

Xilinx Standalone Library Documentation

XilRSA Library v1.4

UG1190 (2017.1) October 4, 2017

Table of Contents

Chapter 1: Overview

Source Files	3
Usage of SHA-256 Functions	3
SHA2 API Example Usage	3

Chapter 2: XilRSA APIs

Overview	4
Function Documentation	4
rsa2048_exp	4
rsa2048_pubexp	5
sha_256	5
sha2_starts	5
sha2_update	6
sha2_finish	6

Appendix A: Additional Resources and Legal Notices

Overview

The XilRSA library provides APIs to use RSA encryption and decryption algorithms and SHA algorithms for Zynq®-7000 All Programmable SoC devices.

Note

The RSA-2048 bit is used for RSA and the SHA-256 bit is used for hash.

For an example on usage of this library, refer to the RSA Authentication application and its documentation.

Source Files

The following is a list of source files shipped as a part of the XilRSA library:

- `librsa.a`: Pre-compiled file which contains the implementation.
- `xilrsa.h`: This file contains the APIs for SHA2 and RSA-20148..

Usage of SHA-256 Functions

When all the data is available on which sha2 must be calculated, the [sha_256\(\)](#) function can be used with appropriate parameters, as described. When all the data is not available on which sha2 must be calculated, use the sha2 functions in the following order:

1. [sha2_update\(\)](#) can be called multiple times till input data is completed.
2. `sha2_context` is updated by the library only; do not change the values of the context.

SHA2 API Example Usage

```
sha2_context ctx;  
sha2_starts(&ctx);  
sha2_update(&ctx, (unsigned char *)in, size);  
sha2_finish(&ctx, out);
```

Following is the source code of the sha2_context class.

```
typedef struct  
{  
    unsigned int state[8];  
    unsigned char buffer[SHA_BLKBYTES];  
    unsigned long long bytes;  
} sha2_context;
```

XiIRSA APIs

Overview

This section provides detailed descriptions of the XiIRSA library APIs.

Functions

- void [rsa2048_exp](#) (const unsigned char *base, const unsigned char *modular, const unsigned char *modular_ext, const unsigned char *exponent, unsigned char *result)
- void [rsa2048_pubexp](#) (RSA_NUMBER a, RSA_NUMBER x, unsigned long e, RSA_NUMBER m, RSA_NUMBER rrm)
- void [sha_256](#) (const unsigned char *in, const unsigned int size, unsigned char *out)
- void [sha2_starts](#) (sha2_context *ctx)
- void [sha2_update](#) (sha2_context *ctx, unsigned char *input, unsigned int ilen)
- void [sha2_finish](#) (sha2_context *ctx, unsigned char *output)

Function Documentation

void [rsa2048_exp](#) (const unsigned char * *base*, const unsigned char * *modular*, const unsigned char * *modular_ext*, const unsigned char * *exponent*, unsigned char * *result*)

This function is used to encrypt the data using 2048 bit private key.

Parameters

<i>modular</i>	A char pointer which contains the key modulus
<i>modular_ext</i>	A char pointer which contains the key modulus extension
<i>exponent</i>	A char pointer which contains the private key exponent
<i>result</i>	A char pointer which contains the encrypted data

Returns

None

void rsa2048_pubexp (RSA_NUMBER *a*, RSA_NUMBER *x*, unsigned long *e*, RSA_NUMBER *m*, RSA_NUMBER *rrm*)

This function is used to decrypt the data using 2048 bit public key.

Parameters

<i>a</i>	RSA_NUMBER containing the decrypted data.
<i>x</i>	RSA_NUMBER containing the input data
<i>e</i>	Unsigned number containing the public key exponent
<i>m</i>	RSA_NUMBER containing the public key modulus
<i>rrm</i>	RSA_NUMBER containing the public key modulus extension.

Returns

None

void sha_256 (const unsigned char * *in*, const unsigned int *size*, unsigned char * *out*)

This function calculates the hash for the input data using SHA-256 algorithm. This function internally calls the sha2_init, updates and finishes functions and updates the result.

Parameters

<i>In</i>	Char pointer which contains the input data.
<i>Size</i>	Length of the input data
<i>Out</i>	Pointer to location where resulting hash will be written.

Returns

None

void sha2_starts (sha2_context * *ctx*)

This function initializes the SHA2 context.

Parameters

<i>ctx</i>	Pointer to sha2_context structure that stores status and buffer.
------------	--

Returns

None

void sha2_update (sha2_context * *ctx*, unsigned char * *input*, unsigned int *ilen*)

This function adds the input data to SHA256 calculation.

Parameters

<i>ctx</i>	Pointer to sha2_context structure that stores status and buffer.
<i>input</i>	Pointer to the data to add.
<i>Out</i>	Length of the input data.

Returns

None

void sha2_finish (sha2_context * *ctx*, unsigned char * *output*)

This function finishes the SHA calculation.

Parameters

<i>ctx</i>	Pointer to sha2_context structure that stores status and buffer.
<i>output</i>	Pointer to the calculated hash data.

Returns

None

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see [Xilinx Support](#).

Solution Centers

See the [Xilinx Solution Centers](#) for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>.



Automotive Applications Disclaimer

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

© Copyright 2017 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.