Xilinx Standalone Library Documentation

XilRSA Library v1.6

UG1190 (v2022.1) April 20, 2022

Xilinx is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing non-inclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards. Follow this link for more information.

Table of Contents

Chapter 1

Overview

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

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

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

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..

Chapter 2

Usage of the 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;
```

Chapter 3

XilRSA APIs

This section provides detailed descriptions of the XilRSA library APIs.

Table 1: **Quick Function Reference**

Type	Name	Arguments
void	rsa2048_exp	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

Functions

rsa2048_exp

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

Prototype

```
void rsa2048_exp(const unsigned char *base, const unsigned char *modular,
const unsigned char *modular_ext, const unsigned char *exponent, unsigned
char *result);
```

Parameters

The following table lists the `rsa2048_exp` function arguments.

Table 2: **rsa2048_exp Arguments**

Type	Name	Description
const unsigned char *	modular	A char pointer which contains the key modulus
const unsigned char *	modular_ext	A char pointer which contains the key modulus extension
const unsigned char *	exponent	A char pointer which contains the private key exponent
unsigned char *	result	A char pointer which contains the encrypted data

Returns

None

rsa2048_pubexp

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

Prototype

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

Parameters

The following table lists the `rsa2048_pubexp` function arguments.

Table 3: **rsa2048_pubexp Arguments**

Type	Name	Description
RSA_NUMBER	a	RSA_NUMBER containing the decrypted data.
RSA_NUMBER	x	RSA_NUMBER containing the input data
unsigned long	e	Unsigned number containing the public key exponent
RSA_NUMBER	m	RSA_NUMBER containing the public key modulus
RSA_NUMBER	rrm	RSA_NUMBER containing the public key modulus extension.

Returns

None

sha_256

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.

Prototype

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

Parameters

The following table lists the `sha_256` function arguments.

Table 4: **sha_256 Arguments**

Type	Name	Description
const unsigned char *	in	Char pointer which contains the input data.
const unsigned int	size	Length of the input data
unsigned char *	out	Pointer to location where resulting hash will be written.

Returns

None

sha2_starts

This function initializes the SHA2 context.

Prototype

```
void sha2_starts(sha2_context *ctx);
```

Parameters

The following table lists the `sha2_starts` function arguments.

Table 5: **sha2_starts Arguments**

Type	Name	Description
sha2_context *	ctx	Pointer to sha2_context structure that stores status and buffer.

Returns

None

sha2_update

This function adds the input data to SHA256 calculation.

Prototype

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

Parameters

The following table lists the `sha2_update` function arguments.

Table 6: **sha2_update Arguments**

Type	Name	Description
sha2_context *	ctx	Pointer to sha2_context structure that stores status and buffer.
unsigned char *	input	Pointer to the data to add.
unsigned int	ilen	Length of the input data.

Returns

None

sha2_finish

This function finishes the SHA calculation.

Prototype

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

Parameters

The following table lists the `sha2_finish` function arguments.

Table 7: **sha2_finish Arguments**

Type	Name	Description
sha2_context *	ctx	Pointer to sha2_context structure that stores status and buffer.
unsigned char *	output	Pointer to the calculated hash data.

Returns

None

Appendix A

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the Vivado® IDE, select **Help → Documentation and Tutorials**.
- On Windows, select **Start → All Programs → Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

***Note*:** For more information on DocNav, see the Documentation Navigator page on the Xilinx website.

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 https://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 https://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