4)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdsaP384SignatureGenerate(
        tsip_ecdsa_byte_data_t *message_hash,
        tsip_ecdsa_byte_data_t *signature,
        tsip_ecc_private_key_index_t *key_index
)
```

Parameters

message_hash	Input	Message or hash value to which to attach signature
pdata		Specifies pointer to array storing the message or hash value
data_length		Specifies effective data length of the array Specify only when Message is selected)
data_type		Selects the data type of message_hash Message : 0 (Can be used except (4)) Hash value : 1
signature	Output	Signature storage destination information
pdata		Specifies pointer to array signature (1) "0 padding(64bit) || signature r(192bit) || 0 padding(64bit) || signature s(192bit)" (2) "0 padding(32bit) || signature r(224bit) || 0 padding(32bit) || signature s(224bit)" (3) "signature r(256bit) || signature s(256bit)" (4) "signature r(384bit) || signature s(384bit)"
data_length		Data length (byte units)
key_index	Input	Input Key Index of ECC private key. (1) ECC P-192 (2) ECC P-224 (3) ECC P-256 (4) ECC P-384

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required by the processing is in use by other processing.
TSIP_ERR_KEY_SET	Invalid Key Index was input.
TSIP_ERR_FAIL	An internal error occurred.
TSIP_ERR_PARAMETER	Input data is invalid.

Description

(1) R_TSIP_EcdsaP192SignatureGenerate, (2) R_TSIP_EcdsaP224SignatureGenerate, and (3) R_TSIP_EcdsaP256SignatureGenerate, when using

When a message is specified in the first argument, message_hash->data_type, a SHA-256 hash of the message text input as the first argument, message_hash->pdata, is calculated, and the signature text is written to the second argument, signature, in accordance with ECDSA P-192, P-224 and P-256 using the private key Key Index input as the third argument, key_index.

When a hash value is specified in the first argument, message_hash->data_type, the signature text for the first XXX bits (=XXX/8 bytes) of the SHA-256 hash value input to the first argument, message_hash->pdata, is written to the second argument, signature, in accordance with ECDSA P-192, P-224 and P-256 using the private key Key Index input as the third argument, key_index.

(4) R_TSIP_EcdsaP384SignatureGenerate when using

The signature text for the first 48 bytes of the SHA-384 hash value input to the first argument, message_hash->pdata, is written to the second argument, signature, in accordance with ECDSA P-384 using the private key Key Index input as the third argument, key_index.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.7.7 R_TSIP_EcdsaP*XXX*SignatureVerification

Format

(1) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_EcdsaP192SignatureVerification(
        tsip_ecdsa_byte_data_t *signature,
        tsip_ecdsa_byte_data_t *message_hash,
        tsip_ecc_public_key_index_t *key_index
)
```

(2) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_EcdsaP224SignatureVerification(
        tsip_ecdsa_byte_data_t *signature,
        tsip_ecdsa_byte_data_t *message_hash,
        tsip_ecc_public_key_index_t *key_index
)
```

(3) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_EcdsaP256SignatureVerification(
        tsip_ecdsa_byte_data_t *signature,
        tsip_ecdsa_byte_data_t *message_hash,
        tsip_ecc_public_key_index_t *key_index
)
```

(4) #include "r_tsip_rx_if.h"
```
e_tsip_err_t R_TSIP_EcdsaP384SignatureVerification(
        tsip_ecdsa_byte_data_t *signature,
        tsip_ecdsa_byte_data_t *message_hash,
        tsip_ecc_public_key_index_t *key_index
)
```

Parameters

signature	Input	Signature text information to be verified
pdata		Specifies pointer to array storing signature text The signature format is (1) ”0 padding(64bit) || signature r(192bit) || 0 padding(64bit) || signature s(192bit)” (2) ”0 padding(32bit) || signature r(224bit) || 0 padding(32bit) || signature s(224bit)” (3) ”signature r(256bit) || signature s(256bit)” (4) ”signature r(384bit) || signature s(384bit)”
message_hash	Input	Message or hash value to be verified
pdata		Specifies pointer to array storing the message or hash value
data_length		Specifies effective data length of the array (Specify only when Message is selected)
data_type		Selects the data type of message_hash Message : 0 (Can be used except (4)) Hash value : 1
key_index	Input	Input Key Index of ECC public key. (1) ECC P-192 (2) ECC P-224 (3) ECC P-256 (4) ECC P-384

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required by the processing is in use by other processing.
TSIP_ERR_KEY_SET	Invalid Key Index was input.
TSIP_ERR_FAIL	An internal error occurred, or signature verification failed.
TSIP_ERR_PARAMETER	Input data is invalid.

Description

(1) R_TSIP_EcdsaP192SignatureVerification, (2) R_TSIP_EcdsaP224SignatureVerification, and (3) R_TSIP_EcdsaP256SignatureVerification, when using

When a message is specified in the second argument, message_hash->data_type, a SHA-256 hash of the message text input as the second argument, message_hash->pdata, is calculated, and the signature text input to the first argument, signature, is validated in accordance with ECDSA P-192 , P-224 and P-256 using the public key Key Index input as the third argument, key_index.

When a hash value is specified in the second argument, message_hash->data_type, the signature text for the first XXX bits (=XXX/8 bytes) of the SHA-256 hash value input to the second argument, message_hash->pdata, input to the first argument, signature, is validated in accordance with ECDSA P-192, P-224 and P-256 using the public key Key Index input as the third argument, key_index.

(4) R_TSIP_EcdsaP384SignatureVerification when using

The signature text for the first 48 bytes of the SHA-384 hash value input to the second argument, message_hash->pdata, input to the first argument, signature, is validated in accordance with ECDSA P-384 using the public key Key Index input as the third argument, key_index.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.8 HASH

4.2.8.1 R_TSIP_Sha*XXX*Init

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Sha1Init(
        tsip_sha_md5_handle_t *handle
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Sha256Init(
        tsip_sha_md5_handle_t *handle
)
```

Parameters

handle	Output	SHA handler (work area)

Return Values

TSIP_SUCCESS :	Normal termination

Description

The R_TSIP_Sha*XXX*Init() function performs preparations for the execution of an SHA1 or SHA256 hash calculation, and writes the result to the first argument, handle. The value of handle is used as an argument in the subsequent R_TSIP_Sha*XXX*Update() function and R_TSIP_Sha1Final() function.

Reentrant

Not supported

4.2.8.2 R_TSIP_Sha*XXX*Update

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Sha1Update(
        tsip_sha_md5_handle_t *handle,
        uint8_t *message,
        uint32_t message_length
)
```
(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Sha256Update(
        tsip_sha_md5_handle_t *handle,
        uint8_t *message,
        uint32_t message_length
)
```

Parameters

handle	Input/Output	SHA handler (work area)
message	Input	message data area
message_length	Input	message data length

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Sha*XXX*Update() function calculates a hash value based on the second argument, message, and the third argument, message_length, utilizing in the first argument, handle, and writes the ongoing status to this first argument (and the value can be gotten with R_TSIP_GetCurrentHashDigestValue()). After message input is completed, call R_TSIP_Sha*XXX*Final().

Reentrant

Not supported

4.2.8.3 R_TSIP_Sha1Final

Format

(1) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_Sha1Final(
        tsip_sha_md5_handle_t *handle,
        uint8_t *digest,
        uint32_t *digest_length
)
```

(2) #include “r_tsip_rx_if.h”
```
e_tsip_err_t R_TSIP_Sha256Final(
        tsip_sha_md5_handle_t *handle,
        uint8_t *digest,
        uint32_t *digest_length
)
```

Parameters

handle	Input	SHA handler (work area)
digest	Output	hash data area
digest_length	Output	**hash data length** (1) SHA1 : 20byte (2) SHA256 : 32byte

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

Using the handle specified in the first argument, handle, the R_TSIP_Sha*XXX*Final() function writes the calculation result to the second argument, digest, and writes the length of the calculation result to the third argument, digest_length.

Reentrant

Not supported

4.2.8.4 R_TSIP_Md5Init

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Md5Init(
        tsip_sha_md5_handle_t *handle
)
```

Parameters

handle	Output	MD5 handler (work area)

Return Values

TSIP_SUCCESS :	Normal termination

Description

The R_TSIP_Md5Init() function prepares to calculate the MD5 hash and writes the result to the first argument, handle. The subsequent functions R_TSIP_Md5Update() and R_TSIP_Md5Final() also use handle as an argument.

Reentrant

Not supported

4.2.8.5 R_TSIP_Md5Update

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Md5Update(
        tsip_sha_md5_handle_t *handle,
        uint8_t *message,
        uint32_t message_length
)
```

Parameters

handle	Input/Output	MD5 handler (work area)
message	Input	message data area
message_length	Input	message data length in bytes

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required for processing is in use by another processing routine.
TSIP_ERR_PARAMETER:	An illegal handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An illegal function was called.

Description

The R_TSIP_Md5Update() function uses the handle specified by the first argument, handle, and calculates a hash value from the second argument, message, and the third argument, message_length, writing the progress along the way to the first argument, handle (and the value can be gotten with R_TSIP_GetCurrentHashDigestValue()). After message input completes, call R_TSIP_Md5Final().

Reentrant

Not supported

4.2.8.6 R_TSIP_Md5Final

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Sha1Final(
        tsip_sha_md5_handle_t *handle,
        uint8_t *digest,
        uint32_t *digest_length
)
```

Parameters

handle	Input	MD5 handler (work area)
digest	Output	hash data area
digest_length	Output	hash data length (16bytes)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required for processing is in use by another processing routine.
TSIP_ERR_PARAMETER:	An illegal handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An illegal function was called.

Description

The R_TSIP_Md5Final() function writes the calculation result to the second argument, digest, and the length of the calculation result to the third argument, digest_length, using the handle specified by the first argument handle.

Reentrant

Not supported

4.2.8.7 R_TSIP_GetCurrentHashDigestValue

Format

```
#include “r_tsip_rx_if.h”
e_tsip_err_t R_TSIP_GetCurrentHashDigestValue(
        tsip_sha_md5_handle_t *handle,
        uint8_t *digest,
        uint32_t *digest_length
)
```

Parameters

handle	Input	SHA,MD5 handler (work area)
digest	Output	current hash data area
digest_length	Output	current hash data length (16, 20, 32 byte)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_PARAMETER:	An illegal handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An illegal function was called.

Description

This function outputs the current value of the hash calculation after executing each Update() function[*1] to the second argument, digest, and the length of the calculation result to the third argument, digest_length, using the handle specified by the first argument handle.

Notes: 1. R_TSIP_Sha1Update(), R_TSIP_Sha256Update() or R_TSIP_Md5Update()

Reentrant

Not supported

4.2.9 HMAC

4.2.9.1 R_TSIP_GenerateSha*XXX*HmacKeyIndex

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateSha1HmacKeyIndex (
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_hmac_sha_key_index_t *key_index
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateSha256HmacKeyIndex (
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_hmac_sha_key_index_t *key_index
)
```

Parameters

encrypted_provisioning_key	Input	Provisioning key wrapped by the DLM server
iv	Input	Initialization vector when generating encrypted_key
encrypted_key	Input	User key encryptedand MAC appended
key_index	Output	Key Index

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required by the processing is in use by other processing.

Description

This API outputs an SHA1-HMAC or SHA256-HMAC Key Index.

For encrypted_key, enter data encrypted with Provisioning Key as shown in 5.3.6 SHA-HMAC.

Refer 3.7.1 Key Injection and Update for how to generate encrypted_provisioning_key, iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.9.2 R_TSIP_UpdateSha*XXX*HmacKeyIndex

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateSha1HmacKeyIndex (
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_hmac_sha_key_index_t *key_index
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateSha256HmacKeyIndex (
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_hmac_sha_key_index_t *key_index
)
```

Parameters

iv	Input	Initialization vector when generating encrypted_key
encrypted_key	Input	User key encrypted with Update Key Ring with MAC appended
key_index	Output	Key Index

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This API updates the Key Index of an SHA1-HMAC or SHA256-HMAC key.

For encrypted_key, enter the data shown in 5.3.6 SHA-HMAC encrypted with the Update Key Ring.

Refer 3.7.1 Key Injection and Update for how to generate iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.9.3 R_TSIP_ShaXXXHmacGenerateInit

Format

(1) #include "r_tsip_rx_if.h"

```
e_tsip_err_t R_TSIP_Sha1HmacGenerateInit(
        tsip_hmac_sha_handle_t *handle,
        tsip_hmac_sha_key_index_t *key_index
)
```

(2) #include "r_tsip_rx_if.h"

```
e_tsip_err_t R_TSIP_Sha256HmacGenerateInit(
        tsip_hmac_sha_handle_t *handle,
        tsip_hmac_sha_key_index_t *key_index
)
```

Parameters

handle	Output	SHA-HMAC handler (work area)
key_index	Input	MAC Key Index area

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	An invalid MAC Key Index was input.
TSIP_ERR_KEY_SET:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

The R_TSIP_Sha*XXX*HmacGenerateInit() function uses the second argument key_index to prepare for execution of SHA1-HMAC or SHA256-HMAC calculation, then writes the result to the first argument handle. When using the TLS cooperation function, use the MAC Key Index generated by the R_TSIP_TlsGenerateSessionKey() function as key_index. The argument handle is used by the subsequent R_TSIP_Sha*XXX*HmacGenerateUpdate() function or R_TSIP_Sha1HmacGenerateFinal() function.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.9.4 R_TSIP_Sha*XXX*HmacGenerateUpdate

Format

(1) #include “r_tsip_rx_if.h”

```
e_tsip_err_t R_TSIP_Sha1HmacGenerateUpdate(
        tsip_hmac_sha_handle_t *handle,
        uint8_t *message,
        uint32_t message_length
)
```

(2) #include “r_tsip_rx_if.h”

```
e_tsip_err_t R_TSIP_Sha256HmacGenerateUpdate(
        tsip_hmac_sha_handle_t *handle,
        uint8_t *message,
        uint32_t message_length
)
```

Parameters

handle	Input /Output	SHA-HMAC handle (work area)
message	Input	Message area
message_length	Input	Message length

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Sha*XXX*HmacGenerateUpdate() function uses the handle specified by the first argument handle, calculates a hash value from the second argument message and third argument message_length, then writes the intermediate result to the first argument handle. After message input finishes, call the R_TSIP_Sha*XXX*HmacGenerateFinal() function.

Reentrant

Not supported

4.2.9.5 R_TSIP_Sha*XXX*HmacGenerateFinal

Format

(1) #include "r_tsip_rx_if.h"

```
e_tsip_err_t R_TSIP_Sha1HmacGenerateFinal(
        tsip_hmac_sha_handle_t *handle,
        uint8_t *mac
)
```

(2) #include "r_tsip_rx_if.h"

```
e_tsip_err_t R_TSIP_Sha256HmacGenerateFinal(
        tsip_hmac_sha_handle_t *handle,
        uint8_t *mac
)
```

Parameters

handle	Input	SHA-HMAC handle (work area)
mac	Output	HMAC area (1) SHA1-HMAC : 20byte (2) SHA256-HMAC : 32byte

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Sha*XXX*HmacGenerateFinal() function uses the handle specified by the first argument handle and writes the calculation result to the second argument mac.

Reentrant

Not supported

4.2.9.6 R_TSIP_Sha*XXX*HmacVerifyInit

Format

(1) #include "r_tsip_rx_if.h"

```
e_tsip_err_t R_TSIP_Sha1HmacVerifyInit(
        tsip_hmac_sha_handle_t *handle,
        tsip_hmac_sha_key_index_t *key_index
)
```

(2) #include "r_tsip_rx_if.h"

```
e_tsip_err_t R_TSIP_Sha256HmacVerifyInit(
        tsip_hmac_sha_handle_t *handle,
        tsip_hmac_sha_key_index_t *key_index
)
```

Parameters

handle	Output	SHA-HMAC handler (work area)
key_index	Input	MAC Key Index area

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET:	An invalid MAC Key Index was input.

Description

The R_TSIP_Sha*XXX*HmacVerifyInit() function uses the first argument key_index to prepare for execution of SHA1-HMAC or SHA256-HMAC calculation, then writes the result to the first argument handle. When using the TLS cooperation function, use the MAC Key Index generated by the R_TSIP_TlsGenerateSessionKey() function as key_index. The argument handle is used by the subsequent R_TSIP_Sha*XXX*HmacVerifyUpdate() function or R_TSIP_Sha*XXX*HmacVerifyFinal() function.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.9.7 R_TSIP_Sha*XXX*HmacVerifyUpdate

Format

(1) #include "r_tsip_rx_if.h"

```
e_tsip_err_t R_TSIP_Sha1HmacVerifyUpdate(
        tsip_hmac_sha_handle_t *handle,
        uint8_t *message,
        uint32_t message_length
)
```

(2) #include "r_tsip_rx_if.h"

```
e_tsip_err_t R_TSIP_Sha256HmacVerifyUpdate(
        tsip_hmac_sha_handle_t *handle,
        uint8_t *message,
        uint32_t message_length
)
```

Parameters

handle	Input / Output	SHA-HMAC handle (work area)
message	Input	Message area
message_length	Input	Message length

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Sha*XXX*HmacVerifyUpdate() function uses the handle specified by the first argument handle, calculates a hash value from the second argument message and third argument message_length, then writes the intermediate result to the first argument handle. After message input finishes, call the R_TSIP_Sha*XXX*HmacVerifyFinal() function.

Reentrant

Not supported

4.2.9.8 R_TSIP_Sha*XXX*HmacVerifyFinal

Format

(1) #include “r_tsip_rx_if.h”

```
e_tsip_err_t R_TSIP_Sha1HmacVerifyFinal(
        tsip_hmac_sha_handle_t *handle,
        uint8_t *mac,
        uint32_t mac_length
)
```

(2) #include “r_tsip_rx_if.h”

```
e_tsip_err_t R_TSIP_Sha256HmacVerifyFinal(
        tsip_hmac_sha_handle_t *handle,
        uint8_t *mac,
        uint32_t mac_length
)
```

Parameters

handle	Input	SHA-HMAC handle (work area)
mac	Input	HMAC area
mac_length	Input	HMAC length

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred, or verification failed.
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_Sha*XXX*HmacVerifyFinal() function uses the handle specified by the first argument handle and verifies the mac value from the second argument mac and third argument mac_length. Input a value in bytes from 4 to 20 for SHA1-HMAC and bytes from 4 to 32 for SHA256-HMAC as mac_length.

<State transition>

The pre-run state is *TSIP enabled*.

After the function runs the state transitions to *TSIP enabled*.

Reentrant

Not supported

4.2.10 DH

4.2.10.1 R_TSIP_Rsa2048DhKeyAgreement

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Rsa2048DhKeyAgreement(
        tsip_aes_key_index_t *key_index,
        tsip_rsa2048_private_key_index_t *sender_private_key_index,
        uint8_t *message,
        uint8_t *receiver_modulus,
        uint8_t *sender_modulus
)
```

Parameters

key_index	Input	Key Index area for AES-128 CMAC operation
sender_private_key_index	Input	Private key generation information used in DH operation. The private key d included in the private key generation information is decrypted and used internally in the TSIP.
message	Input	Message (2048 bits) Set a value smaller than the prime number (d) included in sender_private_key_index.
receiver_modulus	Input	Modular exponentiation result calculated by the receiver + MAC 2048-bit modular exponentiation result || 128-bit MAC
sender_modulus	Output	Modular exponentiation result calculated by the sender + MAC 2048-bit modular exponentiation result || 128-bit MAC

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_KEY_SET:	Invalid Key Index was input.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_FAIL:	An internal error occurred.

Description

Performs DH operation using RSA-2048.

Note that the sender is the TSIP and the receiver is the other key exchange party.

<State transition>

The valid pre-run state is *TSIP Enabled State*.

After the function runs the state transitions to *TSIP Enabled State*.

Reentrant

Not supported

4.2.11 ECDH

4.2.11.1 R_TSIP_EcdhP256Init

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdhP256Init (
        tsip_ecdh_handle_t *handle,
        uint32_t key_type,
        uint32_t use_key_id
)
```

Parameters

handle	Output	ECDH handler (work area)
key_type	Input	Key exchange type 0 : ECDHE 1 : ECDH
user_key_id	Input	0 : key_id not used 1 : key_id used

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_PARAMETER:	Input data is invalid.

Description

The R_TSIP_EcdhP256Init function prepares to perform ECDH key exchange computation and writes the result to the first argument, handle. The succeeding functions R_TSIP_EcdhP256ReadPublicKey, R_TSIP_EcdhP256MakePublicKey, R_TSIP_EcdhP256CalculateSharedSecretIndex, and R_TSIP_EcdhP256KeyDerivation use handle as an argument.

Use the second argument, key_type, to select the type of ECDH key exchange. When ECDHE is selected, the R_TSIP_EcdhP256MakePublicKey function uses the TSIP's random number generation functionality to generate an ECC P-256 key pair. When ECDH is selected, keys installed beforehand are used for key exchange.

Input 1 as the third argument, use_key_id, to use key_id when key exchange is performed. key_id is for applications conforming to the DLMS/COSEM standard for smart meters.

Refer 3.7.1 Key Injection and Update for how to generate key_index.

Reentrant

Not supported

4.2.11.2 R_TSIP_EcdhP256ReadPublicKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdhP256ReadPublicKey (
        tsip_ecdh_handle_t *handle,
        tsip_ecc_public_key_index_t *public_key_index,
        uint8_t *public_key_data,
        tsip_ecdsa_byte_data_t *signature,
        tsip_ecc_public_key_index_t *key_index
)
```

Parameters

handle	Input/ Output	ECDH handler (work area)	
public_key_index	Input	Public key Key Index area for signature verification	
public_key_data	Input	key_id not used	ECC P-256 public key(512bit)
		key_id used	key_id (8bit) || public key s(512bit)
signature	Input	ECDSA P-256 signature of public_key_data	
key_index	Output	Key Index of public_key_data	

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required by the processing is in use by other processing.
TSIP_ERR_KEY_SET:	Invalid Key Index was input.
TSIP_ERR_FAIL:	An internal error occurred, or signature verification failed.
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_EcdhP256ReadPublicKey() function verifies the signature of the ECC P-256 public key of the other ECDH key exchange party. If the signature is correct, it outputs the public_key_data Key Index to the fifth argument.

The first argument, handle, is used as an argument in the subsequent function R_TSIP_EcdhP256CalculateSharedSecretIndex().

R_TSIP_EcdhP256CalculateSharedSecretIndex uses key_index as input to calculate Z.

Reentrant

Not supported

4.2.11.3 R_TSIP_EcdhP256MakePublicKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdhP256MakePublicKey (
        tsip_ecdh_handle_t *handle,
        tsip_ecc_public_key_index_t *public_key_index,
        tsip_ecc_private_key_index_t *private_key_index,
        uint8_t *public_key,
        tsip_ecdsa_byte_data_t *signature,
        tsip_ecc_private_key_index_t *key_index
)
```

Parameters

handle	Input /Output	ECDH handler (work area) When using key_id, input handle->key_id after running R_TSIP_EcdhP256Init().
public_key_index	Input	For ECDHE, input a null pointer. For ECDH, input the Key Index of a ECC P-256 public key.
private_key_index	Input	ECC P-256 private key for signature generation
public_key	Output	User public key (512-bit) for key exchange When using key_id, key_id (8-bit) || user public key (512-bit) || 0 padding (24-bit)
signature	Output	Signature text storage destination information
->pdata		: Specifies pointer to array for storing signature text Signature format: signature r (256-bit) || signature s (256-bit)"
->data_length		: Data length (in byte units)
key_index	Output	For ECDHE, a private key Key Index generated from a random number. Not output for ECDH.

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required by the processing is in use by other processing.
TSIP_ERR_KEY_SET:	Invalid Key Index was input.
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_EcdhP256MakePublicKey() function generates a temporary key pair (Ephemeral Key) and generates calculates a signature with the generate or input key. The signature generated is for DLMS/COSEM, the standard for smart meters.

If ECDHE is specified by the key_type argument of the R_TSIP_EcdhP256Init() function, the TSIP's random number generation functionality is used to generate an ECC P-256 key pair. The public key is output to public_key and the private key is output to key_index.

If ECDH is specified by the key_type argument of the R_TSIP_EcdhP256Init() function, the public key input as public_key_index is output to public_key and nothing is output to key_index.

The succeeding function R_TSIP_EcdhP256CalculateSharedSecretIndex() uses the first argument, handle, as an argument.

The R_TSIP_EcdhP256CalculateSharedSecretIndex() function uses key_index as input to calculate Z.

Reentrant

Not supported

4.2.11.4 R_TSIP_EcdhP256CalculateSharedSecretIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdhP256CalculateSharedSecretIndex (
        tsip_ecdh_handle_t *handle,
        tsip_ecc_public_key_index_t *public_key_index,
        tsip_ecc_private_key_index_t *private_key_index,
        tsip_ecdh_key_index_t *shared_secret_index
)
```

Parameters

handle	Input/Output	ECDH handler (work area)
public_key_index	Input	Public key Key Index whose signature was verified by R_TSIP_EcdhP256ReadPublicKey()
private_key_index	Input	Private key Key Index
shared_secret_index	Output	Key Index of shared secret Z calculated by ECDH key exchange

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required by the processing is in use by other processing.
TSIP_ERR_KEY_SET:	Invalid Key Index was input.
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_EcdhP256CalculateSharedSecretIndex() function uses the ECDH key exchange algorithm to output the Key Index of the shared secret Z derived from the public key of the other key exchange party and your own private key.

Input as the second argument, public_key_index, the public key Key Index whose signature was verified by R_TSIP_EcdhP256ReadPublicKey().

When key_type of R_TSIP_EcdhP256Init() is 0, input as the third argument, private_key_index, the private key Key Index generated from a random number by R_TSIP_EcdhP256MakePublicKey(), and when key_type is other than 0, input the private key Key Index that forms a pair with the second argument of R_TSIP_EcdhP256MakePublicKey().

The subsequent R_TSIP_EcdhP256KeyDerivation() function uses shared_secret_index as key material for outputting the Key Index.

Reentrant

Not supported

4.2.11.5 R_TSIP_EcdhP256KeyDerivation

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdhP256KeyDerivation (
        tsip_ecdh_handle_t *handle,
        tsip_ecdh_key_index_t *shared_secret_index,
        uint32_t key_type,
        uint32_t kdf_type
        uint8_t *other_info,
        uint32_t other_info_length,
        tsip_hmac_sha_key_index_t *salt_key_index,
        tsip_aes_key_index_t *key_index
)
```

Parameters

handle	Input/Output	ECDH handler (work area)
shared_secret_index	Input	Z Key Index calculated by R_TSIP_EcdhP256CalculateSharedSecretIndex
key_type	Input	Derived key type 0: AES-128 1: AES-256 2: SHA256-HMAC
kdf_type	Input	Algorithm used for key derivation calculation 0: AES-128 1: AES-256 2: SHA256-HMAC
other_info	Input	Additional data used for key derivation calculation AlgorithmID || PartyUInfo || PartyVInfo
other_info_length	Input	Data length of other_info (up to 147 byte units)
salt_key_index	Input	Salt Key Index (Input NULL when kdf_type is 0.)
key_index	Output	Key Index corresponding to key_type When the value of key_type is 2, an SHA256-HMAC Key Index is output. key_index can be specified by casting the start address of the area reserved beforehand by the tsip_hmac_sha_key_index_t type with the (tsip_aes_key_index_t*) type.

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required by the processing is in use by other processing.
TSIP_ERR_KEY_SET:	Invalid Key Index was input.
TSIP_ERR_PARAMETER:	An invalid handle was input.
TSIP_ERR_PROHIBIT_FUNCTION:	An invalid function was called.

Description

The R_TSIP_EcdhP256KeyDerivation() function uses the shared secret “Z (shared_secret_index)” calculated by the R_TSIP_EcdhP256CalculateSharedSecretIndex() function as the key material to derive the Key Index specified by the third argument, key_type.The key derivation algorithm is one-step key derivation as defined in NIST SP800-56C. Either SHA-256 or SHA-256 HMAC is specified by the fourth argument, kdf_type. When SHA-256 HMAC is specified, the Key Index output by the R_TSIP_GenerateSha256HmacKeyIndex() function or R_TSIP_UpdateSha256HmacKeyIndex() function is specified as the seventh argument, salt_key_index.

Enter a fixed value for deriving a key shared with the key exchange partner in the fifth argument, other_info.

A Key Index corresponding to key_type is output as the eighth argument, key_index. The correspondences between the types of derived key_index and the functions with which they can be used as listed below.

Derived Key Index	Compatible Functions
AES-128	All AES-128 Init functions and R_TSIP_Aes128KeyUnwrap()
AES-256	All AES-256 Init functions and R_TSIP_Aes256KeyUnwrap()
SHA256-HMAC	R_TSIP_Sha256HmacGenerateInit() and R_TSIP_Sha256HmacVerifyInit()

Reentrant

Not supported

4.2.11.6 R_TSIP_EcdheP512KeyAgreement

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdheP512KeyAgreement(
        tsip_aes_key_index_t *key_index,
        uint8_t *receiver_public_key,
        uint8_t *sender_public_key
)
```

Parameters

key_index	Input	Key Index area for AES-128 CMAC operation
receiver_public_key	Input	Receiver's Brainpool P512r1 public key Q Q(1024bit) || MAC(128bit)
sender_public_key	Output	Sender's Brainpool P512r1 public key Q (1024-bit) Q(1024bit) || MAC(128bit)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET:	Invalid Key Index was input.
TSIP_ERR_FAIL:	An internal error occurred.

Description

Performs an ECDHE operation after generation of a key pair using Brainpool P512r1.

Note that the sender is the TSIP and the receiver is the other key exchange party.

Reentrant

Not supported

4.2.12 KeyWrap

4.2.12.1 R_TSIP_Aes*XXX*KeyWrap

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128KeyWrap (
        tsip_aes_key_index_t *wrap_key_index,
        uint32_t target_key_type,
        tsip_aes_key_index_t *target_key_index,
        uint32_t *wrapped_key
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256KeyWrap (
        tsip_aes_key_index_t *wrap_key_index,
        uint32_t target_key_type,
        tsip_aes_key_index_t *target_key_index,
        uint32_t *wrapped_key
)
```

Parameters

wrap_key_index	Input	(1) AES-128 Key Index used for wrapping (2) AES-256 Key Index used for wrapping
target_key_type	Input	Selects key to be wrapped 0 (R_TSIP_KEYWRAP_AES128): AES-128 2 (R_TSIP_KEYWRAP_AES256): AES-256 Other: Reserved
target_key_index	Input	Key Index to be wrapped target_key_type 0: 13 word size target_key_type 2: 17 word size
wrapped_key	Output	Wrapped key target_key_type 0 : 6 word size target_key_type 2 : 10 word size

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required by the processing is in use by other processing.
TSIP_ERR_KEY_SET	Invalid Key Index was input.
TSIP_ERR_FAIL	An internal error occurred.

Description

The R_TSIP_Aes*XXX*KeyWrap() function uses wrap_key_index, the first argument, to wrap target_key_index, which is input as the third argument. The wrapped key is written to the fourth argument, wrapped_key. This processing conforms to the RFC3394 wrapping algorithm. Use the second argument, target_key_type, to select the key to be wrapped.

Use R_TSIP_Aes128KeyWrap() when the key length used for wrap is 128 bits, and use R_TSIP_Aes256KeyWrap() when the key length used for wrap is 256 bits.

Reentrant

Not supported

4.2.12.2 R_TSIP_Aes128KeyUnwrap

Format

(1)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128KeyUnwrap (
        tsip_aes_key_index_t *wrap_key_index,
        uint32_t target_key_type,
        uint32_t *wrapped_key,
        tsip_aes_key_index_t *target_key_index
)
```

(2)
```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256KeyUnwrap (
        tsip_aes_key_index_t *wrap_key_index,
        uint32_t target_key_type,
        uint32_t *wrapped_key,
        tsip_aes_key_index_t *target_key_index
)
```

Parameters

wrap_key_index	Input	(1) AES-128 Key Index used for unwrapping (2) AES-256 Key Index used for unwrapping
target_key_type	Input	Selects key to be unwrapped 0(R_TSIP_KEYWRAP_AES128): AES-128 2(R_TSIP_KEYWRAP_AES256): AES-256 Other: Reserved
wrapped_key	Input	Wrapped key target_key_type 0 : 6 word size target_key_type 2 : 10 word size
target_key_index	Output	Key Index target_key_type 0 : 13word size target_key_type 2 : 17 word size

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource required by the processing is in use by other processing.
TSIP_ERR_KEY_SET	Invalid Key Index was input.
TSIP_ERR_FAIL	An internal error occurred.

Description

The R_TSIP_Aes128KeyUnwrap function uses wrap_key_index, the first argument, to unwrap wrapped_key, which is input as the third argument. The unwrapped key is written to the fourth argument, target_key_index. This processing conforms to the RFC3394 unwrapping algorithm. Use the second argument, target_key_type, to select the key to be unwrapped.

When the key length used for unwrap is 128 bits, use R_TSIP_Aes128KeyUnwrap(); when the key length used for unwrap is 256 bits, use R_TSIP_Aes256KeyUnwrap().

Reentrant

Not supported

4.2.13 TLS (Common API)

4.2.13.1 R_TSIP_GenerateTlsRsaPublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateTlsRsaPublicKeyIndex(
        uint8_t *encrypted_provisioning_key,
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_tls_ca_certification_public_key_index_t *key_index
)
```

Parameters

encrypted_provisioning_key	Input	Provisioning key wrapped by the DLM server
iv	Input	Initial vector used when generating encrypted_key
encrypted_key	Input	2048-bit RSA public key encrypted in AES 128 ECB mode
key_index	Output	2048-bit RSA public key Key Index used by TLS cooperation function

Return Values

TSIP_SUCCESS	Normal termination
TSIP_ERR_FAIL	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_RESOURCE_CONFLICT	An internal error occurred.

Description

This API outputs a 2048-bit RSA public key Key Index used by the TLS cooperation function.

Refer to 5.3.4 RSA to get data format of Provisioning Key to be encrypted and input to encrypted_key.

Ensure that the areas allocated for encrypted_key and key_index do not overlap.

Refer to 3.7.1 Key Injection and Update for how to generate encrypted_provisioning_key, iv and encrypted_key and how to use key_index.

Reentrant

Not supported

4.2.13.2 R_TSIP_UpdateTlsRsaPublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateTlsRsaPublicKeyKeyIndex(
        uint8_t *iv,
        uint8_t *encrypted_key,
        tsip_tls_ca_public_key_index_t *key_index
)
```

Parameters

iv	Input	Initialization vector when generating encrypted_key
encrypted_key	Input	Public key encrypted Update Key Ring with MAC appended
key_index	Output	RSA 2048-bit public key Key Index used by TLS cooperation function

Return Values

TSIP_SUCCESS	Normal end
TSIP_ERR_FAIL	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This API outputs a 2048-bit RSA public key Key Index used by the TLS cooperation function.

Refer to XXXXX to get data format of Provisioning Key to be encrypted and input to encrypted_key.

Ensure that the areas allocated for encrypted_key and key_index do not overlap.

For the method of generating iv and encrypted_key, and instructions for using key_index, refer to 3.7.1 Key Injection and Update , Key Data Operations.

Reentrant

Not supported

4.2.13.3 R_TSIP_TlsRootCertificateVerification

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsRootCertificateVerification(
        uint32_t *public_key_type,
        uint8_t *certificate,
        uint32_t certificate_length,
        uint32_t public_key_n_start_position,
        uint32_t public_key_n_end_position,
        uint32_t public_key_e_start_position,
        uint32_t public_key_e_end_position,
        uint8_t *signature,
        uint32_t *encrypted_root_public_key
)
```

Parameters

public_key_type	Input	Public key type included in the certificate 0: RSA 2048-bit, 2: ECC P-256, other: reserved
certificate	Input	Root CA certificate bundle (DER format)
certificate_length	Input	Byte length of root CA certificate bundle
public_key_n_start_position	Input	Public key start byte position originating at the address specified by argument certificate public key public_key_type 0: n, 2: Qx
public_key_n_end_position	Input	Public key end byte position originating at the address specified by argument certificate public key public_key_type 0: n, 2: Qx
public_key_e_start_position	Input	Public key start byte position originating at the address specified by argument certificate public key public_key_type 0: e, 2: Qy
public_key_e_end_position	Input	Public key end byte position originating at the address specified by argument certificate public key public_key_type 0: e, 2: Qy
signature	Input	Signature data for root CA certificate bundle Input 256 bytes of signature data. The signature format is "RSA2048 PSS with SHA256".
encrypted_root_public_key	Output	Encrypted ECDSA P256 or RSA2048 public key used by R_TSIP_TlsCertificateVerification or R_TSIP_TlsCertificateVerificationExtension If the value of public_key_type is 0 then 560 bytes are output, and if 2 then 96 bytes.

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.

Description

This API verifies the root CA certificate bundle.

Reentrant

Not supported

4.2.13.4 R_TSIP_TlsCertificateVerification

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsCertificateVerification(
        uint32_t *public_key_type,
        uint32_t *encrypted_input_public_key,
        uint8_t *certificate,
        uint32_t certificate_length,
        uint8_t *signature,
        uint32_t public_key_n_start_position,
        uint32_t public_key_n_end_position,
        uint32_t public_key_e_start_position,
        uint32_t public_key_e_end_position,
        uint32_t *encrypted_output_public_key
)
```

Parameters

public_key_type	Input	Public key type included in the certificate 0: RSA 2048-bit (sha256WithRSAEncryption), 1: RSA 4096-bit (sha256WithRSAEncryption), 2: ECC P-256 (ecdsa-with-SHA256), 3: RSA 2048-bit (RSASSA-PSS), other: reserved
encrypted_input_public_key	Input	Encrypted public key output by R_TSIP_TlsRootCertificateVerification, R_TSIP_TlsCertificateVerification or R_TSIP_TlsCertificateVerificationExtension Data size public_key_type 0,1, 3: 140 words (560 byte), 2: 24 words (96 byte)
certificate	Input	Certificate bundle (DER format)
certificate_length	Input	Byte length of certificate bundle
signature	Input	Signature data for certificate bundle public_key_type:0 Data size is 256 byte Algorithm is sha256WithRSAEncryption public_key_type:1 Data size is 512 byte Algorithm is sha256WithRSAEncryption public_key_type:2 Data size is 64 byte "r(256bit) || s(256bit)" Algorithm is ecdsa-with-SHA256 public_key_type:3 Data size is 256 byte Algorithm is RSASSA-PSS {sha256, mgf1SHA256, 0x20, trailerFieldBC}
public_key_n_start_position	Input	Public key start byte position originating at the address specified by argument certificate public key public_key_type 0,1,3: n, 2: Qx
public_key_n_end_position	Input	Public key end byte position originating at the address specified by argument certificate public key public_key_type 0,1,3: n, 2: Qx
public_key_e_start_position	Input	Public key start byte position originating at the address specified by argument certificate public key public_key_type 0,1,3: n, 2: Qx

public_key_e_end_position	Input	Public key end byte position originating at the address specified by argument certificate public key public_key_type 0,1,3: n, 2: Qx
encrypted_root_public_key	Output	Encrypted public key used by R_TSIP_TlsCertificateVerification, R_TSIP_TlsCertificateVerificationExtension, R_TSIP_TlsEncryptPreMasterSecret WithRsa2048PublicKey or R_TSIP_TlsServersEphemeralEcdhPublicKeyRetrives Data size public_key_type 0,1,3: 140 words (560 byte), 2: 24 words (96 byte)

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This API verifies the signature in the server certificate or intermediate certificate.

This API can be used for same purpose with R_TSIP_TlsCertificateVerificationExtension().

Please use this function when the algorithm of verifying signature and that of obtaining key from certificate are same.

Reentrant

Not supported

4.2.13.5 R_TSIP_TlsCertificateVerificationExtension

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsCertificateVerificationExtension(
        uint32_t *public_key_type,
        uint32_t *public_key_output_type,
        uint32_t *encrypted_input_public_key,
        uint8_t *certificate,
        uint32_t certificate_length,
        uint8_t *signature,
        uint32_t public_key_n_start_position,
        uint32_t public_key_n_end_position,
        uint32_t public_key_e_start_position,
        uint32_t public_key_e_end_position,
        uint32_t *encrypted_output_public_key
)
```

Parameters

public_key_type	Input	Public key type included in the certificate 0: RSA 2048-bit (sha256WithRSAEncryption), 1: RSA 4096-bit (sha256WithRSAEncryption), 2: ECC P-256 (ecdsa-with-SHA256), 3: RSA 2048-bit (RSASSA-PSS), other: reserved
public_key_output_type	Input	Public key type to putput from the certificate 0: RSA 2048-bit (sha256WithRSAEncryption), 1: RSA 4096-bit (sha256WithRSAEncryption), 2: ECC P-256 (ecdsa-with-SHA256), 3: RSA 2048-bit (RSASSA-PSS), other: reserved
encrypted_input_public_key	Input	Encrypted public key output by R_TSIP_TlsRootCertificateVerification, R_TSIP_TlsCertificateVerification or R_TSIP_TlsCertificateVerificationExtension Data size public_key_type 0,1, 3: 140 words (560 byte), 2: 24 words (96 byte)
certificate	Input	Certificate bundle (DER format)
certificate_length	Input	Byte length of certificate bundle
signature	Input	Signature data for certificate bundle public_key_type:0 Data size is 256 byte Algorithm is sha256WithRSAEncryption public_key_type:1 Data size is 512 byte Algorithm is sha256WithRSAEncryption public_key_type:2 Data size is 64 byte "r(256bit) || s(256bit)" Algorithm is ecdsa-with-SHA256 public_key_type:3 Data size is 256 byte Algorithm is RSASSA-PSS {sha256, mgf1SHA256, 0x20, trailerFieldBC}
public_key_n_start_position	Input	Public key start byte position originating at the address specified by argument certificate

		public key public_key_type 0,1,3: n, 2: Qx
public_key_n_end_position	Input	Public key end byte position originating at the address specified by argument certificate public key public_key_type 0,1,3: n, 2: Qx
public_key_e_start_position	Input	Public key start byte position originating at the address specified by argument certificate public key public_key_type 0,1,3: n, 2: Qx
public_key_e_end_position	Input	Public key end byte position originating at the address specified by argument certificate public key public_key_type 0,1,3: n, 2: Qx
encrypted_output_public_key	Output	Encrypted public key used by R_TSIP_TlsCertificateVerification, R_TSIP_TlsCertificateVerificationExtension, R_TSIP_TlsEncryptPreMasterSecret WithRsa2048PublicKey or R_TSIP_TlsServersEphemeralEcdhPublicKeyRetrives Data size public_key_type 0,1,3: 140 words (560 byte), 2: 24 words (96 byte)

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This API verifies the signature in the server certificate or intermediate certificate.

This API can be used for same purpose with R_TSIP_TlsCertificateVerification().

Please use this function when the algorithm of verifying signature and that of obtaining key from certificate are fifferent.

Reentrant

Not supported

4.2.14 TLS (TLS1.2)

4.2.14.1 R_TSIP_TlsGeneratePreMasterSecret

Format

```
#include “r_tsip_rx_if.h”
e_tsip_err_t R_TSIP_TlsGeneratePreMasterSecret(
        uint32_t *tsip_pre_master_secret
)
```

Parameters

tsip_pre_master_secret	Output	pre-master secret data with TSIP-specific conversion used by R_TSIP_TlsGenerateMasterSecre, R_TSIP_TlsEncryptPreMasterSecretWithRsa2048PublicKey or R_TSIP_TlsGenerateExtendedMasterSecret The data size is 80 bytes.

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.

Description

This API generates the encrypted PreMasterSecret.

Reentrant

Not supported

4.2.14.2 R_TSIP_TlsEncryptPreMasterSecretWithRsa2048PublicKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsEncryptPreMasterSecretWithRsa2048PublicKey(
        uint32_t *encryted_public_key,
        uint32_t *tsip_pre_master_secret,
        uint8_t *encrypted_pre_master_secret
)
```

Parameters

encrypted_public_key	Input	Public key data output by R_TSIP_TlsCertificateVerification or R_TSIP_TlsCertificateVerificationExtension 140 word size (560 byte)
tsip_pre_master_secret	Input	pre-master secret data with TSIP-specific conversion output by R_TSIP_TlsGeneratePreMasterSecret
encrypted_pre_master_secret	Output	pre-master secret data that was RSA-2048 encrypted using public_key

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This API RSA-2048 encrypts PreMasterSecret using the public key from the input data.

Reentrant

Not supported

4.2.14.3 R_TSIP_TlsGenerateMasterSecret

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsGenerateMasterSecret(
        uint32_t select_cipher_suite,
        uint32_t *tsip_pre_master_secret,
        uint8_t *client_random,
        uint8_t *server_random,
        uint32_t *tsip_master_secret
)
```

Parameters

select_cipher_suite	Input	Selected cipher suite 0:R_TSIP_TLS_RSA_WITH_AES_128_CBC_SHA 1:R_TSIP_TLS_RSA_WITH_AES_256_CBC_SHA 2:R_TSIP_TLS_RSA_WITH_AES_128_CBC_SHA256 3:R_TSIP_TLS_RSA_WITH_AES_256_CBC_SHA256 4:R_TSIP_TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 5:R_TSIP_TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 6:R_TSIP_TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 7:R_TSIP_TLS_ECDHE_RSSA_WITH_AES_128_GCM_SHA256
tsip_pre_master_secret	Input	Value output by R_TSIP_TlsGeneratePreMasterSecret or R_TSIP_TlsGeneratePreMasterSecretWithEccP256Key
client_random	Input	Value of 32-byte random number reported by ClientHello
server_random	Input	32-byte random number value reported by ServerHello
tsip_master_secret	Output	20 words (80 byte) of master secret data with TSIP-specific conversion is output.

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This API is used to generate the encrypted MasterSecret.

Reentrant

Not supported

4.2.14.4 R_TSIP_TlsGenerateSessionKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsGenerateSessionKey(
        uint32_t select_cipher_suite,
        uint32_t *tsip_master_secret,
        uint8_t *client_random,
        uint8_t *server_random,
        uint8_t *nonce_explicit,
        tsip_hmac_sha_key_index_t *client_mac_key_index,
        tsip_hmac_sha_key_index_t *server_mac_key_index,
        tsip_aes_key_index_t *client_crypt_key_index,
        tsip_aes_key_index_t *server_crypt_key_index,
        uint8_t *client_iv,
        uint8_t *server_iv
)
```

Parameters

select_cipher_suite	Input	cipher_suite number selection 0:R_TSIP_TLS_RSA_WITH_AES_128_CBC_SHA 1:R_TSIP_TLS_RSA_WITH_AES_256_CBC_SHA 2:R_TSIP_TLS_RSA_WITH_AES_128_CBC_SHA256 3:R_TSIP_TLS_RSA_WITH_AES_256_CBC_SHA256 4:R_TSIP_TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 5:R_TSIP_TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 6:R_TSIP_TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 7:R_TSIP_TLS_ECDHE_RSSA_WITH_AES_128_GCM_SHA256
tsip_master_secret	Input	master secret data with TSIP-specific conversion output by R_TSIP_TlsGenerateMasterSecret
client_random	Input	Value of 32-byte random number reported by ClientHello
server_random	Input	32-byte random number value reported by ServerHello
nonce_explicit	Input	Nonce used by cipher suite AES128GCM select_cipher_suite=6-7: 8 bytes
client_mac_key_index	Output	MAC Key Index for client -> server communication
server_mac_key_index	Output	MAC Key Index for server -> client communication
client_crypt_key_index	Output	Common Key Index for client -> server communication
server_crypt_key_index	Output	Common Key Index for server -> client communication
client_iv	Output	In case of select_cipher_suite = 0~5, IV to use in transmission from Client to Server (This is available when using NetX Duo with RX651/RX65N). Except the case, nothing is output.
server_iv	Output	In case of select_cipher_suite = 0~5, IV to use in reception from Server (This is available when using NetX Duo with RX651/RX65N). Except the case, nothing is output.

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This API is used to output keys for TLS communication.

Nothing is output for the client_iv or server_iv argument except the case which is described in the parameters.

Reentrant

Not supported

4.2.14.5 R_TSIP_TlsGenerateVerifyData

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsGenerateVerifyData(
        uint32_t select_verify_data,
        uint32_t *tsip_master_secret,
        uint8_t *hand_shake_hash,
        uint8_t *verify_data
)
```

Parameters

select_verify_data	Input	Client/server type selection 0: R_TSIP_TLS_GENERATE_CLIENT_VERIFY Generate ClientVerifyData 1: R_TSIP_TLS_GENERATE_SERVER_VERIFY Generate ServerVerifyData
tsip_master_secret	Input	master secret data with TSIP-specific conversion output by R_TSIP_TlsGenerateMasterSecret
hand_shake_hash	Input	SHA256 HASH value for entire TLS handshake message
verify_data	Output	VerifyData for Finished message

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This API is used to generate Verify data.

Reentrant

Not supported

4.2.14.6 R_TSIP_TlsServersEphemeralEcdhPublicKeyRetrieves

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsServersEphemeralEcdhPublicKeyRetrieves(
        uint32_t public_key_type,
        uint8_t *client_random,
        uint8_t *server_random,
        uint8_t *server_ephemeral_ecdh_public_key,
        uint8_t *server_key_exchange_signature,
        uint32_t *encrypted_public_key,
        uint32_t *encrypted_ephemeral_ecdh_public_key
)
```

Parameters

public_key_type	Input	Public key type 0 : RSA 2048bit, 1 : Reserved, 2 : ECDSA P-256
client_random	Input	Random number value (32 bytes) reported by ClientHello
server_random	Input	Random number value (32 bytes) reported by ServerHello
server_ephemeral_ecdh_public_key	Input	Ephemeral ECDH public key (uncompressed format) received by server 0padding(24bit) || 04(8bit) || Qx(256bit) || Qy(256bit)
server_key_exchange_signature	Input	ServerKeyExchange signature data public_key_type 0: 560 byte, 2: 96 byte
encrypted_public_key	Input	Encrypted public key for signature verification Encrypted public key data output by R_TSIP_CertificateVerification public_key_type 0: 140 words (560 byte), 2: 24 words (96 byte)
encrypted_ephemeral_ecdh_public_key	Output	Encrypted ephemeral ECDH public key Input to R_TSIP_TlsGeneratePreMasterSecretWithEccP256Key 24 word size (96 byte)

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

Verifies the ServerKeyExchange signature using the input public key data. If the signature is verified successfully, the ephemeral ECDH public key used by R_TSIP_TlsGeneratePreMasterSecretWithEccP256Key is encrypted and output.

Relevant cypher suites: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

Reentrant

Not supported

4.2.14.7 R_TSIP_GenerateTlsP256EccKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ GenerateTlsP256EccKeyIndex(
        tsip_tls_p256_ecc_key_index_t *tls_p256_ecc_key_index,
        uint8_t *ephemeral_ecdh_public_key
)
```

Parameters

tls_p256_ecc_key_index	Output	Key information for generating PreMasterSecret Input to R_TSIP_TlsGeneratePreMasterSecretWithEccP256Key
ephemeral_ecdh_public_key	Output	Ephemeral ECDH public key Public key Qx (256-bit) || public key Qy (256-bit)

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This is an API for generating a key pair from a random number used by the TLS cooperation function for elliptic curve cryptography over a 256-bit prime field.

Reentrant

Not supported

4.2.14.8 R_TSIP_TlsGeneratePreMasterSecretWithEccP256Key

Format

```
#include “r_tsip_rx_if.h”
e_tsip_err_t R_TSIP_ TlsGeneratePreMasterSecretWithEccP256Key(
        uint32_t *encrypted_public_key,
        tsip_tls_p256_ecc_key_index_t *tls_p256_ecc_key_index,
        uint32_t *tsip_pre_master_secret
)
```

Parameters

encrypted_public_key	Input	Encrypted ephemeral ECDH public key output by R_TSIP_TlsServersEphemeralEcdhPublicKey Retrieves
tls_p256_ecc_key_index	Input	Key information output by R_TSIP_GenerateTlsP256EccKeyIndex
tsip_pre_master_secret	Output	Outputs 64 bytes of pre-master secret data on which TSIP-specific conversion has been performed.

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This is an API for generating an encrypted PreMasterSecret using the input data.

Relevant cypher suites: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

Reentrant

Not supported

4.2.14.9 R_TSIP_TlsGenerateExtendedMasterSecret

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsGenerateExtendedMasterSecret(
        uint32_t select_cipher_suite,
        uint32_t *tsip_pre_master_secret,
        uint8_t *digest,
        uint32_t *extended_master_secret
)
```

Parameters

select_cipher_suite	Input	Selected cipher suite 2:R_TSIP_TLS_RSA_WITH_AES_128_CBC_SHA256 3:R_TSIP_TLS_RSA_WITH_AES_256_CBC_SHA256 4:R_TSIP_TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 5:R_TSIP_TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 6:R_TSIP_TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 7:R_TSIP_TLS_ECDHE_RSSA_WITH_AES_128_GCM_SHA256
tsip_pre_master_secret	Input	master secret data with TSIP-specific conversion output by R_TSIP_TlsGeneratePreMasterSecret or R_TSIP_TlsGeneratePreMasterSecretWithEccP256Key
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate concatenated handshake message such as (ClientHello||ServerHello||Certificate||ServerKeyExchange ||CertificateRequest||ServerHelloDone||Certificate ||ClientKeyExchange)
extended_master_secret	Output	20 words (80 byte) of extended master secret data with TSIP-specific conversion is output. Input to R_TSIP_TlsGenerateSessionKey or R_TSIP_TlsGenerateVerifyData

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This API is used to generate the encrypted ExtendedMasterSecret.

Reentrant

Not supported

4.2.15 TLS (TLS1.3)

4.2.15.1 R_TSIP_GenerateTls13P256EccKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ TlsGenerateTls13P256EccKeyIndex(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls_p256_ecc_key_index_t *key_index,
        uint8_t *ephemeral_ecdh_public_key
)
```

Parameters

handle	Input	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
key_index	Output	Ephemeral ECC secret key Key Index Input to R_TSIP_Tls13GenerateEcdhSharedSecret
ephemeral_ecdh_public_key	Output	Ephemeral ECDH public key Public key Qx (256-bit) || public key Qy (256-bit)

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This is an API for generating a key pair from a random number used by the TLS1.3 cooperation function for elliptic curve cryptography over a 256-bit prime field.

Reentrant

Not supported

4.2.15.2 R_TSIP_Tls13GenerateEcdheSharedSecret

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13GenerateEcdheSharedSecret(
        e_tsip_tls13_mode_t mode,
        uint8_t *server_public_key,
        tsip_tls_p256_ecc_key_index_t *key_index,
        tsip_tls13_ephemeral_shared_secret_key_index_t *shared_secret_key_index
)
```

Parameters

mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
server_public_key	Input	Public key provided by the server Qx (256-bit) || public key Qy (256-bit)
key_index	Input	Ephemeral ECC secret key Key Index Output by R_TSIP_Tls13GenerateEcdhSharedSecret
ephemeral_ecdh_public_key	Output	Ephemeral SharedSecret Key Index Input to R_TSIP_Tls13GenerateHandshakeSecret and R_TSIP_Tls13GenerateResumptionHandshakeSecret

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a SharedSecret Key Index from elliptic curve cryptography over a 256-bit prime field with using public key provided by the server and prepared private key used by the TLS1.3 cooperation function.

Cipher Suite: TLS_AES_128_GCM_SHA256, TLS_AES_128_CCM_SHA256

Key Exchange: ECDHE NIST P-256

Reentrant

Not supported

4.2.15.3 R_TSIP_Tls13GenerateHandshakeSecret

Format

```
#include “r_tsip_rx_if.h”
e_tsip_err_t R_TSIP_ Tls13GenerateHandshakeSecret(
        tsip_tls13_ephemeral_shared_secret_key_index_t *shared_secret_key_index,
        tsip_tls13_ephemeral_handshake_secret_key_index_t *handshake_secret_key_index
)
```

Parameters

shared_secret_key_index	Input	Ephemeral SharedSecret Key Index Output by R_TSIP_Tls13GenerateHandshakeSecret
handshake_secret_key_index	Output	Ephemeral HandshakeSecret Key Index Input to R_TSIP_Tls13GenerateClientHandshakeTrafficKey, R_TSIP_Tls13GenerateClientHandshakeTrafficKey and R_TSIP_Tls13GenerateMasterSecret

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a HandshakeSecret Key Index with using the SharedSecret Key Index used by the TLS1.3 cooperation function.

Reentrant

Not supported

4.2.15.4 R_TSIP_Tls13GenerateServerHandshakeTrraficKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13GenerateServerHandshakeTrafficKey(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_handshake_secret_key_index_t *handshake_secret_key_index,
        uint8_t *digest,
        tsip_aes_key_index_t *server_write_key_index,
        tsip_tls13_ephemeral_server_finished_key_index_t *server_finished_key_index
)
```

Parameters

handle	Output	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
handshake_secret_key_index	Input	Ephemeral HandshakeSecret Key Index Output by R_TSIP_Tls13GenerateHandshakeSecret or R_TSIP_Tls13GenerateResumptionHandshakeSecret
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate concatenated handshake message such as (ClientHello||ServerHello)
server_write_key_index	Output	Ephemeral ServerWriteKey Key Index Input to R_TSIP_Tls13DecryptInit
server_finished_key_index	Output	Ephemeral ServerFinishedKey Key Index Input to R_TSIP_Tls13ServerHandshakeVerification

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a ServerWriteKey Key Index and a ServerFinishedKey Key Index with using the HandshakeSecret Key Index used by the TLS1.3 cooperation function.

Reentrant

Not supported

4.2.15.5 R_TSIP_Tls13GenerateClientHandshakeTrraficKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13GenerateClientHandshakeTrafficKey(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_handshake_secret_key_index_t *handshake_secret_key_index,
        uint8_t *digest,
        tsip_aes_key_index_t *client_write_key_index,
        tsip_hmac_sha_key_index_t *client_finished_key_index
)
```

Parameters

handle	Output	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
handshake_secret_key_index	Input	Ephemeral HandshakeSecret Key Index Output by R_TSIP_Tls13GenerateHandshakeSecret or R_TSIP_Tls13GenerateResumptionHandshakeSecret
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate concatenated handshake message such as (ClientHello||ServerHello)
client_write_key_index	Output	Ephemeral ClientWriteKey Key Index Input to R_TSIP_Tls13EncryptInit
client_finished_key_index	Output	Ephemeral ClientFinishedKey Key Index Input to R_TSIP_Sha256HmacGenerateInit

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a ClientWriteKey Key Index and a ClientFinishedKey Key Index with using the HandshakeSecret Key Index used by the TLS1.3 cooperation function.

Reentrant

Not supported

4.2.15.6 R_TSIP_Tls13ServerHandshakeVerification

Format

```
#include “r_tsip_rx_if.h”
e_tsip_err_t R_TSIP_ Tls13ServerHandshakeVerification(
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_server_finished_key_index_t *server_finished_key_index,
        uint8_t *digest,
        uint8_t *server_finished,
        uint32_t *verify_data_index
)
```

Parameters

mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
server_finished_key_index	Input	Ephemeral ServerFinishedKey Key Index Output by R_TSIP_Tls13ServerHandshakeVerification
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate concatenated handshake message such as (ClientHello‖ServerHello‖EncryptedExtensions ‖CertificateRequest‖Certificate‖CertificateVerify)
server_finished	Input	Finished provided by the server Output by R_TSIP_Tls13DecryptFinal to decrypt handshake message
verify_data_index	Output	Result of server handshake verification Input to R_TSIP_Tls13GenerateMasterSecret 8 words (32 bytes)

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.
TSIP_ERR_VERIFICATION_FAIL	Handshake verification failed.

Description

This is an API for verifying the Finished provided from the server used by the TLS1.3 cooperation function.

Reentrant

Not supported

4.2.15.7 R_TSIP_Tls13GenerateMasterSecret

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13GenerateMasterSecret(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_handshake_secret_key_index_t *handshake_secret_key_index,
        uint32_t *verify_data_index,
        tsip_tls13_ephemeral_master_secret_key_index_t *master_secret_key_index
)
```

Parameters

handle	Output	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
handshake_secret_key_index	Input	Ephemeral HandshakeSecret Key Index Output by R_TSIP_Tls13GenerateHandshakeSecret
verify_data_index	Input	Result of server handshake verification Output by R_TSIP_Tls13GenerateMasterSecret
master_secret_key_index	Output	Ephemeral MasterSecret Key Index Input to R_TSIP_Tls13GenerateApplicationTrafficKey and R_TSIP_Tls13GeneratePreSharedKey

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a MasterSecret Key Index with using the HandshakeSecret Key Index used by the TLS1.3 cooperation function.

Reentrant

Not supported

4.2.15.8 R_TSIP_Tls13GenerateApplicationTrraficKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13GenerateApplicationTrafficKey(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_master_secret_key_index_t *master_secret_key_index,
        uint8_t *digest,
        tsip_tls13_ephemeral_app_secret_key_index_t *server_app_secret_key_index,
        tsip_tls13_ephemeral_app_secret_key_index_t *client_app_secret_key_index,
        tsip_aes_key_index_t *server_write_key_index,
        tsip_aes_key_index_t *client_write_key_index
)
```

Parameters

handle	Input / Output	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
master_secret_key_index	Input	Ephemeral MasterSecret Key Index Output by R_TSIP_Tls13GenerateMasterSecret
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate concatenated handshake message such as (ClientHello||ServerHello||EncryptedExtensions ||CertificateRequest||Certificate||CertificateVerify ||ServerFinished)
server_app_secret_key_index	Output	Ephemeral ServerApplicationTrafficSecret key index Input to R_TSIP_Tls13UpdateApplicationTrafficKey
client_app_secret_key_index	Output	Ephemeral ClientApplicationTrafficSecret key index Input to R_TSIP_Tls13UpdateApplicationTrafficKey
server_write_key_index	Output	Ephemeral ServerWriteKey Key Index Input to R_TSIP_Tls13DecryptInit
client_write_key_index	Output	Ephemeral ClientWriteKey Key Index Input to R_TSIP_Tls13EncryptInit

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a ServerWriteKey Key Index, a ClientWriteKey Key Index and each ApplicationTrafficSecret Key Indexes with using the MasterSecret Key Index used by the TLS1.3 cooperation function.

When the server sends Application Data before receiving ClientFinished, the TLS1.3 server function can detect the verification error of ClientFInished only if the server program is not manipulated. When this function is implemented to the system, please consider the risk.

Reentrant

Not supported

4.2.15.9 R_TSIP_Tls13UpdateApplicationTrraficKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13UpdateApplicationTrafficKey(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        e_tsip_tls13_update_key_type_t key_type,
        tsip_tls13_ephemeral_app_secret_key_index_t *input_app_secret_key_index,
        tsip_tls13_ephemeral_app_secret_key_index_t *output_app_secret_key_index,
        tsip_aes_key_index_t *app_write_key_index
)
```

Parameters

handle	Input / Output	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
key_type	Input	Key type to update TSIP_TLS13_UPDATE_SERVER_KEY : Server Application Traffic Secret TSIP_TLS13_UPDATE_CLIENT_KEY : Client Application Traffic Secret
input_app_secret_key_index	Input	Ephemeral Server/Client ApplicationTrafficSecret Key Index Output by R_TSIP_Tls13GenerateApplicationTrafficKey or R_TSIP_Tls13UpdateApplicationTrafficKey
output_app_secret_key_index	Output	Ephemeral Server/Client ApplicationTrafficSecret Key Index Input to R_TSIP_Tls13UpdateApplicationTrafficKey
app_write_key_index	Output	Ephemeral Server/ClientWriteKey Key Index Input to R_TSIP_Tls13EncryptInit or R_TSIP_Tls13DecryptInit

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.
TSIP_ERR_PARAMETER	Input data is illegal.

Description

This is an API for updating an ApplicationTrafficSecret Key Index and corresponding WriteKey Key Index with using the previous ApplicationTrafficSecret Key Index used by the TLS1.3 cooperation function.

Reentrant

Not supported

4.2.15.10 R_TSIP_Tls13GenerateResumptionMasterSecret

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13GenerateResumptionMasterSecret(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_master_secret_key_index_t *master_secret_key_index,
        uint8_t *digest,
        tsip_tls13_ephemeral_res_master_secret_key_index_t *res_master_secret_key_index
)
```

Parameters

handle	Input	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
master_secret_key_index	Input	Ephemeral MasterSecret Key Index Output by R_TSIP_Tls13GenerateHandshakeSecret
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate concatenated handshake message such as (ClientHello||ServerHello||EncryptedExtensions ||CertificateRequest||Certificate||CertificateVerify ||ServerFinished||Certificate||CertificateVerify ||ClientFinished)
res_master_secret_key_index	Output	Ephemeral ResumptionMasterSecret Key Index Input to R_TSIP_Tls13GeneratePreSharedKey

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a ResumptionMasterSecret Key Index with using the MasterSecret Key Index used by the TLS1.3 cooperation function.

According to RFC8446, ephemeral MasterSecret Key Index should be erased after ephemeral ResumptionMasterSecret Key Index is generated by this API.

Reentrant

Not supported

4.2.15.11 R_TSIP_Tls13GeneratePreSharesKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13GeneratePreSharedKey(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_res_master_secret_key_index_t *res_master_secret_key_index,
        uint8_t *ticket_nonce,
        uint32_t ticket_nonce_len,
        tsip_tls13_ephemeral_pre_shared_key_index_t *pre_shared_key_index
)
```

Parameters

handle	Input	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
res_master_secret_key_index	Input	Ephemeral ResumptionMasterSecret Key Index Output by R_TSIP_Tls13GenerateResumptionMasterSecret
ticket_nonce	Input	TicketNonce provided by server When the length of Ticket Nonce is not multiple of 16 byte, include 0 padding to be multiple of 16 byte.
ticket_nonce_len	Input	Byte length of ticket_nonce
pre_shared_key_index	Output	Ephemeral PreSharedKey Key Index Input to R_TSIP_Tls13GeneratePskBinderKey, R_TSIP_Tls13GenerateResumptionHandshakeSecret or R_TSIP_Tls13Generate0RttApplicationWriteKey

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a PreSharedKey Key Index with using the ResumptionMasterSecret Key Index and Ticket Nonce in New Session Ticket used by the TLS1.3 cooperation function.

Reentrant

Not supported

4.2.15.12 R_TSIP_Tls13GeneratePskBinderKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13GeneratePskBinderKey(
        tsip_tls13_handle_t *handle,
        tsip_tls13_ephemeral_pre_shared_key_index_t *pre_shared_key_index,
        tsip_hmac_sha_key_index_t *psk_binder_key_index
)
```

Parameters

handle	Input	Handler to indicate the session (work area)
pre_shared_key_index	Input	Ephemeral PreSharedKey Key Index Output by R_TSIP_Tls13GeneratePreSharedKey
psk_binder_key_index	Output	Ephemeral BinderKey Key Index Input to R_TSIP_Sha256HmacGenerateInit

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a BinderKey Key Index used by the TLS1.3 cooperation function.

Reentrant

Not supported

4.2.15.13 R_TSIP_Tls13GenerateResumptionHandshakeSecret

Format

```
#include “r_tsip_rx_if.h”
e_tsip_err_t R_TSIP_ Tls13GenerateResumptionHandshakeSecret(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_pre_shared_key_index_t *pre_shared_key_index,
        tsip_tls13_ephemeral_shared_secret_key_index_t *shared_secret_key_index,
        tsip_tls13_ephemeral_handshake_secret_key_index_t *handshake_secret_key_index
)
```

Parameters

handle	Input	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
pre_shared_key_index	Input	Ephemeral PreSharedKey Key Index Output by R_TSIP_Tls13GeneratePreSharedKey
shared_secret_key_index	Input	Ephemeral SharedSecret Key Index Output by R_TSIP_Tls13GenerateEcdheSharedSecret
handshake_secret_key_index	Output	Ephemeral HandshakeSecret Key Index Input to R_TSIP_Tls13GenerateServerHandshakeTrafficKey, R_TSIP_Tls13GenerateClientHandshakeTrafficKey or R_TSIP_Tls13GenerateMasterSecret

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a HandshakeSecret Key Index to use Resumption with using the PreSharedKey Key Index used by the TLS1.3 cooperation function.

Only PreSharedKey generated by TSIP is supported, and other PreSharedKey is not supported.

Reentrant

Not supported

4.2.15.14 R_TSIP_Tls13Generate0RttApplicationWriteKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13Generate0RttApplicationWriteKey(
        tsip_tls13_handle_t *handle,
        tsip_tls13_ephemeral_pre_shared_key_index_t *pre_shared_key_index,
        uint8_t *digest,
        tsip_aes_key_index_t *client_write_key_index
)
```

Parameters

handle	Input / Output	Handler to indicate the session (work area)
pre_shared_key_index	Input	Ephemeral PreSharedKey Key Index Output by R_TSIP_Tls13GeneratePreSharedKey
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate handshake message of ClientHello
client_write_key_index	Output	Ephemeral ClientWriteKey Key Index Input to R_TSIP_Tls13EncryptInit

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a ClientWriteKey Key Index to use in 0-RTT with using the PreSharedKey Key Index used by the TLS1.3 cooperation function.

As described in RFC8446 section 2.3, there are risks that the data is not forward secret and there are no guarantees of non-replay between connections when using 0-RTT. When this function is implemented to the system, please consider the risks.

Reentrant

Not supported

4.2.15.15 R_TSIP_Tls13CertificateVerifyGenerate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13CertificateVerifyGenerate(
        uint32_t *key_index,
        e_tsip_tls13_signature_scheme_type_t signature_scheme,
        uint8_t *digest,
        uint8_t *certificate_verify,
        uint32_t *certificate_verify_len
)
```

Parameters

key_index	Input	Private key Key Index to generate signature Output by R_TSIP_GenerateEccP256PrivateKeyIndex, R_TSIP_GenerateEccP256RandomKeyIndex, R_TSIP_UpdateEccP256PrivateKeyIndex, R_TSIP_GenerateRsa2048PrivateKeyIndex, R_TSIP_GenerateRsa2048RandomKeyIndex or R_TSIP_UpdateRsa2048PrivateKeyIndex with casting uint32_t *
signature_scheme	Input	Signature Algorithm
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate concatenated handshake message such as (ClientHello||ServerHello||EncryptedExtensions ||CertificateRequest||Certificate||CertificateVerify ||ServerFinished||Certificate)
certificate_verify	Output	CertificateVerify Output format is described in RFC8446 section 4.4.3
certificate_verify_len	Output	Byte length of certificate_verify

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.
TSIP_ERR_PARAMETER	Input data is illegal.

Description

This is an API for generating the CertificucateVerify sending to the server used by the TLS1.3 cooperation function. Supporting signature algorithm is ecdsa_secp256r1_sha256 and rsa_pss_rsae_sha256.

Reentrant

Not supported

4.2.15.16 R_TSIP_Tls13CertificateVerifyVerification

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13CertificateVerifyVerification(
        uint32_t *key_index,
        e_tsip_tls13_signature_scheme_type_t signature_scheme,
        uint8_t *digest,
        uint8_t *certificate_verify,
        uint32_t certificate_verify_len
)
```

Parameters

key_index	Input	ECC P-256 public key Key Index Output by R_TSIP_TlsCertificateVerification or R_TSIP_TlsCertificateVerificationExtension
signature_scheme	Input	Signature Algorithm
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate concatenated handshake message such as (ClientHello||ServerHello||EncryptedExtensions ||CertificateRequest||Certificate)
certificate_verify	Input	CertificateVerify Input format must be described in RFC8446 section 4.4.3
certificate_verify_len	Input	Byte length of certificate_verify

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred, or signature verification failed.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.
TSIP_ERR_PARAMETER	Input data is illegal.

Description

This is an API for verifying the CertifucateVerify received from the server used by the TLS1.3 cooperation function. Supporting signature algorithm is ecdsa-secp256r1_sha256 and rsa_pss_rsae_sha256.

Reentrant

Not supported

4.2.15.17 R_TSIP_GenerateTls13SVP256EccKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ TlsGenerateTls13SVP256EccKeyIndex(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls_p256_ecc_key_index_t *key_index,
        uint8_t *ephemeral_ecdh_public_key
)
```

Parameters

handle	Input	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
key_index	Output	Ephemeral ECC secret key Key Index Input to R_TSIP_Tls13SVGenerateEcdhSharedSecret
ephemeral_ecdh_public_key	Output	Ephemeral ECDH public key Public key Qx (256-bit) || public key Qy (256-bit)

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.

Description

This is an API for generating a key pair from a random number used by the TLS1.3 cooperation function (server function) for elliptic curve cryptography over a 256-bit prime field.

Reentrant

Not supported

4.2.15.18 R_TSIP_Tls13SVGenerateEcdheSharedSecret

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13SVGenerateEcdheSharedSecret(
        e_tsip_tls13_mode_t mode,
        uint8_t *client_public_key,
        tsip_tls_p256_ecc_key_index_t *key_index,
        tsip_tls13_ephemeral_shared_secret_key_index_t *shared_secret_key_index
)
```

Parameters

mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
client_public_key	Input	Public key provided by the client Qx (256-bit) || public key Qy (256-bit)
key_index	Input	Ephemeral ECC secret key Key Index Output by R_TSIP_Tls13SVGenerateEcdhSharedSecret
ephemeral_ecdh_public_key	Output	Ephemeral SharedSecret Key Index Input to R_TSIP_Tls13SVGenerateHandshakeSecret and R_TSIP_Tls13SVGenerateResumptionHandshakeSecret

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a SharedSecret Key Index from elliptic curve cryptography over a 256-bit prime field with using public key provided by the server and prepared private key used by the TLS1.3 cooperation function (server function).

Cipher Suite : TLS_AES_128_GCM_SHA256, TLS_AES_128_CCM_SHA256

Key Exchange: ECDHE NIST P-256

Reentrant

Not supported

4.2.15.19 R_TSIP_Tls13SVGenerateHandshakeSecret

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13SVGenerateHandshakeSecret(
        tsip_tls13_ephemeral_shared_secret_key_index_t *shared_secret_key_index,
        tsip_tls13_ephemeral_handshake_secret_key_index_t *handshake_secret_key_index
)
```

Parameters

shared_secret_key_index	Input	Ephemeral SharedSecret Key Index Output by R_TSIP_Tls13SVGenerateHandshakeSecret
handshake_secret_key_index	Output	Ephemeral HandshakeSecret Key Index Input to R_TSIP_Tls13SVGenerateClientHandshakeTrafficKey, R_TSIP_Tls13SVGenerateClientHandshakeTrafficKey and R_TSIP_Tls13SVGenerateMasterSecret

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a HandshakeSecret Key Index with using the SharedSecret Key Index used by the TLS1.3 cooperation function (server function).

Reentrant

Not supported

4.2.15.20 R_TSIP_Tls13SVGenerateServerHandshakeTrraficKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13SVGenerateServerHandshakeTrafficKey(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_handshake_secret_key_index_t *handshake_secret_key_index,
        uint8_t *digest,
        tsip_aes_key_index_t *server_write_key_index,
        tsip_hmac_sha_key_index_t *server_finished_key_index
)
```

Parameters

handle	Output	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
handshake_secret_key_index	Input	Ephemeral HandshakeSecret Key Index Output by R_TSIP_Tls13SVGenerateHandshakeSecret or R_TSIP_Tls13SVGenerateResumptionHandshakeSecret
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate concatenated handshake message such as (ClientHello||ServerHello)
server_write_key_index	Output	Ephemeral ServerWriteKey Key Index Input to R_TSIP_Tls13EncryptInit
server_finished_key_index	Output	Ephemeral ServerFinishedKey Key Index Input to R_TSIP_Sha256HmacGenerateInit

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a ServerWriteKey Key Index and a ServerFinishedKey Key Index with using the HandshakeSecret Key Index used by the TLS1.3 cooperation function (server function).

Reentrant

Not supported

4.2.15.21 R_TSIP_Tls13SVGenerateClientHandshakeTrraficKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13SVGenerateClientHandshakeTrafficKey(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_handshake_secret_key_index_t *handshake_secret_key_index,
        uint8_t *digest,
        tsip_aes_key_index_t *client_write_key_index,
        tsip_ephemeral_client_finished_key_index_t *client_finished_key_index
)
```

Parameters

handle	Output	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
handshake_secret_key_index	Input	Ephemeral HandshakeSecret Key Index Output by R_TSIP_Tls13SVGenerateHandshakeSecret or R_TSIP_Tls13SVGenerateResumptionHandshakeSecret
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate concatenated handshake message such as (ClientHello||ServerHello)
client_write_key_index	Output	Ephemeral ClientWriteKey Key Index Input to R_TSIP_Tls13DecryptInit
client_finished_key_index	Output	Ephemeral ClientFinishedKey Key Index Input to R_TSIP_Tls13ClientHandshakeVerifivation

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a ClientWriteKey Key Index and a ClientFinishedKey Key Index with using the HandshakeSecret Key Index used by the TLS1.3 cooperation function (server function).

Reentrant

Not supported

4.2.15.22 R_TSIP_Tls13SVClientHandshakeVerification

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13SVClientHandshakeVerification(
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_client_finished_key_index_t *client_finished_key_index,
        uint8_t *digest,
        uint8_t *client_finished
)
```

Parameters

mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
client_finished_key_index	Input	Ephemeral ClientFinishedKey Key Index Output by R_TSIP_Tls13SVGenerateServerHandshakeTrafficKey
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate concatenated handshake message such as (ClientHello||ServerHello||EncryptedExtensions ||CertificateRequest||Certificate||CertificateVerify)
client_finished	Input	Finished provided by the client Output by R_TSIP_Tls13DecryptFinal to decrypt handshake message

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.
TSIP_ERR_VERIFICATION_FAIL	Handshake verification failed.

Description

This is an API for verifying the Finished provided from the server used by the TLS1.3 cooperation function (server function).

If the return value from this API is TSIP_ERR_VERIFICATION_FAIL, stop the TLS communication which includes the Handshake verified by this API.

Reentrant

Not supported

4.2.15.23 R_TSIP_Tls13SVGenerateMasterSecret

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13SVGenerateMasterSecret(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_handshake_secret_key_index_t *handshake_secret_key_index,
        tsip_tls13_ephemeral_master_secret_key_index_t *master_secret_key_index
)
```

Parameters

handle	Output	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
handshake_secret_key_index	Input	Ephemeral HandshakeSecret Key Index Output by R_TSIP_Tls13SVGenerateHandshakeSecret
master_secret_key_index	Output	Ephemeral MasterSecret Key Index Input to R_TSIP_Tls13SVGenerateApplicationTrafficKey and R_TSIP_Tls13SVGeneratePreSharedKey

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a MasterSecret Key Index with using the HandshakeSecret Key Index used by the TLS1.3 cooperation function (server function).

Reentrant

Not supported

4.2.15.24 R_TSIP_Tls13SVGenerateApplicationTrraficKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13SVGenerateApplicationTrafficKey(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_master_secret_key_index_t *master_secret_key_index,
        uint8_t *digest,
        tsip_tls13_ephemeral_app_secret_key_index_t *server_app_secret_key_index,
        tsip_tls13_ephemeral_app_secret_key_index_t *client_app_secret_key_index,
        tsip_aes_key_index_t *server_write_key_index,
        tsip_aes_key_index_t *client_write_key_index
)
```

Parameters

handle	Input / Output	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
master_secret_key_index	Input	Ephemeral MasterSecret Key Index Output by R_TSIP_Tls13GenerateMasterSecret
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate concatenated handshake message such as (ClientHello||ServerHello||EncryptedExtensions ||CertificateRequest||Certificate||CertificateVerify ||ServerFinished)
server_app_secret_key_index	Output	Ephemeral ServerApplicationTrafficSecret key index Input to R_TSIP_Tls13SVUpdateApplicationTrafficKey
client_app_secret_key_index	Output	Ephemeral ClientApplicationTrafficSecret key index Input to R_TSIP_Tls13SVUpdateApplicationTrafficKey
server_write_key_index	Output	Ephemeral ServerWriteKey Key Index Input to R_TSIP_Tls13EncryptInit
client_write_key_index	Output	Ephemeral ClientWriteKey Key Index Input to R_TSIP_Tls13DecryptInit

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a ServerWriteKey Key Index, a ClientWriteKey Key Index and each ApplicationTrafficSecret Key Indexes with using the MasterSecret Key Index used by the TLS1.3 cooperation function (server function).

When the server sends Application Data before receiving ClientFinished, the TLS1.3 server function can detect the verification error of ClientFInished only if the server program is not manipulated. When this function is implemented to the system, please consider the risk.

Reentrant

Not supported

4.2.15.25 R_TSIP_Tls13SVUpdateApplicationTrraficKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13SVUpdateApplicationTrafficKey(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        e_tsip_tls13_update_key_type_t key_type,
        tsip_tls13_ephemeral_app_secret_key_index_t *input_app_secret_key_index,
        tsip_tls13_ephemeral_app_secret_key_index_t *output_app_secret_key_index,
        tsip_aes_key_index_t *app_write_key_index
)
```

Parameters

handle	Input / Output	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
key_type	Input	Key type to update TSIP_TLS13_UPDATE_SERVER_KEY : Server Application Traffic Secret TSIP_TLS13_UPDATE_CLIENT_KEY : Client Application Traffic Secret
input_app_secret_key_index	Input	Ephemeral Server/Client ApplicationTrafficSecret Key Index Output by R_TSIP_Tls13SVGenerateApplicationTrafficKey or R_TSIP_Tls13SVUpdateApplicationTrafficKey
output_app_secret_key_index	Output	Ephemeral Server/Client ApplicationTrafficSecret Key Index Input to R_TSIP_Tls13SVUpdateApplicationTrafficKey
app_write_key_index	Output	Ephemeral Server/ClientWriteKey Key Index Input to R_TSIP_Tls13EncryptInit or R_TSIP_Tls13DecryptInit

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.
TSIP_ERR_PARAMETER	Input data is illegal.

Description

This is an API for updating an ApplicationTrafficSecret Key Index and corresponding WriteKey Key Index with using the previous ApplicationTrafficSecret Key Index used by the TLS1.3 cooperation function (server function).

Reentrant

Not supported

4.2.15.26 R_TSIP_Tls13SVGenerateResumptionMasterSecret

Format

```
#include “r_tsip_rx_if.h”
e_tsip_err_t R_TSIP_ Tls13SVGenerateResumptionMasterSecret(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_master_secret_key_index_t *master_secret_key_index,
        uint8_t *digest,
        tsip_tls13_ephemeral_res_master_secret_key_index_t *res_master_secret_key_index
)
```

Parameters

handle	Input	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
master_secret_key_index	Input	Ephemeral MasterSecret Key Index Output by R_TSIP_Tls13SVGenerateHandshakeSecret
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate concatenated handshake message such as (ClientHello||ServerHello||EncryptedExtensions ||CertificateRequest||Certificate||CertificateVerify ||ServerFinished||Certificate||CertificateVerify ||ClientFinished)
res_master_secret_key_index	Output	Ephemeral ResumptionMasterSecret Key Index Input to R_TSIP_Tls13SVGeneratePreSharedKey

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a ResumptionMasterSecret Key Index with using the MasterSecret Key Index used by the TLS1.3 cooperation function (server function).

According to RFC8446, ephemeral MasterSecret Key Index should be erased after ephemeral ResumptionMasterSecret Key Index is generated by this API.

Reentrant

Not supported

4.2.15.27 R_TSIP_Tls13SVGeneratePreSharesKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13SVGeneratePreSharedKey(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_res_master_secret_key_index_t *res_master_secret_key_index,
        uint8_t *ticket_nonce,
        uint32_t ticket_nonce_len,
        tsip_tls13_ephemeral_pre_shared_key_index_t *pre_shared_key_index
)
```

Parameters

handle	Input	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
res_master_secret_key_index	Input	Ephemeral ResumptionMasterSecret Key Index Output by R_TSIP_Tls13SVGenerateResumptionMasterSecret
ticket_nonce	Input	TicketNonce provided by server When the length of Ticket Nonce is not multiple of 16 byte, include 0 padding to be multiple of 16 byte.
ticket_nonce_len	Input	Byte length of ticket_nonce
pre_shared_key_index	Output	Ephemeral PreSharedKey Key Index Input to R_TSIP_Tls13SVGeneratePskBinderKey, R_TSIP_Tls13SVGenerateResumptionHandshakeSecret or R_TSIP_Tls13SVGenerate0RttApplicationWriteKey

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a PreSharedKey Key Index with using the ResumptionMasterSecret Key Index and TicketNonce in NewSessionTicket used by the TLS1.3 cooperation function (server function).

Reentrant

Not supported

4.2.15.28 R_TSIP_Tls13SVGeneratePskBinderKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13SVGeneratePskBinderKey(
        tsip_tls13_handle_t *handle,
        tsip_tls13_ephemeral_pre_shared_key_index_t *pre_shared_key_index,
        tsip_hmac_sha_key_index_t *psk_binder_key_index
)
```

Parameters

handle	Input	Handler to indicate the session (work area)
pre_shared_key_index	Input	Ephemeral PreSharedKey Key Index Output by R_TSIP_Tls13SVGeneratePreSharedKey
psk_binder_key_index	Output	Ephemeral BinderKey Key Index Input to R_TSIP_Sha256HmacVerifyInit

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a BinderKey Key Index used by the TLS1.3 cooperation function (server function).

Reentrant

Not supported

4.2.15.29 R_TSIP_Tls13SVGenerateResumptionHandshakeSecret

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13SVGenerateResumptionHandshakeSecret(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_mode_t mode,
        tsip_tls13_ephemeral_pre_shared_key_index_t *pre_shared_key_index,
        tsip_tls13_ephemeral_shared_secret_key_index_t *shared_secret_key_index,
        tsip_tls13_ephemeral_handshake_secret_key_index_t *handshake_secret_key_index
)
```

Parameters

handle	Input	Handler to indicate the session (work area)
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
pre_shared_key_index	Input	Ephemeral PreSharedKey Key Index Output by R_TSIP_Tls13SVGeneratePreSharedKey
shared_secret_key_index	Input	Ephemeral SharedSecret Key Index Output by R_TSIP_Tls13SVGenerateEcdheSharedSecret
handshake_secret_key_index	Output	Ephemeral HandshakeSecret Key Index Input to R_TSIP_Tls13SVGenerateServerHandshakeTrafficKey, R_TSIP_Tls13SVGenerateClientHandshakeTrafficKey or R_TSIP_Tls13SVGenerateMasterSecret

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a HandshakeSecret Key Index to use Resumption with using the PreSharedKey Key Index used by the TLS1.3 cooperation function (server function).

Only PreSharedKey generated by TSIP is supported, and other PreSharedKey is not supported.

Reentrant

Not supported

4.2.15.30 R_TSIP_Tls13SVGenerate0RttApplicationWriteKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13SVGenerate0RttApplicationWriteKey(
        tsip_tls13_handle_t *handle,
        tsip_tls13_ephemeral_pre_shared_key_index_t *pre_shared_key_index,
        uint8_t *digest,
        tsip_aes_key_index_t *client_write_key_index
)
```

Parameters

handle	Input / Output	Handler to indicate the session (work area)
pre_shared_key_index	Input	Ephemeral PreSharedKey Key Index Output by R_TSIP_Tls13SVGeneratePreSharedKey
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate handshake message of ClientHello
client_write_key_index	Output	Ephemeral ClientWriteKey Key Index Input to R_TSIP_Tls13DecryptInit

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

This is an API for generating a ClientWriteKey Key Index to use in 0-RTT with using the PreSharedKey Key Index used by the TLS1.3 cooperation function (server function).

As described in RFC8446 section 2.3, there are risks that the data is not forward secret and there are no guarantees of non-replay between connections when using 0-RTT. When this function is implemented to the system, please consider the risks.

Reentrant

Not supported

4.2.15.31 R_TSIP_Tls13SVCertificateVerifyGenerate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13SVCertificateVerifyGenerate(
        uint32_t *key_index,
        e_tsip_tls13_signature_scheme_type_t signature_scheme,
        uint8_t *digest,
        uint8_t *certificate_verify,
        uint32_t *certificate_verify_len
)
```

Parameters

key_index	Input	Private key Key Index to generate signature Output by R_TSIP_GenerateEccP256PrivateKeyIndex, R_TSIP_GenerateEccP256RandomKeyIndex, R_TSIP_UpdateEccP256PrivateKeyIndex, R_TSIP_GenerateRsa2048PrivateKeyIndex, R_TSIP_GenerateRsa2048RandomKeyIndex or R_TSIP_UpdateRsa2048PrivateKeyIndex with casting uint32_t *
signature_scheme	Input	Signature Algorithm
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate concatenated handshake message such as (ClientHello||ServerHello||EncryptedExtensions ||CertificateRequest||Certificate||CertificateVerify ||ServerFinished||Certificate)
certificate_verify	Output	CertificateVerify Output format is described in RFC8446 section 4.4.3
certificate_verify_len	Output	Byte length of certificate_verify

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.
TSIP_ERR_PARAMETER	Input data is illegal.

Description

This is an API for generating the CertificucateVerify sending to the server used by the TLS1.3 cooperation function (server function). Supporting signature algorithm is ecdsa_secp256r1_sha256 and rsa_pss_rsae_sha256.

Reentrant

Not supported

4.2.15.32 R_TSIP_Tls13SVCertificateVerifyVerification

Format

```
#include “r_tsip_rx_if.h”
e_tsip_err_t R_TSIP_ Tls13SVCertificateVerifyVerification(
        uint32_t *key_index,
        e_tsip_tls13_signature_scheme_type_t signature_scheme,
        uint8_t *digest,
        uint8_t *certificate_verify,
        uint32_t certificate_verify_len
)
```

Parameters

key_index	Input	ECC P-256 public key Key Index Output by R_TSIP_TlsCertificateVerification or R_TSIP_TlsCertificateVerificationExtension
signature_scheme	Input	Signature Algorithm
digest	Input	Message hash calculated with SHA256 Output by R_TSIP_Sha256Final to calculate concatenated handshake message such as (ClientHello||ServerHello||EncryptedExtensions ||CertificateRequest||Certificate)
certificate_verify	Input	CertificateVerify Input format must be described in RFC8446 section 4.4.3
certificate_verify_len	Input	Byte length of certificate_verify

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred, or signature verification failed.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.
TSIP_ERR_PARAMETER	Input data is illegal.

Description

This is an API for verifying the CertifucateVerify received from the server used by the TLS1.3 cooperation function (server function). Supporting signature algorithm is ecdsa-secp256r1_sha256 and rsa_pss_rsae_sha256.

Reentrant

Not supported

4.2.15.33 R_TSIP_Tls13EncryptInit

Format

```
#include “r_tsip_rx_if.h”
e_tsip_err_t R_TSIP_ Tls13EncryptInit(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_phase_t phase,
        e_tsip_tls13_mode_t mode,
        e_tsip_tls13_cipher_suite_t cipher_suite,
        tsip_aes_key_index_t *key_index,
        uint32_t payload_length
)
```

Parameters

handle	Output	Handler to indicate the session (work area)
phase	Input	Communication phase TSIP_TLS13_PHASE_HANDSHAKE : Handshake phase TSIP_TLS13_PHASE_APPLICATION : Application phase
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
cipher_suite	Input	Cipher suite TSIP_TLS13_CIPHER_SUITE_AES_128_GCM_SHA256 : TLS_AES_128_GCM_SHA256 TSIP_TLS13_CIPHER_SUITE_AES_128_CCM_SHA256 : TLS_AES_128_CCM_SHA256
key_index	Input	Ephemeral Key Index to encrypt
payload_length	Input	Payload length

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

The R_TSIP_TLS13EncryptInit() function performs preparations for the execution of an encrypt calculation used by the TLS1.3 cooperation function, and writes the result to the first argument, handle. The value of handle is used as an argument in the subsequent R_TSIP_Tls13EncryptUpdate() function and R_TSIP_Tls13EncryptFinal() function.

Reentrant

Not supported

4.2.15.34 R_TSIP_Tls13EncryptUpdate

Format

```
#include “r_tsip_rx_if.h”
e_tsip_err_t R_TSIP_ Tls13EncryptUpdate(
        tsip_tls13_handle_t *handle,
        uint8_t *plain,
        uint8_t *cipher,
        uint32_t plain_length
)
```

Parameters

handle	Input / Output	Handler to indicate the session (work area)
plain	Input	Plaintext data area
cipher	Output	Ciphertext data area
plain_length	Input	Plaintext data length

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_PARAMETER	Input data is illegal.

Description

The R_TSIP_Tls13EncryptUpdate() function encrypts the plaintext specified in the second argument, plain, using the values specified for client_write_key_index in R_TSIP_Tls13EncryptInit(). Inside this function, the data that is input by the user is buffered until the input values of plain exceed 16 bytes. After the input data from plain reaches 16 bytes or more, the encryption result is output to the ciphertext data area specified in the third argument, cipher. The length of the plain to input is specified in the fourth argument, payload_length. For this, specify not the total byte count for the plain input data, but rather the data length to input when the user calls this function. If the input value plain is not divisible by 16 bytes, that will be padded inside the function.

Specify areas for plain and cipher not to overlap, excluding the case that they are same address.

Reentrant

Not supported

4.2.15.35 R_TSIP_Tls13EncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13EncryptFinal(
        tsip_tls13_handle_t *handle,
        uint8_t *cipher,
        uint32_t *cipher_length
)
```

Parameters

handle	Input / Output	Handler to indicate the session (work area)
cipher	Output	Ciphertext data area
cipher_length	Output	Ciphertext data length

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred, or signature verification failed.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_PARAMETER	Input data is illegal.

Description

If there is 16-byte fractional data indicated by the total data length of the value of plain that was input by R_TSIP_Tls13EncryptUpdate(), the R_TSIP_Tls13EncryptFinal() function will output the result of encrypting that fractional data to the ciphertext data area specified in the second argument, cipher. Here, the portion that does not reach 16 bytes will be padded with zeros. For cipher, specify RAM address that are multiples of 4.

Reentrant

Not supported

4.2.15.36 R_TSIP_Tls13DecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13EncryptInit(
        tsip_tls13_handle_t *handle,
        e_tsip_tls13_phase_t phase,
        e_tsip_tls13_mode_t mode,
        e_tsip_tls13_cipher_suite_t cipher_suite,
        tsip_aes_key_index_t *key_index,
        uint32_t payload_length
)
```

Parameters

handle	Output	Handler to indicate the session (work area)
phase	Input	Communication phase TSIP_TLS13_PHASE_HANDSHAKE : Handshake phase TSIP_TLS13_PHASE_APPLICATION : Application phase
mode	Input	Handshake protocol to use TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake TSIP_TLS13_MODE_RESUMPTION : Resumption TSIP_TLS13_MODE_0_RTT : 0-RTT
cipher_suite	Input	Cipher suite TSIP_TLS13_CIPHER_SUITE_AES_128_GCM_SHA256 : TLS_AES_128_GCM_SHA256 TSIP_TLS13_CIPHER_SUITE_AES_128_CCM_SHA256 : TLS_AES_128_CCM_SHA256
key_index	Input	Ephemeral Key Index to decrypt
payload_length	Input	Payload length

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Incorrect Key Index was input.

Description

The R_TSIP_TLS13DecryptInit() function performs preparations for the execution of a decrypt calculation used by the TLS1.3 cooperation function, and writes the result to the first argument, handle. The value of handle is used as an argument in the subsequent R_TSIP_Tls13DecryptUpdate() function and R_TSIP_Tls13DecryptFinal() function.

Reentrant

Not supported

4.2.15.37 R_TSIP_Tls13DecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13DecryptUpdate(
        tsip_tls13_handle_t *handle,
        uint8_t *cipher,
        uint8_t *plain,
        uint32_t cipher_length
)
```

Parameters

handle	Input / Output	Handler to indicate the session (work area)
cipher	Input	Ciphertext data area
plain	Output	Plaintext data area
cipher_length	Input	Ciphertext data length

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_PARAMETER	Input data is illegal.

Description

The R_TSIP_Tls13DecryptUpdate() function decrypts the ciphertext specified in the second argument, cipher, using the values specified for server_write_key_index in R_TSIP_Tls13DecryptInit(). Inside this function, the data that is input by the user is buffered until the input values of cipher exceed 16 bytes. After the input data from cipher reaches 16 bytes or more, the decryption result is output to the plaintext data area specified in the third argument, plain. The length of the cipher to input is specified in the fourth argument, cipher_length. For this, specify not the total byte count for the cipher input data, but rather the data length to input when the user calls this function. If the input value cipher is not divisible by 16 bytes, that will be padded inside the function.

Specify areas for plain and cipher not to overlap, excluding the case that they are same address.

Reentrant

Not supported

4.2.15.38 R_TSIP_Tls13DecryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_ Tls13DecryptFinal(
        tsip_tls13_handle_t *handle,
        uint8_t *plain,
        uint32_t *plain_length
)
```

Parameters

handle	Input / Output	Handler to indicate the session (work area)
plain	Output	Plaintext data area
plain_length	Output	Plaintext data length

Return Values

TSIP_SUCCESS :	Normal end
TSIP_ERR_FAIL:	An internal error occurred, or signature verification failed.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by this processing routine was in use by another processing routine.
TSIP_ERR_PARAMETER	Input data is illegal.

Description

If there is 16-byte fractional data indicated by the total data length of the value of cipher that was input by R_TSIP_Tls13DecryptUpdate(), the R_TSIP_Tls13DecryptFinal() function will output the result of decrypting that fractional data to the plaintext data area specified in the second argument, plain. Here, the portion that does not reach 16 bytes will be padded with zeros. For plain, specify RAM address that are multiples of 4.

Reentrant

Not supported

4.2.16 Firmware Update

4.2.16.1 R_TSIP_StartUpdateFirmware

Format

#include "r_tsip_rx_if.h"

e_tsip_err_t R_TSIP_StartUpdateFirmware(void);

Parameters

none

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.

Description

State Transit to the *Firm Update State*.

Reentrant

Not supported

4.2.16.2 R_TSIP_GenerateFirmwareMAC

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateFirmwareMAC(
        uint32_t *InData_KeyIndex,
        uint32_t *InData_SessionKey,
        uint32_t *InData_UpProgram,
        uint32_t *InData_IV,
        uint32_t *OutData_Program,
        uint32_t MAX_CNT,
        TSIP_GEN_MAC_CB_FUNC_T p_callback,
        tsip_firmware_generate_mac_resume_handle_t *tsip_firmware_generate_mac_resume_handle
)
```

Parameters

InData_KeyIndex	Input	Key Index area for decrypting InData_SessionKey and generating firmware MAC values
InData_SessionKey	Input	Session key area for decrypting encrypted firmware and verifying checksum values
InData_UpProgram	Input	512 words (2048 bytes) area for temporarily storing encrypted firmware data.
InData_IV	Input	Initial vector area for decrypting the encrypted firmware.
OutData_Program	Output	512 words (2048 bytes) area for temporarily storing decrypted firmware data.
MAX_CNT	Input	The word size for encrypted firmware+MAC word size. Encrypted firmware value should be a multiple of 4. MAC word size is 4 words (128bit). Encrypted firmware data minimum size is 16 words, so, MAX_CNT minimum size is 20.
p_callback	Input	It is called multiple times when user's action is required. The contents of teh action is determined by teh enum TSIP_FW_CB_REQ_TYPE.
tsip_firmware_generate_mac_resume_handle	Input	R_TSIP_GenerateFirmwaraMAC handler (work area)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	An internal error occurred.
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_KEY_SET	Input illegal Key Index.
TSIP_ERR_CALLBACK_UNREGIST	p_callback value is illlegal.
TSIP_ERR_PARAMETER	Input data is illegal.
TSIP_RESUME_FIRMWARE_GENERATE_MAC	There is additional processing. It is necessary to call the API again.

Description

This function decrypts the firmware and generates new MAC for the encrypted firmware and the firmware checksum value. User can update the firmware by writing the decrypted firmware and new MAC value to the Flash ROM. Refer to Section 8 for operating the firmware updates.

The encryption algorithm uses AES-CBC and the MAC uses AES-CMAC.This API is called in the following order.

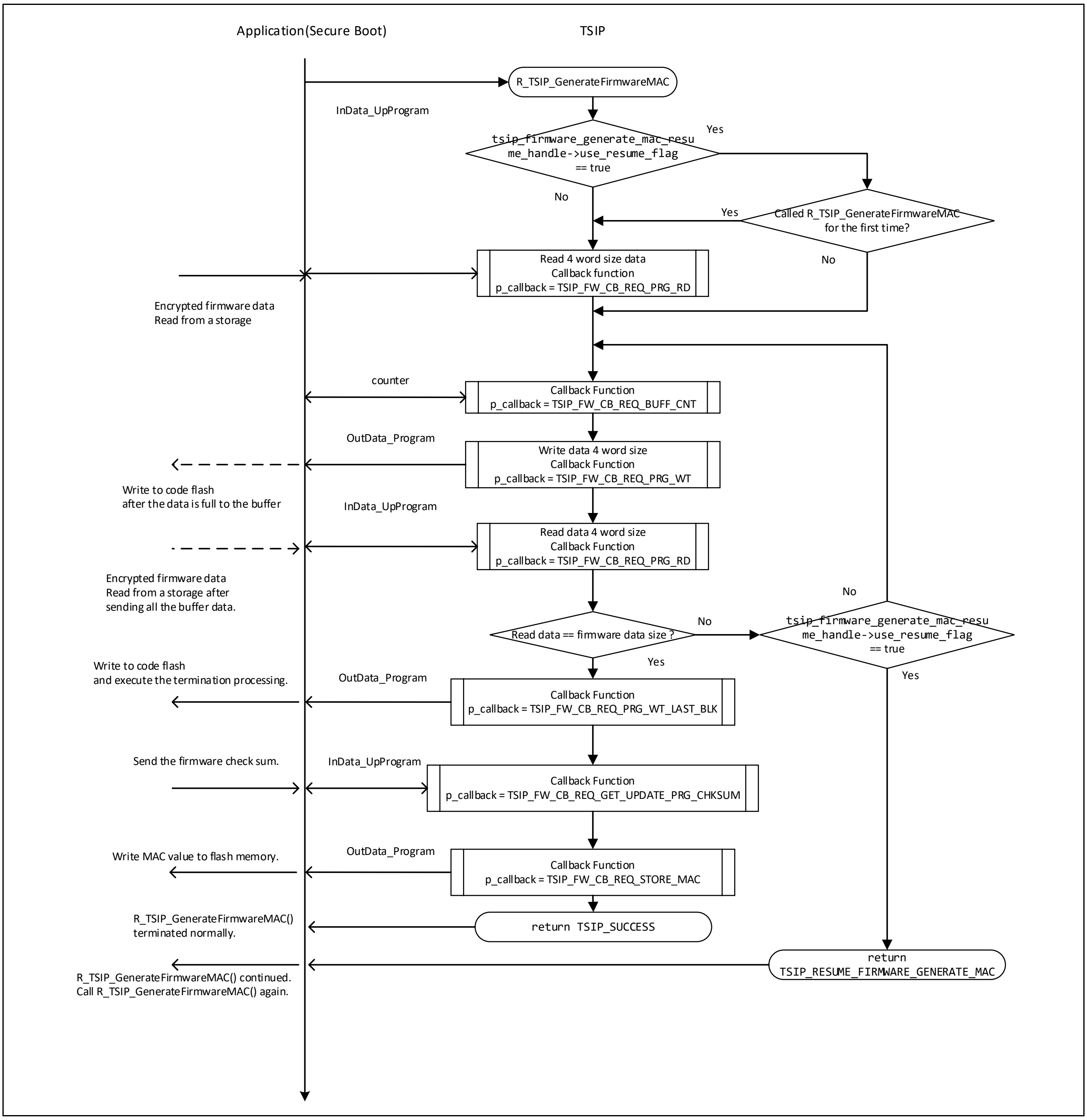

Figure 4-1 Flowchart of Calling of Callback Functions

Processing to read and write firmware data is performed in 4-word units. Therefore, the following procedure is used to call the callback function registered in the seventh argument p_callback. The string in parentheses () is the type of processing specified by the first argument “req_type” of the callback function p_callback.

1. Adjust increment (TSIP_FW_CB_REQ_BUFF_CNT).
2. Write decrypted firmware to storage destination (TSIP_FW_CB_REQ_PRG_WT).
3. Store encrypted firmware in InData_UpProgram (TSIP_FW_CB_REQ_PRG_RD).

It is not necessary to perform the processing in the callback function every time. Perform processing appropriate to the InData_Program and OutData_Program sizes that were reserved.

For example, if a 512-word buffer was reserved, adjust the increment to match the buffer position of the 512 / 4 = 128th time (TSIP_FW_CB_REQ_BUFF_CNT), write to the storage destination (TSIP_FW_CB_REQ_PRG_WT), and store the encrypted firmware in InData_UpProgram (TSIP_FW_CB_REQ_PRG_RD).

For the write request to the final storage destination, specify req_type = TSIP_FW_CB_REQ_PRG_WT_LAST_BLK (not TSIP_FW_CB_REQ_PRG_WT).

This API is called again by the callback function p_callback after reading and writing of the all of the firmware has completed. Check that the 1st argument "req_type"of the callback function p_callback is TSIP_FW_CB_REQ_GET_UPDATE_PRG_CHKSUM, then, pass the checksum value to the 4th argument "InData_UpProgram" of p_callback. This API generates the firmware MAC value after reading the checksum value, when the checksum value is OK. MAC value is passed to the user using the 5th argument "OutData_Program" when the 1st argument "req_type" of callback function p_callback is TSIP_FW_CB_REQ_STORE_MAC. Store the MAC value in the flash area.

If called when tsip_firmware_generate_mac_resume_handle.use_resume_flag is set to true, this API operates as a firmware update start and update function but does not perform firmware update processing in its entirety. If there is additional processing remaining, a value of TSIP_RESUME_FIRMWARE_GENERATE_MAC is returned. Continue to call R_TSIP_GenerateFirmwareMAC() until a value of TSIP_SUCCESS is returned. A return value of TSIP_SUCCESS indicates that firmware update processing has completed successfully.

Reentrant

Not supported

4.2.16.3 R_TSIP_VerifyFirmwareMAC

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_VerifyFirmwareMAC(
        uint32_t *InData_Program,
        uint32_t MAX_CNT,
        uint32_t *InData_MAC
)
```

Parameters

InData_Program	Input	Firmware
MAX_CNT	Input	The word size for firmware+MAC word size. This value should be a multiple of 4. MAC word size is 4 words (16byte). Firmware data minimum size is 16 words, so, MAX_CNT minimum size is 20.
InData_MAC	Input	MAC value to be compared (16byte)

Return Values

TSIP_SUCCESS :	Normal termination
TSIP_ERR_FAIL:	Illegal MAC value
TSIP_ERR_RESOURCE_CONFLICT:	A resource conflict occurred because a hardware resource needed by the processing routine was in use by another processing routine.
TSIP_ERR_PARAMETER	Input data is illegal.

Description

This function verifies the MAC value using firmware. This function will call firm_read_mac() function after all of firmware are read. Pass the MAC value that is generated by R_TSIP_GenerateFirmwareMAC(). For the 3rd argument "InData_Mac", pass the MAC value generated by R_TSIP_GenerateFirmwareMAC().

The MAC verification algorithm uses AES-CMAC.

4.2.16.4 TSIP_GEN_MAC_CB_FUNC_T Type

Format

```
#include "r_tsip_rx_if.h"
typedef void (*TSIP_GEN_MAC_CB_FUNC_T)(
TSIP_FW_CB_REQ_TYPE req_type,
uint32_t iLoop,
uint32_t *counter,
uint32_t *InData_UpProgram,
uint32_t *OutData_Program,
uint32_t MAX_CNT)
```

Parameters

req_type	Input	request contents (TSIP_FW_CB_REQ_TYPE)
iLoop	Input	loop counts (WORD unit)
counter	Input	offset for the area references
InData_UpProgram	Input	same address as the 3rd argument "InData_UpProgram" of R_TSIP_GenerateFirmwareMAC()
OutData_Program	Input /Output	same address as the 5th argument "OutData_Program" of R_TSIP_GenerateFirmwareMAC()
MAX_CNT	Input	same value as the 6th argument "MAX_CNT" of R_TSIP_GenerateFirmwareMAC()

Return Values

none

Description

This function is used in the R_TSIP_GenerateFirmwareMAC and is registered in the 7th argument of this function.

This is used to store the decrypted firmware and MAC at user side.
The area size of InData_UpProgram and OutData_Program should be the multiple of 4, and require at least 4 words. InData_UpProgram and OutData_Program should be the same size. The enclosed sample program is the size of the minimum code flash write unit.

This callback function is called in the R_TSIP_GenerateFirmwareMAC for multiple applications.
The application is stored in the 1st argument "req_type".

The 1st argument ”req_type” has the value defined by the enum TSIP_FW_CB_REQ_TYPE.

```
typedef enum
{
    TSIP_FW_CB_REQ_PRG_WT = 0u,
    TSIP_FW_CB_REQ_PRG_RD,
    TSIP_FW_CB_REQ_BUFF_CNT,
    TSIP_FW_CB_REQ_PRG_WT_LAST_BLK,
    TSIP_FW_CB_REQ_GET_UPDATE_PRG_CHKSUM,
    TSIP_FW_CB_REQ_STORE_MAC,
}TSIP_FW_CB_REQ_TYPE;
```

According to this value, the user takes necessary actions.

<req_type = TSIP_FW_CB_REQ_PRG_WT>

This is the storage request of the decrypted firmware.

TSIP Module makes this request accordingly after storing the data in the 5th argument “OutData_Program” by 4-word unit.

The processing is not required on each request.

Store the decrypted firmware according to the area secured at user side. For example, when the areas are secured for 8 words, store the firmware decrypted when noticed twice.

The sum of the size decrypted is stored in the 2nd argument ”iLoop”.

The maximum value of the ”iLoop” in this request is the value subtracting 4 words from the 6th argument “MAX_CNT”. The last 4 words and the firmware not stored are handled in the request of <req_type = TSIP_FW_CB_REQ_PRG_WT_LAST_BLK>.

<req_type = TSIP_FW_CB_REQ_PRG_RD>

This is the request for obtaining the firmware checksum value for the firmware to be updated.

TSIP Module makes this request accordingly before processing the decryption by 4-word unit.

The system is the same as <req_type = TSIP_FW_CB_REQ_PRG_WT>.

Store the firmware in the 4th argument ”InData_UpProgram” according to the area secured at user side.

<req_type = TSIP_FW_CB_REQ_BUFF_CNT,>

This is the offset value request when referring to the 4th argument ”InData_UpProgram” and the 5th argument ”OutData_Program”.

Return the value with 4-word increment for the 3rd argument “counter” to the 3rd argument ”counter”.

When exceeding the size secured in the 4th argument “InData_UpProgram” and the 5th argument “OutData_Program”, restore the 3rd argument “counter” to its default settings.

<req_type = TSIP_FW_CB_REQ_PRG_WT_LAST_BLK>

This request is made when the last block of the encrypted firmware is decrypted. Store the areas that cannot be stored by the decrypted firmware at this time.

<req_type = TSIP_FW_CB_REQ_GET_UPDATE_PRG_CHKSUM>

This is the request for obtaining the firmware checksum value for the firmware to be updated.

Store the checksum value in the 4th argument “InData_UpProgram”. The checksum is 16byte in length.

<req_type = req_type = TSIP_FW_CB_REQ_STORE_MAC>

The MAC for the decrypted firmware is output.

The MAC (for 16bytes) is stored in the 5th argument “OutData_Program”.

The 6th argument ”MAX_CNT” is the same value as the R_TSIP_GenerateFirmwareMAC()’s.

4.3 User-defined functions

This section describes the user-defined functions called by the TSIP driver.

4.3.1 user_sha384_fucntion

Format

#include “r_tsip_rx_config.h”

uint32_t user_sha384_function (
    uint8_t *message,
    uint8_t *digest,
    uint32_t message_length)

Parameters

message	Input	First address of the message
digest	Output	Hash calculation result storage address (48 bytes)
message_length	Input	Number of valid bytes in message

Return Values

0	Success
Other than 0	Error was occuerd

Description

Since SHA384 is not supported by TSIP in HW, the following API requires the user to create a SHA384 function for signature generation/verification. To use the following API, please enable TSIP_USER_SHA_384_ENABLED in r_tsip_rx_config.h and prepare user_sha384_function function.

- R_TSIP_EcdsaP384SignatureGenerate
- R_TSIP_EcdsaP384SignatureVerification

This function can be defined when the TSIP_USER_SHA_384_ENABLED configuration is enabled.
This function performs a SHA384 hash calculation for the area from the address specified in the argument message to the argument message_length bytes.
The calculation result should be stored in the address specified in the argument digest.

4.3.2 user_lock_fucntion

Format

```
#include "r_tsip_rx_config.h"
void user_lock_function (
        void)
```

Parameters

None

Return Values

None

Description

To use for access management described in 3.2, user must implement function to take the resource for exclusive control. To use the function, implement user_lock_function with enabling TSIP_MULTI_THREADING in r_tsip_rx_config.h.

When the secure boot function is also used with the access management, locate this function to the secure boot area.

4.3.3 user_unlock_fucntion

Format

#include “r_tsip_rx_config.h”

void user_unlock_function (

void)

Parameters

None

Return Values

None

Description

To use for access management described in 3.2, user must implement function to release the resource for exclusive control. To use the function, implement user_lock_function with enabling TSIP_MULTI_THREADING in r_tsip_rx_config.h.

When the secure boot function is also used with the access management, locate this function to the secure boot area.

4.4 Using Renesas Secure Flash Programmer

4.4.1 Generating Encrypted Key Files

Figure 4-2 shows the **Key Wrap** tab in Renesas Secure Flash Programmer.

Set information in "provisioning key File Path" and "encrypted provisioning key File Path" of "provisioning key". Set "provisioning key File Path" to **sample.key** in the FITDemos folder and "encrypted provisioning key File Path" to **sample.key_enc.key**.

Enter setting values then click the **Generate Key Files...** button to generate the Encrypted Key files (**key_data.c** and **key_data.h**).

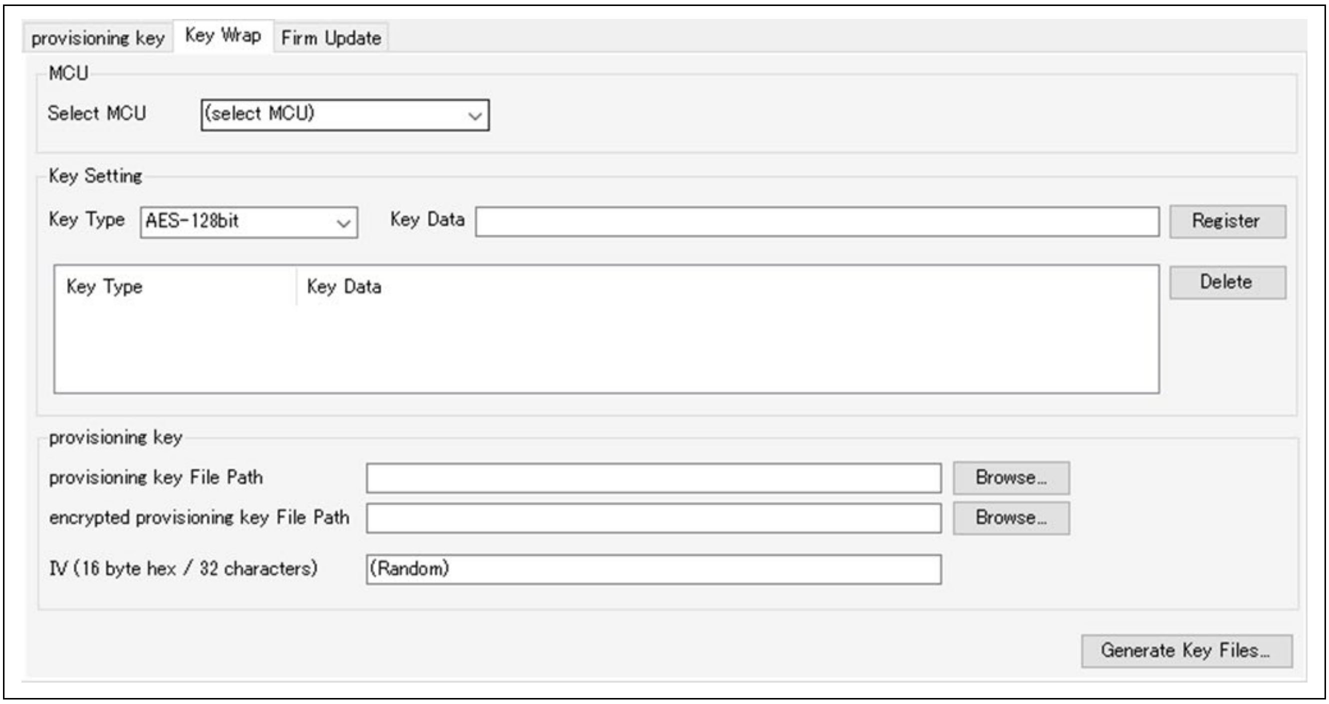

Figure 4-2 Key Wrap Tab in Renesas Secure Flash Programmer

4.4.1.1 KeyData Format

Enter the following data in big-endian order in the Key Data field of the Key Wrap tab.

- AES 128-bit Data Format

Bytes	128-bit
0-15	AES 128 key data

- AES 256-bit Data Format

Bytes	256-bit
0-31	AES 256 key data

- Triple-DES Data Format

Bytes	64-bit	64-bit	64-bit
0-23	DES key data 1	DES key data 2	DES key data 3

- 2-Key TDES Data Format

Bytes	64-bit	64-bit
0-15	DES key data 1	DES key data 2

- DES Data Format

Bytes	64-bit
0-7	DES key data 1

DES key data is 8-bit data with 1 bit of odd parity added to the 7 bits of key data.

The format of DES key data is as follows.

DES Key n								
Byte No	0		1		...	8	
Bit	7-1	0	7-1	0	...	7-1	0
Data	Key Data	Odd Parity	Key Data	Odd Parity	...	Key Data	Odd Parity

For Example:

With parity, DES user key 0x0000000000000000000000 becomes 0x0101010101010101, 0xFFFFFFFFFFFFFFFFFF becomes 0xFEFEFEFEFEFEFEFE, and 0x01020304050607 becomes 0x018080614029190E.

- ARC4 Data Format

Bytes	2048-bit
0-255	ARC4 key data 1

- SHA1-HMAC Data Format

Bytes	160bit
0-19	SHA1-HMAC key data

- SHA256-HMAC Data Format

Bytes	256bit
0-31	SHA256-HMAC key data

- RSA 1024-Bit Public Data Format (132 byte)

Bytes	1024-bit	32-bit
0-131	128-byte RSA Modulus n data	4-byte RSA Exponent e data

- RSA 1024-Bit Private Data Format (256 byte)

Bytes	1024-bit	1024-bit
0-255	128-byte RSA Modulus n data	128-byte RSA Decryption Exponent d data

- RSA 1024-Bit All Data Format (260 byte)

Bytes	1024-bit	32-bit	1024-bit
0-259	128-byte RSA Modulus n data	4-byte RSA Exponent e data	128-byte RSA Decryption Exponent d data

- RSA 2048-bit Public Data Format (260 byte)

Bytes	2048-bit	32-bit
0-259	256-byte RSA Modulus n data	4-byte RSA Exponent e data

- RSA 2048-bit Private Data Format (512 byte)

Bytes	2048-bit	2048-bit
0-511	256-byte RSA Modulus n data	256-byte RSA Decryption Exponent d data

- RSA 2048-Bit All Data Format (516 byte)

Bytes	2048-bit	32-bit	2048-bit
0-515	256-byte RSA Modulus n data	4-byte RSA Exponent e data	256-byte RSA Decryption Exponent d data

- RSA 3072-bit Public Data Format (388 byte)

Bytes	3072-bit	32-bit
0-387	384-byte RSA Modulus n data	4-byte RSA Exponent e data

- RSA 4096-bit Public Data Format (516 byte)

Bytes	4096-bit	32-bit
0-515	512-byte RSA Modulus n data	4-byte RSA Exponent e data

- ECC 192-Bit Public Data Format (48 bytes)

Bytes	192-bit	192-bit
0-47	24-byte ECC public key Qx data	24-byte ECC public key Qy data

- ECC 192-Bit Pravate Data Format (24 bytes)

Bytes	192-bit
0-23	24-byte ECC private key data

- ECC 192-Bit All Data Format (72 bytes)

Bytes	192-bit	192-bit	192-bit
0-71	24-byte ECC public key Qx data	24-byte ECC public key Qy data	24-byte ECC private key data

- ECC 224-Bit Public Data Format (56 bytes)

Bytes	224-bit	224-bit
0-55	28-byte ECC public key Qx data	28-byte ECC public key Qy data

- ECC 224-Bit Private Data Format (28 bytes)

Bytes	224-bit
0-27	28-byte ECC private key data

- ECC 224-Bit All Data Format (84 bytes)

Bytes	224-bit	224-bit	224-bit
0-83	28-byte ECC public key Qx data	28-byte ECC public key Qy data	28-byte ECC private key data

- ECC 256-Bit Public Data Format (64 bytes)

Bytes	256-bit	256-bit
0-63	32-byte ECC public key Qx data	32-byte ECC public key Qy data

- ECC 256-Bit Private Data Format (32 bytes)

Bytes	256-bit
0-31	32-byte ECC private key data

- ECC 256-Bit All Data Format (96 bytes)

Bytes	256-bit	256-bit	256-bit
0-95	32-byte ECC public key Qx data	32-byte ECC public key Qy data	32-byte ECC private key data

- ECC 384-Bit Public Data Format (96 bytes)

Bytes	384-bit	384-bit
0-95	48-byte ECC public key Qx data	48-byte ECC public key Qy data

- ECC 384-Bit Private Data Format (48 bytes)

Bytes	384-bit
0-47	48-byte ECC private key data

- ECC 384-Bit All Data Format (144 bytes)

Bytes	384bit	384bit	384bit
0-143	48-byte ECC public key Qx data	48-byte ECC public key Qy data	48-byte ECC private key data

5. Appendix

5.1 Confirmed Operation Environment

The operation of the driver has been confirmed in the following environment.

Table 5.1 Confirmed Operation Environment

Item	Description
Integrated development environment	Renesas Electronics e^2 studio 2022-10 IAR Embedded Workbench for Renesas RX 4.20.01
C compiler	Renesas Electronics C/C++ Compiler for RX Family (CC-RX) V3.04.00 Compile options: The following option has been added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.3.0.202202 Compile options: The following option has been added to the default settings of the integrated development environment. -std = gnu99
	IAR C/C++ Compiler for Renesas RX version 4.20.01 Compile options: Default settings of the integrated development environment
Renesas Secure Flash Programmer (GUI tool)	The following software is required: Microsoft .NET Framework 4.5 or later
Endian order	Big endian/little endian
Module version	Ver.1.17
Board used	Renesas Starter Kit for RX231 (B version) (product No.: R0K505231S020BE) Renesas Solution Starter Kit for RX23W (with TSIP) (product No.: RTK5523W8BC00001BJ) Renesas Starter Kit+ for RX65N-2MB (with TSIP) (product No.: RTK50565N2S10010BE) Renesas Starter Kit for RX66T (with TSIP) (product No.: RTK50566T0S00010BE) Renesas Starter Kit+ for RX671 (product No.: RTK55671xxxxxxxxx) Renesas Starter Kit+ for RX72M (with TSIP) (product No.: RTK5572MNHSxxxxxxx) Renesas Starter Kit+ for RX72N (with TSIP) (product No.: RTK5572NNHSxxxxxxx) Renesas Starter Kit for RX72T (with TSIP) (product No.: RTK5572TKCS00010BE)

5.2 Troubleshooting

(1) Q: I added the FIT module to my project, but when I build it I get the error "Could not open source file 'platform.h'."

A: The FIT module may not have been added to the project properly. Refer to the documents listed below to confirm if the method for adding FIT modules:

- Using CS+
  Application note: "RX Family: Adding Firmware Integration Technology Modules to CS+ Projects" (R01AN1826)
- Using e^2 studio
  Application note: "RX Family: Adding Firmware Integration Technology Modules to Projects" (R01AN1723)

When using the FIT module, the board support package FIT module (BSP module) must also be added to the project. Refer to the application note "RX Family: Board Support Package Module Using Firmware Integration Technology" (R01AN1685) for instructions for adding the BSP module.

(2) Q: I want to use the FIT Demos e^2 studio sample project on CS+.

A: Visit the following webpage for instructions:
"Porting From the e^2 studio to CS+"
> "Convert an Existing Project to Create a New Project With CS+"
https://www.renesas.com/jp/ja/products/software-tools/tools/migration-tools/migration-e2studio-to-csplus.html

Note: In step 5, the [Q0268002] dialog box may appear if the box next to "Backup the project composition files after conversion" is checked. If you click "Yes" in the [Q0268002] dialog box, you must then re-input the compiler include path.

5.3 User key encryption format

The user key is wrapped using the Provisioning Key and iv when injecting the user key, or using the Update Key Ring and iv when updating the key. The format of the key data to be wrapped at that time depends on the cryptographic algorithm. This chapter shows the data format of the user key to be encrypted (User Key) and the data format of the wrapped key (Encrypted User Key).

Refer to 3.7.1 Key Injection and Update for information on encryption methods.

5.3.1 AES

5.3.1.1 AES 128bit Key

User Key

byte	16			
	4	4	4	4
0-15	128 bit AES key			

Encrypted Key

byte	16			
	4	4	4	4
0-15	encrypted_user_key(128bit AES key)			
16-31	MAC			

5.3.1.2 AES 256bit Key

User Key

byte	16			
	4	4	4	4
0-31	256 bit AES key			

Encrypted Key

byte	16			
	4	4	4	4
0-31	encrypted_user_key(256bit AES key)			
32-47	MAC			

5.3.2 DES

User Key

byte	16			
	4	4	4	4
0-7	56bit DES key with odd parity1 *			
8-15	56bit DES key with odd parity2 *			
16-23	56bit DES key with odd parity3 *			

Encrypted Key

byte	16			
	4	4	4	4
0-23	encrypted_user_key(56bit DES key with odd parity1 || 56bit DES key with odd parity2 || 56bit DES key with odd parity3)			
24-39	MAC			

note: Odd parity should be added to every 7 bits of key data.
For example DES key data = 0x00000000000000 -> 0x0101010101010101
DES key data = 0xFFFFFFFFFFFFFF -> 0xFEFEFEFEFEFEFEFE

For 2-DES, put the same key in 56bit DES key with odd parity1 and 56bit DES key with odd parity3.

For DES, 56bit DES key with odd parity1, 56bit DES key with odd parity2, and 56bit DES key with odd parity3 should all have the same value.

5.3.3 ARC4

User Key

byte	16			
	4	4	4	4
0-255	ARC4			

Encrypted Key

byte	16			
	4	4	4	4
0-255	encrypted_user_key(ARC4)			
256-272	MAC			

5.3.4 RSA

5.3.4.1 RSA 1024bit Key

(1) Public key

User Key

byte	16			
	4	4	4	4
0-127	RSA 1024bit Modulus n			
128-143	RSA 1024bit Exponent e	0 padding		

Encrypted Key

byte	16			
	4	4	4	4
0-143	encrypted_user_key(RSA 1024bit n || e || 0 padding)			
144-159	MAC			

(2) Private Key

User Key

byte	16			
	4	4	4	4
0-127	RSA 1024bit Modulus n			
128-255	RSA 1024bit Decryption Exponent d			

Encrypted Key

byte	16			
	4	4	4	4
0-255	encrypted_user_key(RSA 1024bit n || d)			
256-271	MAC			

5.3.4.2 RSA 2048bit Key

(1) Public key

User Key

byte	16			
	4	4	4	4
0-255	RSA 2048bit Modulus n			
256-271	RSA 2048bit Exponent e	0 padding		

Encrypted Key

byte	16			
	4	4	4	4
0-271	encrypted_user_key(RSA 2048bit n || e || 0 padding)			
272-287	MAC			

(2) Private Key

User Key

byte	16			
	4	4	4	4
0-255	RSA 2048bit Modulus n			
256-511	RSA 2048bit Decryption Exponent d			

Encrypted Key

byte	16			
	4	4	4	4
0-511	encrypted_user_key(RSA 2048bit n || d)			
512-527	MAC			

5.3.4.3 RSA 3072bit Key

(1) Public key

User Key

byte	16			
	4	4	4	4
0-383	RSA 3072bit Modulus n			
384-399	RSA 3072bit Exponent e	0 padding		

Encrypted Key

byte	16			
	4	4	4	4
0-399	encrypted_user_key(RSA 3072bit n || e || 0 padding)			
400-415	MAC			

(2) Private Key

User Key

byte	16			
	4	4	4	4
0-383	RSA 3072bit Modulus n			
384-767	RSA 3072bit Decryption Exponent d			

Encrypted Key

byte	16			
	4	4	4	4
0-511	encrypted_user_key(RSA 3072bit n || d)			
512-527	MAC			

5.3.4.4 RSA 4096bit Key

(1) Public key

User Key

byte	16			
	4	4	4	4
0-511	RSA 4096bit Modulus n			
512-527	RSA 4096bit Exponent e	0 padding		

Encrypted Key

byte	16			
	4	4	4	4
0-527	encrypted_user_key(RSA 4096bit n || e || 0 padding)			
528-543	MAC			

(2) Private Key

User Key

byte	16			
	4	4	4	4
0-511	RSA 4096bit Modulus n			
512-1023	RSA 4096bit Decryption Exponent d			

Encrypted Key

byte	16			
	4	4	4	4
0-1023	encrypted_user_key(RSA 4096bit n || d)			
1024-1039	MAC			

5.3.5 ECC

5.3.5.1 ECC P192bit Key

(1) Public key

User Key

byte	16			
	4	4	4	4
0-31	0 padding			
	ECC P-192 bit Public key Qx			
32-63	0 padding			
	ECC P-192 bit Public key Qy			

Encrypted Key

byte	16			
	4	4	4	4
0-63	encrypted_user_key(0 padding || ECC P-192bit Qx || 0 padding || ECC P-192bit Qy)			
64-79	MAC			

(2) Private Key

User Key

byte	16			
	4	4	4	4
0-31	0 padding			
	ECC P-192 bit Private key d			

Encrypted Key

byte	16			
	4	4	4	4
0-31	encrypted_user_key(0 padding || ECC P-192bit d)			
32-47	MAC			

5.3.5.2 ECC P224bit Key

(1) Public key

User Key

byte	16			
	4	4	4	4
0-31	0 padding			
	ECC P-224 bit Public key Qx			
32-63	0 padding			
	ECC P-224 bit Public key Qy			

Encrypted Key

byte	16			
	4	4	4	4
0-63	encrypted_user_key(0 padding || ECC P-224bit Qx || 0 padding || ECC P-224bit Qy)			
64-79	MAC			

(2) Private Key

User Key

byte	16			
	4	4	4	4
0-31	0 padding			
	ECC P-224 bit Private key d			

Encrypted Key

byte	16			
	4	4	4	4
0-31	encrypted_user_key(0 padding || ECC P-224bit d)			
32-47	MAC			

5.3.5.3 ECC P256 Key

(1) Public key

User Key

byte	16			
	4	4	4	4
0-31	ECC 256 bit Public key Qx			
32-63	ECC 256 bit Public key Qy			

Encrypted Key

byte	16			
	4	4	4	4
0-63	encrypted_user_key(ECC 256bit Qx || Qy)			
64-79	MAC			

(2) Private Key

User Key

byte	16			
	4	4	4	4
0-31	ECC 256 bit Private key d			

Encrypted Key

byte	16			
	4	4	4	4
0-31	encrypted_user_key(ECC P-256bit d)			
32-47	MAC			

5.3.5.4 ECC P384 Key

(1) Public key

User Key

byte	16			
	4	4	4	4
0-47	ECC 384 bit Public key Qx			
48-95	ECC 384 bit Public key Qy			

Encrypted Key

byte	16			
	4	4	4	4
0-95	encrypted_user_key(ECC 384bit Qx || Qy)			
96-111	MAC			

(2) Private Key

User Key

byte	16			
	4	4	4	4
0-47	ECC 384 bit Private key d			

Encrypted Key

byte	16			
	4	4	4	4
0-47	encrypted_user_key(0 padding || ECC 384bit d)			
48-63	MAC			

5.3.6 SHA-HMAC

5.3.6.1 SHA1-HMAC Key

User Key

byte	16			
	4	4	4	4
0-31	HMAC-SHA1 Key			0 padding

Encrypted Key

byte	16			
	4	4	4	4
0-31	encrypted_user_key(HMAC-SHA1 || 0 padding)			
32-47	MAC			

5.3.6.2 SHA256-HMAC Key

User Key

byte	16			
	4	4	4	4
0-31	HMAC-SHA256 Key			

Encrypted Key

byte	16			
	4	4	4	4
0-31	encrypted_user_key(HMAC-SHA256)			
32-47	MAC			

5.3.7 Update Key Ring

User Key

byte	16			
	4	4	4	4
0-15	AES 128bit CBC Key			
16-31	AES 128bit CBCMAC Key			

Encrypted Key

byte	16			
	4	4	4	4
0-31	encrypted_user_key(AES 128bit CBC Key || CBCMAC Key)			
32-47	MAC			

5.4 Asymmetirc key Public Key Index format

Public keys for asymmetric key cryptography contain plaintext information in the Key Index. Therefore, plaintext information can be extracted from the key generation information using TSIP's key generation function. The data format of each cryptographic algorithm is as follows

5.4.1 RSA

The Key Index structure tsip_rsa*XXX*_public_key_index_t of the RSA public key contains the plain text data of the public key in the members value.key_n and value_e.

The Modulus and Exponent values are output in big-endian order in key_n and key_e, respectively.

5.4.2 ECC

The member value.key_q of the Key Index structure tsip_ecc_public_key_index_t of the ECC public key contains the plain text data of the public key. key_q format is as follows.

5.4.2.1 ECC P-192

byte	128 bit			
	32bit	32bit	32bit	32bit
0-15	0 padding		ECC P-192 Public key Qx	
16-31	ECC P-192 Public key Qx(continuation)			
32-47	0 padding		ECC P-192 Public key Qy	
48-63	ECC P-192 Public key (continuation)			
64-79	Key Index management information			

5.4.2.2 ECC P-224

byte	128 bit			
	32bit	32bit	32bit	32bit
0-15	0 padding	ECC P-224 Public key Qx		
16-31	ECC P-224 Public key Qx(continuation)			
32-47	0 padding	ECC P-224 Public key Qy		
48-63	ECC P-224 Public key Qy(continuation)			
64-79	Key Index management information			

5.4.2.3 ECC P-256

byte	128 bit			
	32bit	32bit	32bit	32bit
0-31	ECC P-256 Public key Qx			
32-63	ECC P-256 Public key Qy			
64-79	Key Index management information			

5.4.2.4 ECC P-384

byte	128 bit			
	32bit	32bit	32bit	32bit
0-47	ECC P-384 Public key Qx			
48-95	ECC P-384 Public key Qy			
96-111	Key Index management information			

6. Reference Documents

User's Manual: Hardware
User's Manual: Hardware
(The latest versions can be downloaded from the Renesas Electronics website.)

Technical Update/Technical News
(The latest versions can be downloaded from the Renesas Electronics website.)

User's Manual: Development Environment
RX Family CC-RX Compiler User's Manual (R20UT3248)
(The latest versions can be downloaded from the Renesas Electronics website.)

Website and Support

Renesas Electronics Website
https://www.renesas.com/jp/ja/
Inquiries
https://www.renesas.com/jp/ja/support/contact.html

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jul 10, 2020	-	First release.
1.11	Dec. 31, 2020		• Added DH 2048-bit and ECDHE 512-bit functions • Unified descriptions of iv parameter of R_TSIP_GenerateXXXKeyIndex() and R_TSIP_UpdateXXXKeyIndex() • For R_TSIP_EcdhP256Init(), ECDH(AES GCM128 with IV) is deleted. • Changed R_TSIP_AesXXXKeyWrap() and R_TSIP_AesXXXKeyUnwrap() to common APIs to both TSIP and TSIP-Lite
1.12	Jun. 30, 2021		• Added the sample indicates how to use AES cryptograpy and how to implement TLS
1.13	Aug. 31, 2021		• Added support for RX671
1.14	Oct. 22, 2021		• Added support fot TLS1.3 cooperation function (only RX65N)
1.15	May. 31, 2022		• Added support for TLS1.3 cooperationfunction (for RX66N, RX72M, RX72N) • Added support for TLS1.2 RSA 4096-bit • Added API to get current hash digest value • Removed ref folder.
1.16	Sep. 15. 2022		• Added support for TLS1.3 cooperationfunction (Resumption, 0-RTT) • Added support for AES-CTR • Added support for RSA 3072/4096-bit
1.17	Jan. 20. 2023		• Revised configuration of section in the Application Note • Added support for TLS1.3 server (for RX65N, RX72N)

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.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

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/.