Summary

The LibXilSecure library provides APIs to access secure hardware on the Zynq® UltraScale+™ MPSoC devices.

This library includes:

- SHA-3 engine hash functions
- AES for symmetric key encryption
- RSA for authentication

These are grouped into the Configuration and Security Unit (CSU) on the Zynq UltraScale+ MPSoC device.

XilSecure APIs

The following list is a list of functions for Zynq® UltraScale+™ MPSoC devices which are grouped by their respective function You can click on a link to go directly to the function section.

Source Files

- `xsecure_hw.h`

`s32 XSecure_Sha3Initialize(XSecure_Sha3 *InstancePtr, XCsuDma* CsuDmaPtr)`

Description	This API initializes a specific Xsecure_Sha3 instance so that it is ready to be used.
Parameters	instancePtr is a pointer to the XSecure_Sha3 instance.
	CsuDmaPtr is the pointer to the XCsuDma instance.
Returns	XST_SUCCESS if initialization was successful.

`void XSecure_Sha3Start(XSecure_Sha3 *InstancePtr)`

Description	This API configures the secure stream switch (SSS) and starts the SHA-3 engine.
Parameters	`InstancePtr` is a pointer to the `XSecure_Sha3` instance.
Returns	None

`void XSecure_Sha3Update(XSecure_Sha3 *InstancePtr, const u8 *Data, const u32 Size)`

Description	This API updates hash for new input data block. This is used after `XSecure_Sha3Start` to update more input data.
	InstancePtr is a pointer to the XSecure_Sha3 instance.
Parameters	Data is the pointer to the input data for hashing
	Size of the input data in bytes
Returns	None

`void XSecure_Sha3Finish(XSecure_Sha3 *InstancePtr, u8 *Hash)`

Description	This API sends the last block; for example, the padding when block size is not multiple of 104 bytes. The final hash is ready at this stage.
Parameters	InstancePtr is a pointer to the XSecure_Sha3 instance.
	Hash is the pointer to location where resulting hash will be written
Returns	None

void **XSecure_Sha3Digest**(XSecure_Sha3 *InstancePtr, const u8 *In, const u32 Size u8 *Out)

Description	This API calculates SHA-3 Digest on the given input data. In effect, this does the complete operation that can be achieved by calling `XSecure_Sha3Start`, `XSecure_Sha3Update`, and `XSecure_Sha3Finish` in succession.
Parameters	`InstancePtr` is a pointer to the `XSecure_Sha3` instance. `In` is the pointer to the input data for hashing size of the input data in bytes. `Out` is the pointer to location where resulting hash is written.
Returns	None

Example Usage

`XSecure_Sha3Example.c` : This example is a simple application using SHA-3 device to calculate 384 bit hash on Hello World string.

A more typical use case of SHA-3 has been illustrated in RSA example XSecure_RsaExample.c where it is used to calculate hash of boot image as a step in authentication process.

RSA

This block decrypts data based on RSA-4096 algorithm. A utility function to compare the signature with expected signature (Verification) is also provided.

API Summary

The following is a summary list of APIs provided for using RSA module.. Descriptions of the APIs follow the list.

s32 XSecure_RsaInitialize(XSecure_Rsa *InstancePtr, u8 *Mod, u8 *ModExt, u8 *ModExpo)

s32 XSecure_RsaDecrypt(XSecure_Rsa *InstancePtr, u8 *EncText, u8 *Result)

u32 XSecure_RsaSignVerification(u8 *Signature, u8 *Hash, u32 HashLen)

s32 **XSecure_RsaInitialize**(**XSecure_Rsa** *InstancePtr, u8 *Mod, u8 *ModExt, u8 *ModExpo)

Description	This API initializes a specific `Xsecure_Rsa` instance so that it is ready to be used.
Parameters	`InstancePtr` is a pointer to the `XSecure_Rsa` instance. `Mod` is the pointer to Modulus used for authentication. `ModExt` is the pointer to precalculated R^2 Mod N value used for authentication. `ModExpo` is the pointer to the exponent (public key) used for authentication.
Returns	`XST_SUCCESS` if decryption was successful.

`s32` **`XSecure_RsaDecrypt`**`(XSecure_Rsa *InstancePtr, u8 *EncText, u8 *Result)`

Description	This API handles the RSA decryption from end-to-end.
Parameters	`InstancePtr` is a pointer to the `XSecure_Rsa` instance. `Result` is the pointer to decrypted data generated by RSA. `EncText` is the pointer to the data (hash) to be decrypted.
Returns	`XST_SUCCESS` if decryption was successful.

`u32` **`XSecure_RsaSignVerification`**`(u8 *Signature, u8 *Hash, u32 HashLen)`

Description	This API matches the decrypted data with expected data.
Parameters	`InstancePtr` is a pointer to the XSecure_Rsa instance. `Signature` is the pointer to RSA signature for data to be authenticated. `Hash` is the pointer to expected hash data. `HashLen` is the length of Hash used.
Returns	`XST_SUCCESS` if decryption was successful.

Example Usage

`XSecure_RsaExample.c` : This example deals with RSA based authentication of FSBL in a Zynq MPSoC boot image. The boot image signature is decrypted using RSA- 4096 algorithm. Resulting digest is matched with SHA digest calculated on the FSBL using SHA-3 driver.

The authenticated boot image should be loaded in memory through JTAG and address of the boot image should be passed to the function. By default, the example assumes that the authenticated image is present at location `0x04000000` (DDR), which can be changed as required.

AES Functions

This block can encrypt/decrypt data using AES-GCM algorithm. Decryption using keyrolling is also supported. The key, IV and format of encrypted data should be strictly in accordance with the Bootgen specified format.

Format of Encrypted Image

The format of the encrypted data should be the same as the one used by Bootgen for encrypting Zynq UtlraScale + MPSoC device boot images. The bootgen encrypted images have a secure header at the beginning followed by any number of blocks.

API Summary

The following is a summary list of APIs provided for using AES module. Descriptions of the APIs follow the list.

32 XSecure_AesInitialize(XSecure_Aes *InstancePtr, XCsuDma *CsuDmaPtr,u32 KeySel, u32* Iv, u32* Key)

u32 XSecure_AesDecrypt(XSecure_Aes *InstancePtr, u8 *Dst, const u8 *Src, *32 Length)

void XSecure_AesEncrypt(XSecure_Aes *InstancePtr, u8 *Dst, const u8 *Src 32 Len)

32 **XSecure_AesInitialize**(XSecure_Aes *InstancePtr, XCsuDma *CsuDmaPtr,u32 KeySel, u32* Iv, u32* Key)

Description	This API initializes the instance pointer.
Parameters	`InstancePtr` is a pointer to the `XSecure_Aes` instance. `CsuDmaPtr` is the pointer to the `XCsuDma instance`. `KeySel` is the key source for decryption, can be KUP (user- provided) or device key. `Iv` is pointer to the Initialization Vector for decryption. `Key` is the pointer to Aes decryption key in case KUP key is used. Passes `Null` if device key is to be used.
Returns	`XST_SUCCESS` upon success.

u32 **XSecure_AesDecrypt**(XSecure_Aes *InstancePtr, u8 *Dst, const u8 *Src, *32 Length)

Description	This function handles the AES-GCM decryption.
Parameters	`InstancePtr` is a pointer to the `XSecure_Aes` instance. `Src` is the pointer to encrypted data source location `Dst` is the pointer to location where decrypted data will be written. `Length` is the expected total length of decrypted image/data.
Returns	`XST_SUCCESS` if encryption passed and GCM tag matched. `XSECURE_CSU_AES_IMAGE_LEN_MISMATCH` if Image length did not match. `XST_FAILURE` in case of failure.

void **XSecure_AesEncrypt**(XSecure_Aes *InstancePtr, u8 *Dst, const u8 *Src 32 Len)

Description	This API encrypts input data using encryption engine.
Parameters	`InstancePtr` is a pointer to the XSecure_Aes instance.
Returns	None

void **XSecure_AesReset**(XSecure_Aes *InstancePtr)

Description	This API resets the AES engine.
Parameters	`InstancePtr` is a pointer to the `XSecure_Aes` instance.
Returns	None

Example Usage

`XSecure_AesExample.c:` This examples illustrates AES usage with decryption of a Zynq UltraScale + MP SoC boot image placed at a predefined location in memory. User can select the key type (device key or user-selected KUP key). The example assumes that the boot image is present at `0x0400000` (DDR); consequently, the image must be loaded at that address through JTAG. The example decrypts the boot image and returns `XST_SUCCESS` or `XST_FAILURE` based on whether the GCM tag was successfully matched.