Renesas Microcomputer

RSA Library: User's Manual

Introduction

This document describes a software library (The RSA Library) for Renesas microcomputer.

This manual provides information needed to create application programs using the RSA Library.

In addition to this manual, an "Introduction Guide" is available for each supported microcontroller as documentation of the RSA Library.

This document contains information on program ROM/RAM size and processing performance, as well as specific usage notes, for the supported microcontroller. The Introduction Guide should be referred to alongside this manual.

The software of the RSA Library was created with the specifications contained in the standard listed below as the basis. It should be referred to alongside this manual.

PKCS #1 v2.1: RSA Cryptography Standard

Target Device

Renesas Microcomputer

Contents

1. Overview

1.1 Overview of RSA

Please refer to the following site for the generic information about RSA.

"http://en.wikipedia.org/wiki/RSA_%28cryptosystem%29"

1.2 Visualization RSA Signature Generation and Verification

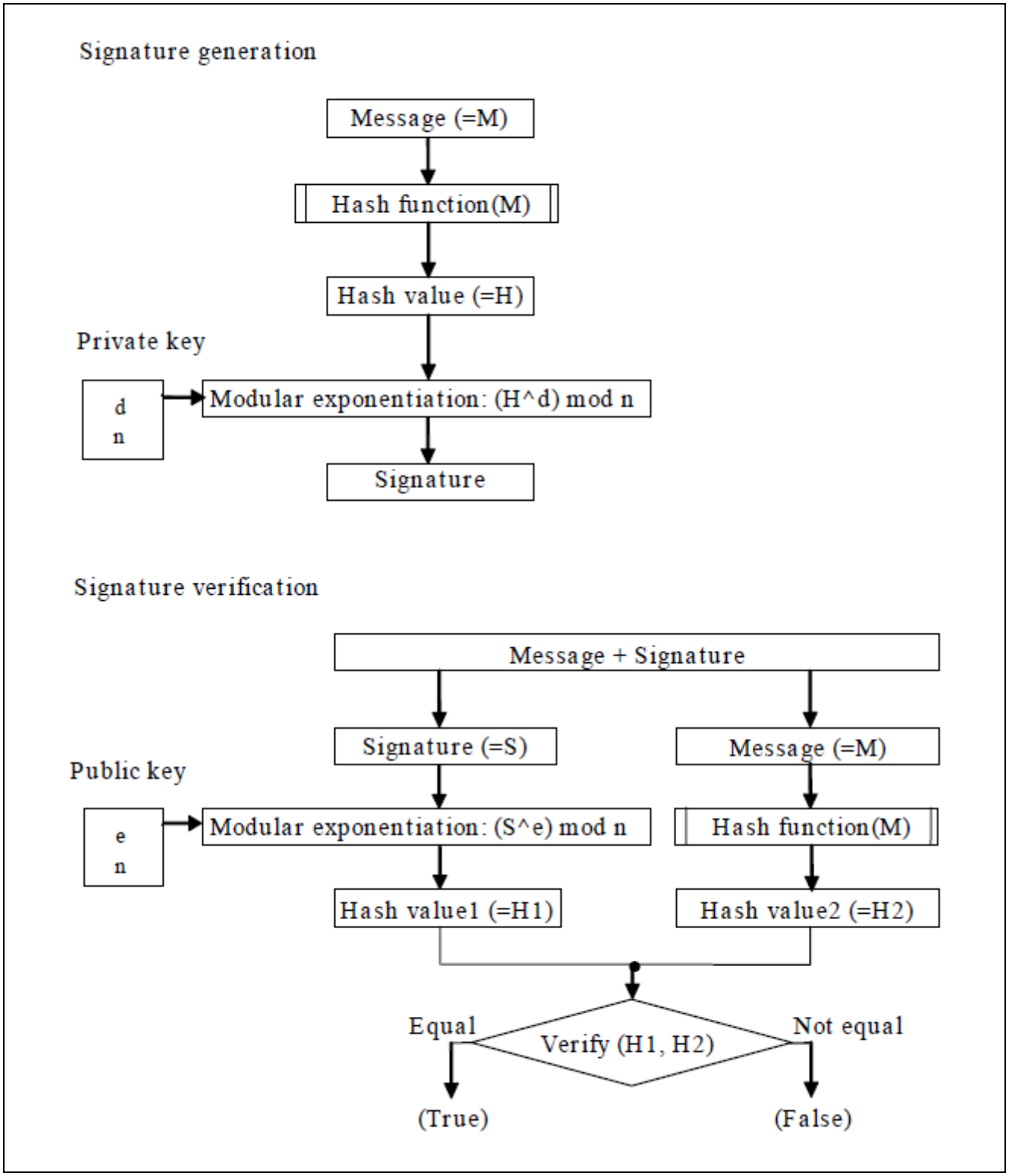

Figure 1-1 Visualization RSA Signature Generation and Verification

2. Overview of the RSA Library

2.1 Software Stack Structure

The software stack structure of the RSA Library is shown below.

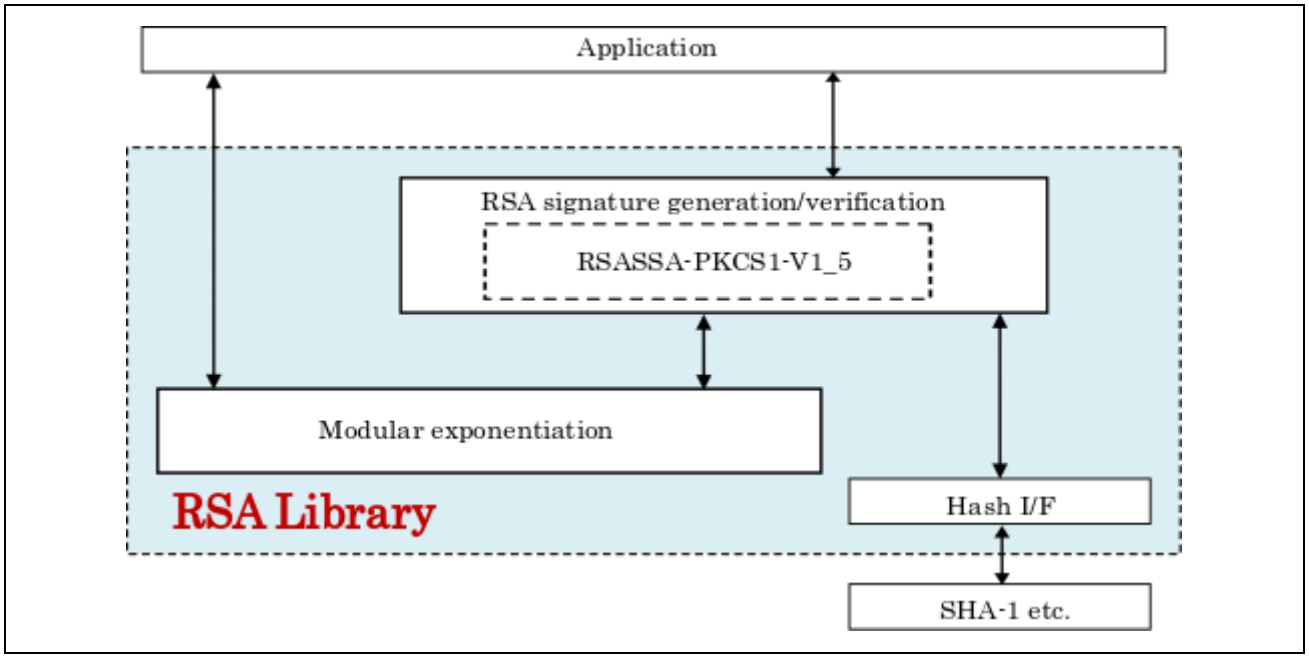

Figure 2-1 The RSA Library Structure

Usage Example

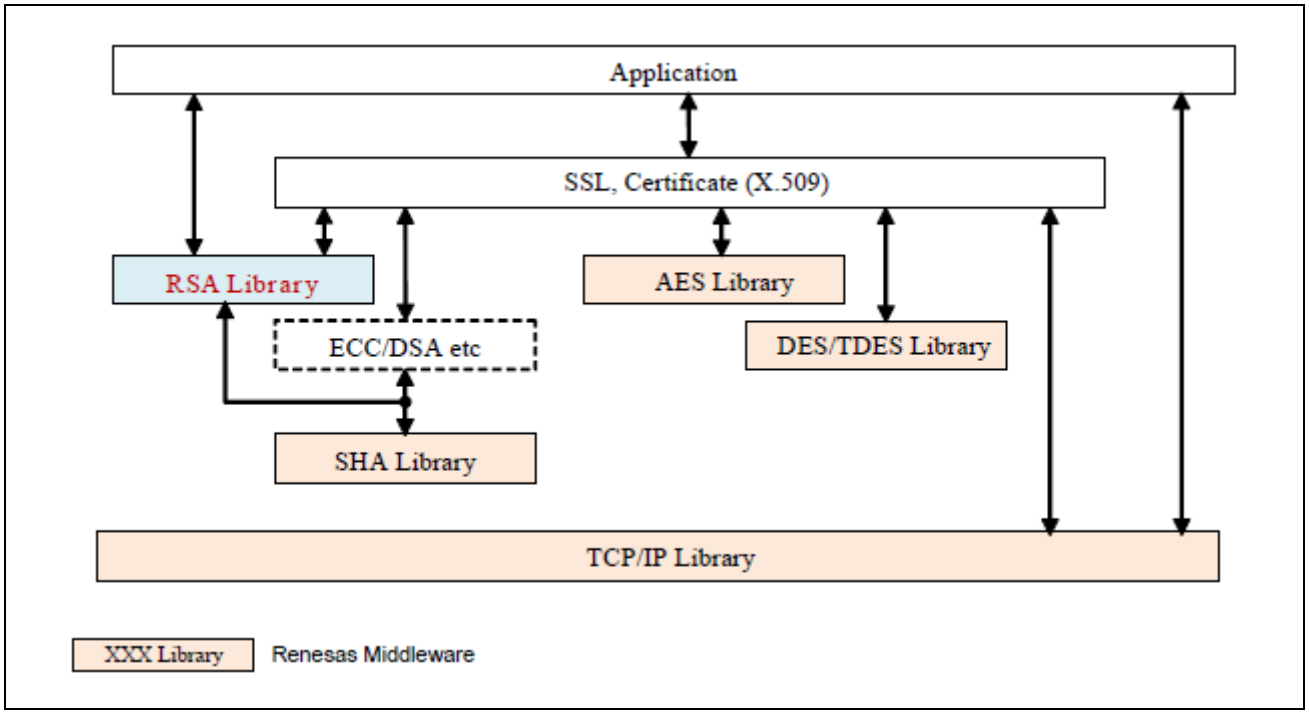

Figure 2-2 Example of Application Using the RSA Library

2.2 Specification of the RSA Library

Following are some of the main specifications of the RSA Library.

Table 2-1. Specifications of the RSA Library

Item	Specification
Signature generation scheme	RSASSA-PKCS1-V1_5 *1
Signature verification scheme	RSASSA-PKCS1-V1_5 *1
Modular exponentiation	Supported
Key length	Max. 2,048 bits
Key generation function	None

Caution *1. The user must create the hash function.

2.3 RSA Key Data

The RSA Library does not include functionality for generating RSA keys. Software for this purpose must be prepared by the user ahead of time.

Also, the key data must already be present in the ROM or RAM before an RSA library function is run.

The key data is not cleared after the RSA Library function finishes, so the user must make sure the data is cleared if necessary.

2.4 Handling of Memory Used for Calculations

When performing calculations, the RSA Library temporarily stores intermediate calculation data in a work area. Since it might be possible to guess the key information from the intermediate calculation data, each RSA library function clears the work area used before the function completes.

2.5 Distribution Format

Table 2-2. The RSA Library Distribution Contents

Type	File name	Description
Library	r_rsa.h r_mw_version.h r_stdint.h rsa_api.c mc_lib.c mc_lib.h rsa_internal_header.h	The software source code for providing RSA functionality.
Sample program	Project file for each supported microcontroller. Please refer to Introduction Guide.	Project file that can be used to verify the operation of an application in Renesas integrated development environment. Also, associated file directories. Software source code for testing basic functions such as RSA signature generation/verification, and Modular exponentiation.

2.6 Program Development Procedure

The RSA Library is distributed in library format. To use it, link it to the application program.

Shows an Figure 2-3,“Application Program Development Flowchart”.

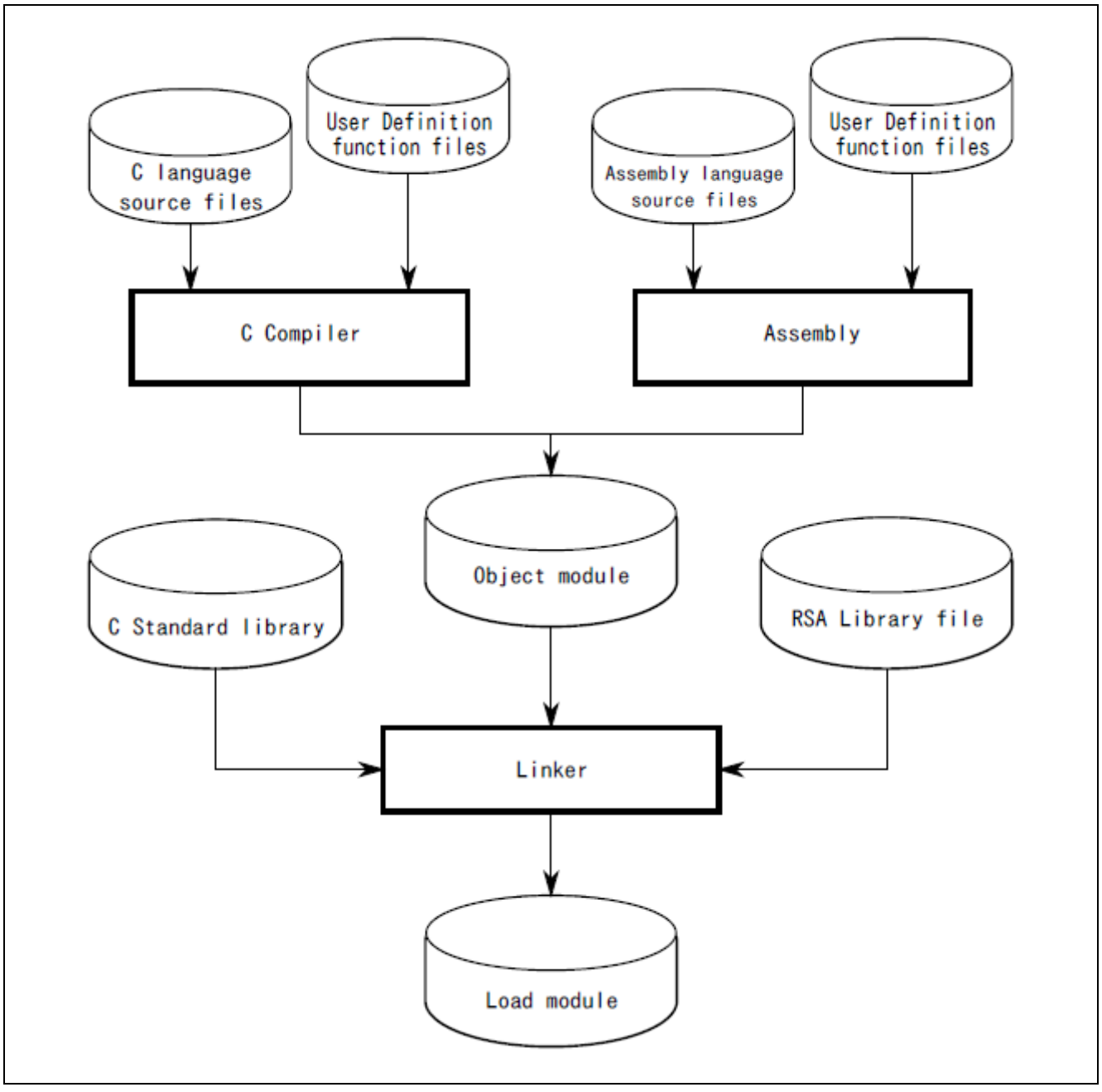

Figure 2-3 Application Program Development Flowchart

3. Library type definitions

This section describes the definitions of the data types used by the RSA Library.

3.1 Library type definitions

The RSA Library uses types defined in the header file "r_stdint.h".

Table 3-1. Basic Data Types

Data Type	Typedef	Number of bytes
signed char	int8_t	1
signed short	int16_t	2
signed long	int32_t	4
unsigned char	uint8_t	1
unsigned short	uint16_t	2
unsigned long	uint32_t	4

4. Library Structures

This section describes the data structures used by the RSA library.

4.1 R_RSA_WORK_t - Work Memory Structure

The R_RSA_WORK_t structure defines the work area used by the RSA library for calculations. The application must allocate this structure to an area in RAM. When parallel instances of the RSA library run at the same time, multiple structure variables should be provided to pass each work area to the API.

Table 4.1 Members of R_RSA_WORK_t

Data Type	Member Name	Explanation
uint32_t	work[R_RSA_WORK_SIZE]	RSA work area Needs 3680 bytes

4.2 R_RSA_BYTEDATA_t - Byte String Structure

The R_RSA_BYTEDATA_t structure defines the input/output information for calling RSA library APIs. The application must allocate this structure to an area in RAM.

Table 4.2 Structure members of R_RSA_BYTEDATA_t

Data Type	Structure element	Explanation
uint8_t *	p_adr	Data start address
uint16_t	len	Valid size (bytes) starting from p_adr

4.3 R_RSA_KEY_t - RSA Key Information Structure

R_RSA_KEY_t structure is key information referred by RSA Library. R_RSA_KEY_t structure includes the modulus and exponentiation used as key. This modulus and exponentiation are defined as R_RSA_BYTEDAT_t structure that includes data pointer and data length.

This data pointer points the data that front of 1 byte is not 0x00. When the data that front of 1 byte is 0x00, it is not normally processed in each API. "RSA_PARAM_ERR" is returned to a return value.

User defined this structure as variable. User specifies each value in this structure. Please specify the front of data address to the p_adr using modulus or exponentiation. Please specify the valid byte number of modulus or exponentiation to the "len". Hereafter, this document explains about this "len" as "Valid size of modulus-n". Or, "Valid size of exponentiation".

Table 4.3 Structure members of R_RSA_KEY_t

Data Type	Structure element	Explanation
R_RSA_BYTEDATA_t	key_n	Modulus(n of public key(e, n), or n of private key (d, n))
R_RSA_BYTEDATA_t	key_ed	Exponent(e of public key(e, n), or d of private key (d, n))

If the public key e value (e, n) is 65537 (0x10001), the setting is as follows (n content omitted).

Setting Example

```
const uint8_t e[3] = {0x01,0x00,0x01};
const uint8_t n[256];
R_RSA_KEY_t rsa_key =
{
    n,      /* key_n.p_adr */
    256,    /* key_n.len */
    e,      /* key_ed.p_adr */
    3,      /* key_ed.len */
}
```

5. Macro Definitions

This section describes the macro definitions used by the RSA Library. The following macros are defined in r_rsa.h.

5.1 Return Value

The RSA Library APIs return the value shown in the following table data.

Table 5.1 Return Value Macro

Macro Name	Value	Significance
RSA_OK	0	Normal end
RSA_PARAM_ERR	-1	Parameter error end
RSA_USER_DEF_FUNC_ERR	-2	User definition function error end
RSA_MOD_EXP_NG	-3	Calculation failed by Modular exponentiation API
RSA_ENC_NG	-4	(Unused)
RSA_DEC_NG	-5	(Unused)
RSA_SIG_GEN_NG	-6	Signature generation failed by signature generation API
RSA_SIG_VERIFY_NG	-7	Signature verification failed by signature verification API

5.2 Macros for Defining Hash Type Specified to Signature Generation/Verification APIs

The following macros define the hash type used for signature generation and verification.

The RSA library calls an interface function from the signature generation and verification APIs, with the following macro values as arguments.

The hash function must be provided by the user.

Table 5.2 Hash Definition Macros

Macro Name	Value	Meaning
RSA_HASH_MD2	0x00	MD2
RSA_HASH_MD5	0x01	MD5
RSA_HASH_SHA1	0x02	SHA-1
RSA_HASH_SHA256	0x03	SHA-256
RSA_HASH_SHA384	0x04	SHA-384
RSA_HASH_SHA512	0x05	SHA-512

6. Library functions

This section describes the APIs of the RSA Library.

Details of each API are described by the following format.

< Format >

Shows a format in which the function is called. The header file indicated in #include "header file" is the standard header file necessary to execute the function described here. Always be sure to include it.

< Argument >

The letters I and O respectively mean that the parameter is input data or output data. If marked by IO, it means input/output data.

< Return Value >

Shows the value returned by the function. The explanation about a return value is written by Description.

< Description >

Describes specifications of the function.

< Notes >

Shows the precautions when use the function.

< Using Example >

Shows the usage example of the function.

< Making Example >

Shows an example of the function create.

6.1 R_rsa_signature_generate_pkcs

Signature Generation (RSASSA-PKCS1-V1_5)

< Format >

#include "r_rsa.h"

int16_t R_rsa_signature_generate_pkcs (
R_RSA_BYTEDATA_t *p_mes,
R_RSA_BYTEDATA_t *p_sig,
R_RSA_KEY_t *p_key,
uint8_t hash_type,
R_RSA_WORK_t *p_wk);

< Argument >

Argument	I/O	Explanation
p_mes	I	The message information to be signed
p_sig	I/O	Signature text storage destination information
p_key	I	Key information
hash_type	I	Hash method
p_wk	I/O	Work area

< Return Value >

Return value	Explanation
RSA_OK	Normal end
RSA_PARAM_ERR	Parameter error end
RSA_USER_DEF_FUNC_ERR	User definition function error end
RSA_SIG_GEN_NG	Signature generation failed

< Description >

This API generates a signature according to RSASSA-PKCS1-V1_5.

The lower limit of modulus-n valid byte size is different according to the argument "hash_type". User needs to prepare the bigger Key data than the following table.

Please specify the bigger data more than 0 byte into p_mes. If message is 0 byte data size, please specify excluding 0-address into p_adr.

Please specify the bigger data more than modulus-n valid data size into the signature text storage.

Please set these areas into RAM area.

- the area pointed p_sig

- the area pointed p_sig->p_adr

- the area pointed p_wk

Please implement the user definition function R_rsa_if_hash().

Invalid address (=address 0) is specified, this API returns parameter error.

In case API returns RSA_OK, result of signature will be output into p_sig->p_adr, and output byte number will be output into p_sig->len.

Output byte number will be same to valid size of modulus-n.

Table 6.1 The lower limit of modulus-n valid byte size

The value of the argument "hash_type"	The lower limit of modulus-n valid byte size
RSA_HASH_MD2	45
RSA_HASH_MD5	45
RSA_HASH_SHA1	46
RSA_HASH_SHA256	62
RSA_HASH_SHA384	78
RSA_HASH_SHA512	94

< Using Example >

```
int16_t         ret;
uint8_t         mes_buff[5] = {'a','b','c','d','e'};
uint8_t         sig_buff[256];
R_RSA_WORK_t    rsa_work;
R_RSA_BYTEDATA_t  sig_text;
R_RSA_BYTEDATA_t  message;
extern R_RSA_KEY_t   rsa_private_key;

message.p_adr = mes_buff;
message.len = 5;
sig_text.p_adr = sig_buff;
sig_text.len = 256;

ret = R_rsa_signature_generate_pkcs(
    &message,
    &sig_text,
    &rsa_private_key,
    RSA_HASH_SHA256,
    &rsa_work);
if(RSA_OK == ret)
{
    /* Signature generation was successful. */
}
```

6.2 R_rsa_signature_verify_pkcs

Signature Verification (RSASSA-PKCS1-V1_5)

< Format >

```
#include "r_rsa.h"

int16_t R_rsa_signature_verify_pkcs (
                    R_RSA_BYTEDATA_t *p_sig,
                    R_RSA_BYTEDATA_t *p_mes,
                    R_RSA_KEY_t *p_key,
                    uint8_t hash_type,
                    R_RSA_WORK_t *p_wk);
```

< Argument >

Argument	I/O	Explanation
p_sig	I	The signature information to be verified
p_msg	I	The message information to be verified
p_key	I	Key information
hash_type	I	Hash method
p_wk	I/O	Work area

< Return Value >

Return value	Explanation
RSA_OK	Normal end
RSA_PARAM_ERR	Parameter error end
RSA_USER_DEF_FUNC_ERR	User definition function error end
RSA_SIG_VERIFY_NG	Verification failed

< Description >

This API verifies the signature according to RSASSA-PKCS1-V1_5.

The lower limit of modulus-n valid byte size is different according to the argument "hash_type". User needs to prepare the bigger Key data than table in the R_rsa_signature_generate_pkcs() secion.

Please specify the same size (modulus-n valid data size) into the signature information to be verified.

Please set the bigger data more than 0 byte into the message information to be verified.

Please set this area into RAM area.

- the area pointed p_wk

Please implement the user definition function R_rsa_if_hash().

Invalid address (=address 0) is specified, this API returns parameter error.

< Using Example >

```
int16_t         ret;
extern uint8_t    mes_buff[5];
extern uint8_t    sig_buff[256];
R_RSA_WORK_t    rsa_work;
R_RSA_BYTEDATA_t  sig_text;
R_RSA_BYTEDATA_t  message;
extern R_RSA_KEY_t    rsa_public_key;

message.p_adr = mes_buff;
message.len = 5;
sig_text.p_adr = sig_buff;
sig_text.len = 256;

ret = R_rsa_signature_verify_pkcs(
    &sig_text,
    &message,
    &rsa_public_key,
    RSA_HASH_SHA256,
    &rsa_work);
if(RSA_OK == ret)
{
    /* Signature Verify was successful. */
}
```

6.3 R_rsa_mod_exp

Modular exponentiation

< Format >

```
#include "r_rsa.h"

int16_t R_rsa_mod_exp (
                        R_RSA_BYTEDATA_t *p_input,
                        R_RSA_BYTEDATA_t *p_output,
                        R_RSA_KEY_t *p_key,
                        R_RSA_WORK_t *p_wk);
```

< Argument >

Argument	I/O	Explanation
p_input	I	Calculation information
p_output	I/O	Calculated data storage destination information
p_key	I	Exponent of modulus exponentiation (e or d) and modulus (n) information
p_wk	I/O	Work area

< Return Value >

Return value	Explanation
RSA_OK	Normal end
RSA_PARAM_ERR	Parameter error end
RSA_MOD_EXP_NG	Calculation failed

< Description >

This API performs Modular exponentiation.

p_output = (p_input ^ p_key->key_e) MOD p_key->key_n

Please specify the (1~modulus valid data size) data into input area.

Please specify the bigger data more than modulus-n valid data size into output area.

Please set these areas into RAM area.

- the area pointed p_output
- the area pointed p_output->p_adr
- the area pointed p_wk

Invalid address (=address 0) is specified, this API returns parameter error.

In case API returns RSA_OK, result of the data will be output into p_output->p_adr, and output byte number will be output into p_output->len.

Output byte number will be same to valid size of modulus-n.

This API is called automatically in API (R_rsa_signature_generate_pkcs, R_rsa_signature_verify_pkcs) which does signature generation/signature verification. Therefore, the user does not have to call this API when executing these two APIs.

< Using Example >

```
int16_t         ret;
uint8_t         inbuff[5] = {'a','b','c','d','e'};
extern uint8_t    outbuff[256];
R_RSA_WORK_t    rsa_work;
R_RSA_BYTEDATA_t  output;
R_RSA_BYTEDATA_t  input;
extern R_RSA_KEY_t    rsa_key;

input.p_adr = inbuff;
input.len = 5;
output.p_adr = outbuff;
output.len = 256;

ret = R_rsa_mod_exp(
    &input,
    &output,
    &rsa_key,
    &rsa_work);
if(RSA_OK == ret)
{
    /* Successful. */
}
```

7. User Definition Functions

This section describes the user definition functions called by the RSA Library.

Prototypes of these functions and the processing that must be implemented by each function are described. The user must create program code to implement each of these functions.

7.1 R_rsa_if_hash

Calculate Hash

< Format >

```
#include "r_rsa.h"

int16_t R_rsa_if_hash (
                        uint8_t * p_mes ,
                        uint8_t * p_hash ,
                        uint16_t mes_len ,
                        uint8_t hash_type );
```

< Argument >

Argument	I/O	Explanation
p_mes	I	Start address of message
p_hash	O	Hash calculation result storage address
mes_len	I	Valid byte count of message
hash_type	I	Hash type

< Return Value >

Return value	Explanation
0	Hash calculation successful
Other than 0	Hash calculation failed

< Description >

This function must be implemented in order to run R_rsa_signature_generate() and R_rsa_signature_verify().

The hash is calculated for the number of bytes of data specified by the argument mes_len, starting from the address specified by the argument p_mes.

The hash method is specified by the argument hash.

The calculation result should be stored at the address specified by the argument p_hash.

The calculation result and the size of the stored calculation result differ depending on the hash method.

Table 7.1 Size (Bytes) Stored at Address Specified by p_hash

The value of the argument "hash_type"	Size (Bytes) Stored at Address Specified by p_hash
RSA_HASH_MD2	16
RSA_HASH_MD5	16
RSA_HASH_SHA1	20
RSA_HASH_SHA256	32
RSA_HASH_SHA384	48
RSA_HASH_SHA512	64

< Making Example >

Using Renesas SHA Library

```
#include "r_sha.h"

int16_t R_rsa_if_hash(
    uint8_t *p_mes,
    uint8_t *p_hash,
    uint16_t mes_len,
    uint8_t hash_type)
{
    R_sha_handle sha_work;
    int8_t hash_ret; /* SHA Library return type */
    int16_t ret;

    ret = -1;
    switch(hash_type)
    {
        case RSA_HASH_SHA1:
            /* SHA-1 calculation */
            hash_ret = R_Sha1_HashDigest( p_mes,
                p_hash,
                mes_len,
                (R_SHA_INIT | R_SHA_FINISH),
                (void*)&sha_work);
            if(R_PROCESS_COMPLETE == hash_ret)
            {
                ret = 0;
            }
            break;
        case RSA_HASH_SHA256:
            /* SHA-256 calculation */
            hash_ret = R_Sha256_HashDigest( p_mes,
                p_hash,
                mes_len,
                (R_SHA_INIT | R_SHA_FINISH),
                (void *)&sha_work);
            if(R_PROCESS_COMPLETE == hash_ret)
            {
                ret = 0;
            }
            break;
        case RSA_HASH_MD2:
            /* MD2 calculation */
            break;
        case RSA_HASH_MD5:
            /* MD5 calculation */
            break;
        case RSA_HASH_SHA384:
            /* SHA-384 calculation */
#if !(defined(__RL78__) || defined(__ICCRL78__) || defined(__CCRL__))
            hash_ret = R_Sha384_HashDigest( p_mes,
                p_hash,
                mes_len,
                (R_SHA_INIT | R_SHA_FINISH),
```

```
                (void *)&sha_work);
            if(R_PROCESS_COMPLETE == hash_ret)
            {
                ret = 0;
            }
#endif
            break;
        case RSA_HASH_SHA512:
            /* SHA-512 calculation */
            break;
        default:
            break;
    }
    return ret;
}
```

Note) In the attached sample code r_sample_rsa_if.c, you can select how to generate Hash Digest with the following options.

Definition	Description
#define USE_SHA_LIB (0) * The default value is "0: Hash Digest (fixed value)".	Select how to generate Hash Digest. 0: Use Hash Digest (fixed value) 1: Generate Hash Digest with SHA library

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar 31, 2014	−	First Edition issued
1.01	Sep 01, 2016	−	Address lists are revised. Revision History is added.
2.00	Apr 23, 2021	−	RSAES-PKCS1-V1_5 is no longer supported. As a result, the function specifications of R_rsa_encrypt_pkcs, R_rsa_decrypt_pkcs, and R_rsa_if_rand have been deleted.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.