Socionext Secure Module

FIPS 140-2 Non-Proprietary Security Policy

Version: 1.2

Table of Contents

1. Introduction

This is a non-proprietary Cryptographic Module Security Policy for the Socionext Secure Module with hardware version 0x00000001 and firmware version 0x00010004. This Security Policy describes how the module meets the security requirements of Federal Information Processing Standards (FIPS) Publication 140-2, which details the U.S. and Canadian government requirements for cryptographic modules. More information about the FIPS 140-2 standard and validation program is available on the National Institute of Standards and Technology (NIST) and the Canadian Centre for Cyber Security (CCCS) Cryptographic Module Validation Program (CMVP) website at https://csrc.nist.gov/Projects/Cryptographic-Module-Validation-Program.

This policy was prepared as part of the Level 1 FIPS 140-2 validation of the module. The Socionext Secure Module is referred to as the module in this document.

1.1. Purpose

There are three major reasons that a security policy is required

• It is required for FIPS 140-2 validation.

• It allows individuals and organizations to determine whether the implemented module satisfies the stated security policy.

• It allows individuals and organizations to determine whether the described capabilities, the level of protection, and access rights provided by the cryptographic module meet their security requirements.

1.2. Target Audience

This document is part of the package of documents that are submitted for FIPS 140-2 conformance validation of the module. It is intended for the following people:

- Developers working on the release
- FIPS 140-2 testing lab
- Cryptographic Module Validation Program (CMVP)

1.3. Additional References

[FIPS 140-2]	Security Requirements for Cryptographic Modules, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf, 2001
[FIPS 140-2 IG]	Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/documents/fips140-2/fips1402ig.pdf, 2018
[SP800-90A Rev.1]	Recommendation for Random Number Generation Using Deterministic Random Bit Generators, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf, 2015
[SP 800-90B]	Recommendation for the Entropy Sources Used for Random Bit Generation, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf, 2018

[SP 800-38A] Recommendation for Block Cipher Modes of Operation: Methods and Techniques, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf, 2001

[SP 800-38B] Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38b.pdf, 2016

[SP 800-38D] Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf, 2007

[SP 800-38F] Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf, 2012

[SP 800-56A Rev. 3] Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf, 2018

[SP 800-56C Rev. 1] Recommendation for Key-Derivation Methods in Key-Establishment Schemes, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr1.pdf, 2018

[SP 800-108] Recommendation for Key Derivation Using Pseudorandom Functions (Revised), https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-108.pdf, 2009

[FIPS 198-1] The Keyed-Hash Message Authentication Code (HMAC), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.198-1.pdf, 2008

[FIPS 197] Advanced Encryption Standard (AES), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf, 2001

[FIPS 180-4] Secure Hash Standard (SHS), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf, 2015

2. Module Specification

2.1. Module Description

The module is a hardware cryptographic module implemented as a sub-chip system running on a single-chip standalone processor and is classified as a sub-chip cryptographic subsystem contained within a single chip embodiment for the purpose of FIPS 140-2 validation.

2.2. Module Validation Level

The module is intended to meet requirements of FIPS 140-2 at an overall Security Level 1. The following table shows the security level claimed for each of the eleven sections that comprise the validation:

Table 2-1 Security Levels

FIPS140-2 Security Requirement Area	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services and Authentication	3
Finite State Model	1
Physical Security	1
Operational Environment	N/A
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	2
Mitigation of Other Attacks	N/A

2.3. Cryptographic Module Boundary

The module was tested as a sub-chip cryptographic subsystem implemented within Zynq® UltraScale+™ XCZU9EG-2FFVB1156E MPSoC (referred to as the FPGA in this document), which is mounted on ZCU102, a general purpose evaluation board provided by Xilinx Inc.

The following figure shows the block diagram of the FPGA and the module.

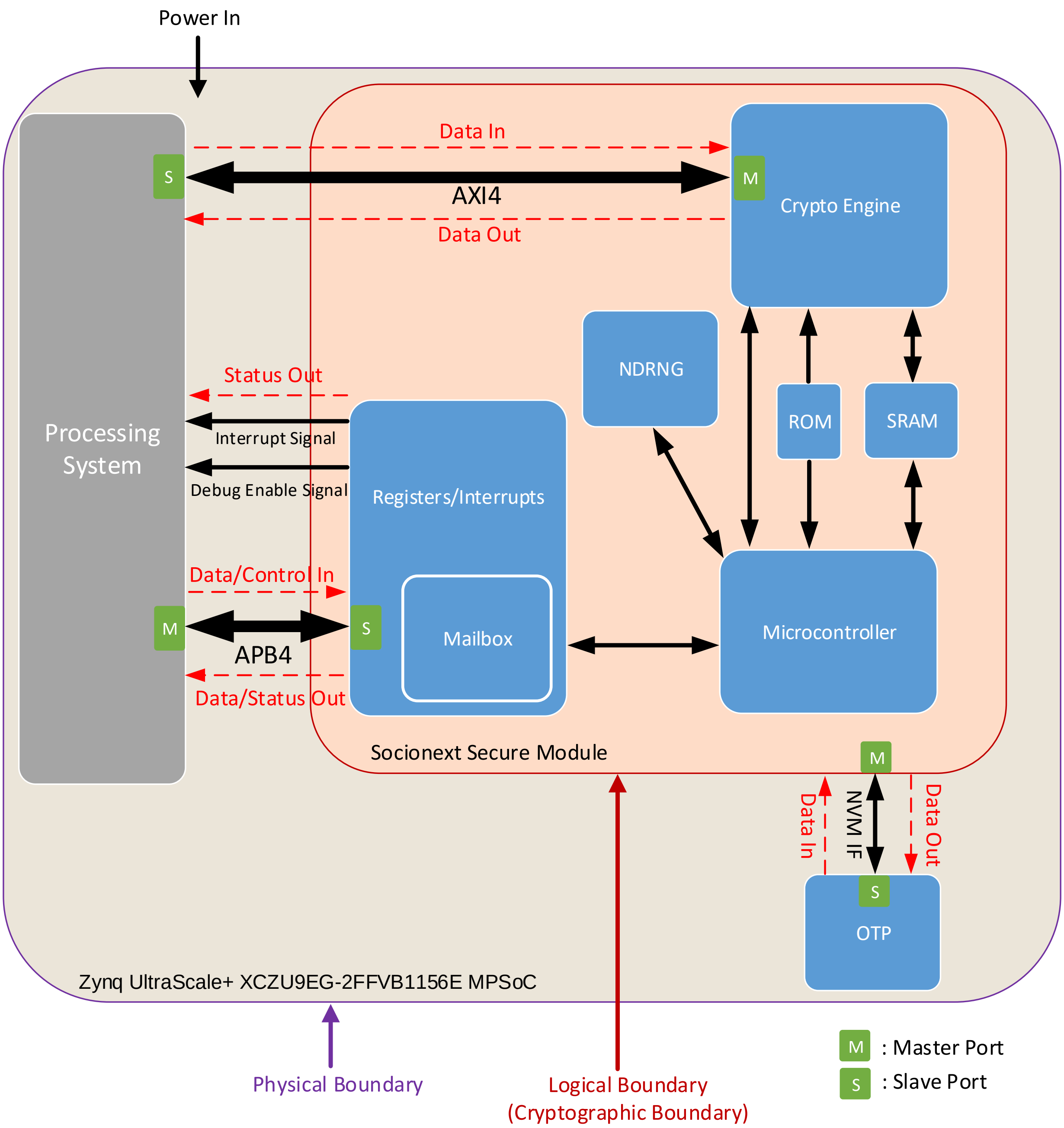

Figure 2-1 Socionext Secure Module Block Diagram

The physical boundary of the module is the physical boundary of the FPGA. Consequently, the embodiment of the module is a single-chip cryptographic module. The logical boundary (Cryptographic

Boundary) of the module is the Socionext Secure Module. The module is classified as a single-chip hardware module for the purpose of FIPS 140-2 validation. SRAM in the Figure 2-1 is referred to as "the SRAM" and OTP in the Figure 2-1 is referred to as "the OTP" in the rest of this document.

2.4. Approved, Allowed or Vendor Affirmed Security Functions

The following table shows the approved or allowed security functions used in the module. No FIPS non-approved security functions or vendor affirmed security functions are implemented by the module.

Table 2-2 Approved or Allowed Security Functions

Cryptographic Function	Algorithm	Relevant standard	Certificate Number
Symmetric Encryption and Decryption	AES Encryption/Decryption in ECB/CBC/CTR modes with 128/192/256 key	FIPS 197 SP 800-38A	C571
	Triple-DES Encryption/Decryption in CBC mode	SP 800-67 SP 800-38A	C571
Symmetric Encryption and Decryption with Authentication	AES-GCM Encryption/Decryption with 128/192/256 key	FIPS 197 SP 800-38D FIPS 140-2 IG A.5	C572
Message Digest	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 SHA-512/224, SHA-512/256	FIPS180-4	C571
Message Authentication Code	HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, HMAC-SHA-512/224, HMAC-SHA-512/256	FIPS 198-1	C571
	AES-CMAC Generation with 128/192/256 key	FIPS 197 SP 800-38B	C571
Digital Signature	RSA PKCS#1_v1.5 Signature Generation/Signature Verification 2048-3072, RSAPSS Signature Generation/Signature Verification 2048-3072 ECDSA Signature Generation/Signature Verification P-224/P-256/P-384/P-521	FIPS 186-4	C571
Key Derivation	KDF in Counter Mode using HMAC-SHA-256 Note: The location of 32-bits counter is before the fixed input data.	SP800-108	C571
Key Agreement	ECC CDH (Elliptic Curve Cryptography Cofactor Diffie-Hellman) Primitive P-224/P-256/P-384/P-521	SP800-56A Revised	C571
Key Pair Generation	ECDSA P-224/P-256/P-384/P-521	FIPS 186-4	C571

Public Key Verification	ECDSA P-224/P-256/P-384/P-521	FIPS 186-4	C571
Key Transport (Key Wrapping)	AES-GCM Encryption/Decryption with 256-bits key[*1]	FIPS 197 SP 800-38D FIPS 140-2 IG A.5 FIPS 140-2 IG D.9	C571[*2]
DRBG	HASH_DRBG	SP800-90A	C571
NDRNG			Allowed
Key Generation	CKG	SP800-133	Vendor Affirmed

*1: The module supports 2 modes for AES-GCM Encryption IV listed in the table below. Here, AES-GCM IV is 96-bits. AES-GCM Decryption IV should be always supplied from outside of the module.
*2: AES-GCM Encryption/Decryption generating IV internally and using 128/192/256-bits key lengths is validated under CAVP (Cert. #C571), but only 256-bits key length is supported by the module. That is, when the IV is generated internally, the AES-GCM using 128/192-bits key lengths cannot be used by an operator of the module.

Table 2-3

mode	description
RBG-based Construction	If Key Transport function is used, the module generates 96-bits AES-GCM Encryption IV which is random bits generated by the approved Hash_DRBG. The random number from the NDRBG which the module holds is used as the entropy input of Hash-DRBG.
Deterministic Construction	If Symmetric Encryption and Decryption with Authentication function is used, AES-GCM Encryption IV is generated and provided from inside of the single chip but outside of the module. The IV length is 96-bits. This IV is 96-bits and required to be generated by the FIPS approved GCM IV generation.

2.5. Modes of Operation

The module only supports FIPS-approved mode of operation and only supports FIPS-approved and allowed security functions. No other modes of operation and security functions are implemented by the module. Therefore, when the module is powered up and successfully completes the power up self-teset, the module enters the FIPS-approved mode of operation.
Loading a firmware version that is not listed in Section 2.1 will cause the module to be running in a Non-FIPS validated state.

3. Ports and Interfaces

The module supports ports and interfaces listed in the table below.

Table 3-1 Ports and Interfaces

FIPS Interface	Ports
Data Input	Mailbox over APB / DMA IF over AXI / NVM IF
Data Output	Mailbox over APB / DMA IF over AXI / NVM IF
Control Input	Mailbox and Registers over APB
Status Output	Registers over APB / Interrupt / Debug Enable

The "Mailbox over APB" means access to the Mailbox via APB4 shown in Figure 2-1. The mailbox consists of SRAM and is used for command/data input and data output.
The "DMA IF over AXI" means access from the Crypto Engine via AXI4 in Figure 2-1.
Basically, the module is controlled by command input via the mailbox but interrupt related control is done by registers.
Debug Enable port is used to show the status that Debug Enable command, which described later, has been successful.

4. Roles, Authentication and Services

4.1. Roles and Authentication

The module supports two roles: a Crypto Officer and an User. The Crypto Officer is basically for module setup and initialization but it can use all services except Debug Enable service.

There are two kinds of Users: an User and a Debug User.

The User is for general cryptographic services. The Debug User is allowed to use Debug Enable service only. These Users are identified by User ID in a command.

The Crypto Officer and the Users are authenticated with the right ID and Password.

It is possible for the Crypto Officer to prohibit the User's access to its CSPs. The Debug User can access only CSPs necessary for Debug Enable service.

Table 4-1 lists roles supported by the module along with their description and authentication method.

Table 4-1 Roles

Role		Role Description	Authentication Type	Authentication Method
Crypto Officer (CO)		All services are allowed except Debug Enable service.	Identity-based authentication	256-bits ECDSA signature verification (only for Authentication CO service) / 32-bits Password comparison
User	User	General cryptographic services are allowed. Compared to CO, module initialization related services are not allowed.	Identity-based authentication	32-bits Password comparison
	Debug User (DU)	Only Debug Enable service is available.		

Strengths of authentication mechanism are listed in the table below.
A 32-bits password is a 32-bits bit-string used to authenticate an operator. 256-bits ECDSA signature is used to authenticate Crypto Officer Role by Authentication CO service.

Table 4-2 Strengths of Authentication

Authentication Data	Strength of Authentication
256-bits ECDSA signature	256-bits ECDSA signature has 128-bits of security. The probability of signature that a single random authentication attempt will succeed or a false acceptance will occur is $1/2^{128}$ which is less than 1/1,000,000. When the attempt fails, the module waits at least one second, during which any attempt is ignored. Thus, the maximum authentication rate is 60 per minute and the probability that random authentication attempts will succeed within a one-minute interval is $60/2^{128}$ which is less than 1/100,000.
32-bits Password	A 32-bits password is a 32-bits bit-string user to authenticate an operator. The probability of a 32-bits bit-string password that a single random authentication attempt (by guessing the bit-string password value) will succeed or a false acceptance will occur is $1/2^{32}$ which is less than 1/1,000,000. When the attempt fails, the module waits at least one second, during which any attempt is ignored. Thus, the maximum authentication rate is 60 per minute and the probability that random authentication attempts will succeed within a one-minute interval is $60/2^{32}$ which is less than 1/100,000.

4.2. Services

The module supports services following table shows. All services require authentication except for Self-Test, Status Check and Version Check.

The following table lists services implemented by the module along with their description.

Table 4-3 Authenticated Services

	Service	Roles			Description
		CO	User *1		
			U	DU	
1	Symmetric Encryption/Decryption	X	X		This service encrypts or decrypts supplied data using AES-ECB, AES-CBC, AES-CTR or Triple DES Key.
2	Symmetric Encryption/Decryption with Authentication	X	X		This service encrypts or decrypts supplied data with authentication using AES-GCM Key.
3	Hash	X	X		This service generates a message digest using Message Digest function.
4	MAC	X	X		This service generates Message Authentication Code on a supplied data using Keyed Hash function or AES-CMAC Key.
5	Random Number Generation	X	X		This service generates a random number using the DRBG generate function. A random number, which is unmodified output by this service, can be used as a Key Derive Key.
6	RSA-PKCS #1 v1.5 Sign	X	X		This service generates a RSA-PKCS#1 v1.5 signature on a supplied data.
7	RSA-PKCS #1 v1.5 Verify	X	X		This service verifies a RSA-PKCS#1 v1.5 signature on a supplied data with a supplied RSA Public Key.
8	RSA-PSS Sign	X	X		This service generates a RSA-PSS signature on a supplied data.
9	RSA-PSS Verify	X	X		This service verifies a RSA-PSS signature on a supplied data with a supplied RSA Public Key.
10	ECDSA Sign	X	X		This service generates an ECDSA signature on a supplied data.
11	ECDSA Verify	X	X		This service verifies an ECDSA signature on a supplied data with supplied ECDSA Public Keys.
12	ECC CDH Key Agreement	X	X		This service generates Shared Secret with other party's ECDH Public Key.
13	ECC Multiplication	X	X		This service computes ECDSA/ECDH Public Key with ECDSA/ECDH Private Key, which is imported or

					generated by a corresponding service.
14	ECC Public-Key Verify	X	X		This service verifies supplied ECDSA/ECDH Public Key by checking if the following conditions hold. 1. $y_Q^2=x_Q^3+ax_Q+b$ (mod p) 2. $nQ=O$ Here, $Q=(x_Q,y_Q)$ is the ECDSA/ECDH Public Key.
15	Key Derivation	X	X		This Service generates an AES-ECB Key, an AES-CBC Key, an AES-CTR Key, an AES-GCM Key, a Triple DES Key, a HMAC Key, an AES-CMAC Key, a Key Derive Key, a Key Wrap Key, a pair of ECDSA Private and Public Keys or a pair of ECDH Private and Public Keys.
16	Import Key	X	X		This service stores a key supplied from outside of the module into the SRAM inside the module. This service can also be used to copy a Key Wrap Key or a Key Derive Key in the OTP into the SRAM. If a supplied key is encrypted by key-wrap algorithm, which is AES-GCM, the supplied key is stored into the SRAM after key-unwrapping.
17	Export Key	X	X		This service encrypts a CSP stored in the SRAM with key-wrap algorithm, which is AES-GCM, and outputs the encrypted CSP. If a CSP is Shared Secret generated by ECC CDH Key Agreement service, this service outputs Shared Secret in plaintext by 2-step procedure.
18	Delete Key	X	X		This service zeroizes CSPs in the SRAM.
19	Clear OTP	X			This service zeroizes CSPs in the OTP.
20	Delete All Keys	X			This service zeroizes All CSPs in the SRAM and the OTP.
21	Random Number Generator Configuration	X			This service gives configuration parameters of NDRNG and does the DRBG instantiate function.
22	Self-Test	No authentication required			This service performs Self-Tests described in Chapter 9. This service is invoked automatically at power-up of the module. This service is not provided as command via the mailbox.
23	Initialization	X			This service is used to write CSPs and PSPs necessary for operation into the OTP in plaintext. This service is never invoked more than one time.
24	Authentication CO	X			This service authenticates Crypto Officer by 2-step

					procedure. If authentication succeeds, Crypto Officer Password is returned in plaintext via the mailbox.
25	Firmware Load	X			This service loads second firmware of the module using AES-GCM. This service is optional and performed during Authentication CO service.
26	Define User	X			This service enters User ID and User Password into the SRAM in plaintext.
27	Monotonic counter increment	X			This service increments the monotonic counter in the OTP. The monotonic counter is stored in plaintext. The monotonic counter is implemented to support a firmware rollback prevention. The processing system can detect a firmware rollback by making reference to it.
28	Monotonic counter read	X	X		This service returns the value of the monotonic counter in the OTP in plaintext.
29	Write OTP	X			This service performs a write operation of a Key Wrap Key, a Key Derive Key, a Hash Digest or User Defined Data into the OTP in plaintext.
30	Read OTP	X	X		This service performs a read operation of the User Defined Data from the OTP in plaintext.
31	Public Key Hash Profile	X			This service performs a read operation of a Hash Digest of Public Key or the status of Public Key Valid Flag from the OTP, or an update operation of the status of Public Key Valid Flag. A Hash Digest of Public Key is supposed to use in the boot procedure of the processing system.
32	Debug Enable			X	This service controls the debug enable signal by 2-step procedure. Firstly, the module generates a "Challenge" that is a random number generated by DRBG implemented by the module. Secondly, the module compares supplied "Response" from outside, which is supposed to be computed by the Debug User by using SHA-256 Hash of a "Challenge" concatenated with Debug Enable Password, to the right value computed by the module.

					If the comparison succeeds, the module makes debug enable signal high.
33	Status Check	No authentication required			This service returns the status of the module.
34	Version Check	No authentication required			This service returns the hardware and the firmware version of the module.

*1) U: User, DU: Debug User

In the following table, lists of type of access to CSPs are shown.
The access types to CSPs are denoted as follows:

- 'R': the item is read or referenced by the service
- 'W': the item is written or updated by the service
- 'Z': the item is zeroized by the service

Table 4-4 Type of Access to CSPs within Services

	Service	CSP	Type of Access
1	Symmetric Encryption/Decryption	AES-ECB Key (128, 192, 256-bits) AES-CBC Key (128, 192, 256-bits) AES-CTR Key (128, 192, 256-bits) Triple DES Key (192-bits) Crypto Officer Password User Password	R
2	Symmetric Encryption/Decryption with Authentication	AES-GCM Key (128, 192, 256-bits) Crypto Officer Password User Password	R
3	Hash	Crypto Officer Password User Password	R
4	MAC	HMAC Key (160, 224, 256, 384, 512-bits) / AES-CMAC Key (128, 192, 256-bits) Crypto Officer Password User Password	R
5	Random Number Generation	DRBG Internal State Crypto Officer Password User Password	R
		DRBG Internal State	W
6	RSA-PKCS #1 v1.5 Sign	RSA Private Key Crypto Officer Password User Password	R

7	RSA-PKCS #1 v1.5 Verify	Crypto Officer Password User Password	R
8	RSA-PSS Sign	RSA Private Key Crypto Officer Password User Password	R
9	RSA-PSS Verify	Crypto Officer Password User Password	R
10	ECDSA Sign	ECDSA Private Key Crypto Officer Password User Password	R
11	ECDSA Verify	Crypto Officer Password User Password	R
12	ECC CDH Key Agreement	Shared Secret	W
		ECDH Private Key Crypto Officer Password User Password	R
13	ECC Multiplication	ECDSA Private Key, ECDH Private Key Crypto Officer Password User Password	R
14	ECC Public-Key Verify	Crypto Officer Password User Password	R
15	Key Derivation	Key Derive Key Crypto Officer Password User Password	R
16	Import Key	Key Wrap Key, Key Derive Key Crypto Officer Password User Password	R
		AES-ECB Key, AES-CBC Key, AES-CTR Key, AES-GCM Key, Triple DES Key, HMAC Key, AES-CMAC Key, RSA Private Key, ECDSA Private Key, ECDH Private Key, Key Derive Key, Key Wrap Key	W
17	Export Key	Key Wrap Key, AES-ECB Key, AES-CBC Key, AES-CTR Key, AES-GCM Key, Triple DES Key, HMAC Key, AES-CMAC Key, RSA Private Key, ECDSA Private Key, ECDH Private Key, Key Derive Key, Shared Secret Crypto Officer Password User Password	R
18	Delete Key	Key Wrap Key, AES-ECB Key, AES-CBC Key, AES-CTR Key, AES-GCM Key, Triple DES Key, HMAC Key, AES-CMAC Key,	Z

		RSA Private Key, ECDSA Private Key, ECDH Private Key, Key Derive Key, Shared Secret	
		Crypto Officer Password User Password	R
19	Clear OTP	Key Wrap Key, Key Derive Key	Z
		Crypto Officer Password	R
20	Delete All Keys	Key Wrap Key, AES-ECB Key, AES-CBC Key, AES-CTR Key, AES-GCM Key, Triple DES Key, HMAC Key, AES-CMAC Key, RSA Private Key, ECDSA Private Key, ECDH Private Key, Key Derive Key, Shared Secret, Crypto Officer Password, User Password, Debug User Password, Debug Enable Password, DRBG Internal State	Z
		Crypto Officer Password	R
21	Random Number Generator Configuration	DRBG Entropy Input, DRBG Nonce Input Crypto Officer Password	R
22	Self-Test	N/A	
23	Initialization	Crypto Officer Password, Debug User Password, Debug Enable Password, Key Wrap Key	W
		Crypto Officer Password	R
24	Authentication CO	Crypto Officer Password	R
25	Firmware Load	Key Wrap Key Crypto Officer Password	R
26	Define User	User Password	W
		Crypto Officer Password	R
27	Monotonic counter increment	Crypto Officer Password	R
28	Monotonic counter read	Crypto Officer Password User Password	R
29	Write OTP	Key Wrap Key, Key Derive Key	W
		Crypto Officer Password	R
30	Read OTP	Crypto Officer Password User Password	R
31	Public Key Hash Profile	Crypto Officer Password	R
32	Debug Enable	Debug User Password Debug Enable Password	R
33	Status Check	N/A	
34	Version Check	N/A	

*Key Wrap Key is AES-GCM Key which is 256-bits.

5. Physical Security

The module is a hardware module implemented as a sub-chip and is identified as a single-chip standalone module. The physical boundary is considered to be the perimeter of the FPGA. The FPGA conforms to the Level 1 requirements for physical security. The FPGA is a production grade component with industry standard passivation applied.

6. Operational Environment

The operational environment is non-modifiable. Therefore, this section is not applicable.

7. Cryptographic Key Management

7.1. Critical Security Parameters

The following table summarizes the generation, entry, storage, zeroization and output of CSPs that are used by the cryptographic services implemented in the module.

A number before a service name corresponds to the service number in “Table 4-3 Authenticated Services”.

Table 7-1 Critical Security Parameters

Name	Generation / Entry	Storage / Zeroization	Output
Key Derive Key	Internally generated by using 5. Random Number Generation service or 15. Key Derivation service, or entered in encrypted format by 16. Import Key service.	Stored in the SRAM/OTP in plaintext. Zeroized by 18. Delete Key service, 19. Clear OTP Service or 20. Delete All Keys Service.	Output in encrypted format by 17. Export Key service.
Key Wrap Key	Internally generated by 15. Key Derivation service, or entered in plaintext by 23. Initialization service, or entered in encrypted format by 16. Import Key service.	Stored in the SRAM/OTP in plaintext. Zeroized by 18. Delete Key service, 19. Clear OTP Service or 20. Delete All Keys Service.	Output in encrypted format by 17. Export Key service.
DRBG Entropy Input	Obtained from NDRNG by 21. Random Number Generator Configuration service	Not stored, then no need to be zeroized.	Never exit the module.
DRBG Nonce Input	Obtained from NDRNG by 21. Random Number Generator Configuration service	Not stored, then no need to be zeroized.	Never exit the module.
DRBG Internal State (Value, Constant and Counter)	Derived from entropy string as defined by [SP800-90A]	Stored in the SRAM in plaintext. Zeroized by 20. Delete All Keys Service.	Never exit the module.
AES-ECB Key	Internally generated by 15. Key Derivation service, or entered in plaintext / encrypted format by 16. Import Key service.	Stored in the SRAM in plaintext. Zeroized by 20. Delete All Keys Service.	Output in encrypted format by 17. Export Key service.
AES-CBC Key	Internally generated by 15. Key Derivation service, or entered in plaintext /	Stored in the SRAM in plaintext. Zeroized by 18. Delete Key service or 20. Delete All Keys	Output in encrypted format by 17.

	encrypted format by 16. Import Key service.	Service.	Export Key service.
AES-CTR Key	Internally generated by 15. Key Derivation service, or entered in plaintext / encrypted format by 16. Import Key service.	Stored in the SRAM in plaintext. Zeroized by 18. Delete Key service or 20. Delete All Keys Service.	Output in encrypted format by 17. Export Key service.
AES-GCM Key	Internally generated by 15. Key Derivation service, or entered in plaintext / encrypted format by 16. Import Key service.	Stored in the SRAM in plaintext. Zeroized by 18. Delete Key service or 20. Delete All Keys Service.	Output in encrypted format by 17. Export Key service.
Triple DES Key	Internally generated by 15. Key Derivation service, or entered in plaintext / encrypted format by 16. Import Key service.	Stored in the SRAM in plaintext. Zeroized by 18. Delete Key service or 20. Delete All Keys Service.	Output in encrypted format by 17. Export Key service.
HMAC Key	Internally generated by 15. Key Derivation service, or entered in plaintext / encrypted format by 16. Import Key service.	Stored in the SRAM in plaintext. Zeroized by 18. Delete Key service or 20. Delete All Keys Service.	Output in encrypted format by 17. Export Key service.
AES-CMAC Key	Internally generated by 15. Key Derivation service, or entered in plaintext / encrypted format by 16. Import Key service.	Stored in the SRAM in plaintext. Zeroized by Delete 18. Key service or 20. Delete All Keys Service.	Output in encrypted format by 17. Export Key service.
RSA Private Key	Entered in plaintext / encrypted format by 16. Import Key service.	Stored in the SRAM in plaintext. Zeroized by 18. Delete Key service or 20. Delete All Keys Service.	Output in encrypted format by 17. Export Key service.
ECDSA Private Key	Internally generated by 15. Key Derivation service, or entered in plaintext / encrypted format by 16. Import Key service.	Stored in the SRAM in plaintext. Zeroized by 18. Delete Key service or 20. Delete All Keys Service.	Output in encrypted format by 17. Export key service.
ECDH Private Key	Internally generated by 15. Key Derivation service, or	Stored in the SRAM in plaintext. Zeroized by 18. Delete Key	Output in encrypted

	entered in plaintext / encrypted format by 16. Import Key service.	service or 20. Delete All Keys Service.	format by 17. Export Key service.
Shared Secret	Internally generated by 12. ECC CDH Key Agreement service.	Stored in the SRAM in plaintext. Zeroized by 18. Delete Key service or 20. Delete All Keys Service.	Output in plaintext by 17. Export Key service.
Crypto Officer Password	Entered in plaintext by 23. Initialization service.	Stored in the OTP in plaintext. Zeroized by 20. Delete All Keys Service.	Output in plaintext by 24. Authentication CO service.
User Password	Entered in plaintext by 26. Define User service.	Stored in the SRAM in plaintext. Zeroized by 20. Delete All Keys Service.	Never exit the module.
Debug User Password	Entered in plaintext by 23. Initialization service.	Stored in the OTP in plaintext. Zeroized by 20. Delete All Keys Service.	Never exit the module.
Debug Enable Password	Entered in plaintext by 23. Initialization service.	Stored in the OTP in plaintext. Zeroized by 20. Delete All Keys Service.	Never exit the module.

7.2. Public Security Parameters

The following table summarizes the PSPs that are used by the cryptographic services implemented in the module.

Table 7-2 Public Security Parameters

Name	Description / Usage
ECDH Public Key	Used to compute Shared Secret.
ECDSA Public Key	Used by ECDSA Verify service.
RSA Public Key	Used by RSA-PKCS#1 v1.5 Verify service or RSA-PSS Verify public key service.

7.3. Random Number Generation

The module uses the HASH_DRBG for key generation. The inputs to the HASH_DRBG, that is the entropy input and the nonce input, are random bits which are collected from the NDRBG that consists of a series of ring oscillators. The minimum size of the entropy input and nonce input is 256-bits and 0-bits respectively. Therefore, in this case, Min-entropy of the seed (the entropy and the nonce) is approximately 256-bits at least. However, it is recommended that the size of the nonce is more than half of the entropy input (i.e., more than 128-bits) as security cushion. In this case, approximately 384-bits of security strength is provided for the HASH_DRBG at least.

7.4. Zeroization

The following three zeroization services are available.

- Delete All Keys service
- Delete Key service
- Clear OTP service

The Delete All Keys service zeroizes all CSPs. On the other hand, the other two services partly zeroize CSPs. The Delete Key service zeroizes the CSPs in the SRAM. The Clear OTP service zeroizes the CSPs in the OTP. For the CSPs that are zeroized by each service, refer to Table 4-4.

8. EMI/EMC

The module hardware component cannot be certified by the FCC as it is not a standalone device. It is a sub-chip to be embedded in a custom SoC. And a device which uses the custom SoC would undergo standard FCC certification for EMI/EMC.

According to 47 Code of Federal Regulations, Part 15, Subpart B, Unintentional Radiators, the module is not subject to EMI/EMC regulations because it is a subassembly that is sold to an equipment manufacturer for further fabrication. That manufacturer is responsible for obtaining the necessary authorization for the equipment with the module embedded prior to further marketing to a vendor or to a user.

9. Self-Tests

9.1. Power up self-tests

Power up self-tests consist of integrity tests and cryptographic algorithm tests.

9.1.1. Integrity Tests

The following table shows the list of integrity test that is part of power up self-test of the module.

Table 9-1 Integrity Test

Target	Test
ROM Firmware	32-bits CRC Check of the firmware in ROM

If the integrity test fails, the module enters error state.

9.1.2. Cryptographic Algorithm Tests

The following table shows the list of cryptographic algorithm tests that are part of power up self-test of the module.

According to IG 9.3, SHA-1, SHA-256 and SHA-512 are not necessary to be self tested, since each one of them is self-tested as a part of HMAC-SHA1, HMAC-SHA-256 and HMAC-SHA-512 respectively. Also, SHA-512/224 and SHA-512/256 are not necessary to be self-tested, since they are self-tested as a part of HMAC-SHA-512.

According to IG 9.4, since HMAC-SHA-256 and HMAC-SHA-512 are self-tested, SHA-224, SHA-384, HMAC-SHA-224 and HMAC-SHA-384 are not necessary to be self-tested.

According to IG 9.4, since HMAC-SHA-512 is self-tested, HMAC-SHA512/224 and HMAC-SHA-512/256 are not necessary to be self-tested.

According to IG 9.4, since AES (ECB/CBC) is self-tested, AES Encryption/Decryption in CTR is not necessary to be self-tested.

According to IG 9.4, since AES-CMAC is self-tested, AES Encryption/Decryption in GCM is not necessary to be self-tested.

According to IG 9.4, since RSA Signature Generation and Verification (PKCS#1_v1.5) are self-tested, RSAPSS Signature Generation and Verification are not necessary to be self-tested.

Table 9-2 Cryptographic Algorithm Test

Algorithm	Test
AES (ECB/CBC) encryption/decryption with 128-bits key	KAT
Triple-DES (CBC) encryption/decryption	KAT
HMAC-SHA-1/HMAC-SHA-256/HMAC-SHA-512	KAT
AES-CMAC	KAT
RSA Signature Generation (PKCS#1_v1.5, 2048-bits)	KAT
RSA Signature Verification (PKCS#1_v1.5, 2048-bits)	KAT
ECDSA Signature Generation (P-224)	KAT

ECDSA Signature Verification (P-224)	KAT
ECC Cofactor Diffie-Hellman Primitive “Z” Computation (P-256)	KAT
Hash-DRBG	KAT (SP 800-90A DRBG health testing for the instantiate, generate and uninstantiate functions)
KDF in Counter Mode using HMAC-SHA-256	KAT

If KAT fails, the module enters error state.

9.2. Conditional self-tests

The module performs conditional self-tests in the following cases.

(1) When firmware load service is executed
Authentication with AES-GCM for the loaded firmware image is performed.

(2) When random number generation service is executed with instantiate option
A Repetition Count Test (RCT) and an Adaptive Proportion Test (APT) are performed, whenever the DRBG is seeded.

(3) When key derivation service is executed for ECDSA and ECDH
A pair-wise consistency test on asymmetric keys generated for either ECDSA or ECDH is performed.
If conditional self-test fails, the module enters error state.

10. Design Assurance

10.1. Crypto Officer Guidance

The following descriptions are the services for startup the module, which are invoked by Crypto Officer. Assumptions regarding user behavior that is relevant to the secure operation of the module are described in Section 10.2.

The procedures of delivery and initialization operation are prescribed by "Socionext Secure Module User's Manual". If necessary, please see it.

Initialization

If Initialization service has not been invoked yet, Crypto Officer shall initialize the module. Initialization service is invoked only one time. In Initialization service, the module authenticates Crypto Officer with default Crypto Officer ID and Password and Crypto Officer shall update Crypto Officer ID and Password.

Authentication CO and Firmware Load

If Initialization service has been already invoked, Crypto Officer shall get Crypto Officer Password by Authentication CO service. If Firmware stored in the memory where is out of the module is loaded, Firmware Load service can be invoked as a part of Authentication CO service. If Firmware Load service is performed, the module is not FIPS-approved mode.

10.2. User Guidance

- The Shared Secret which is output of Export Key service shall be managed properly by a user.
- In order to meet the FIPS 140-2 IG 1.20 requirement, a user shall not input any keys in plaintext, if those are input directly from outside of the cryptographic physical boundary (that is, directly to the subsystem from outside of the FPGA).
- In order to meet the FIPS 140-2 IG A.13 requirement, a user shall check the number of the encryption with the same Triple DES Key is less than 2^{16}.
- If AES-GCM IVs are deterministically generated by the module using the protocols such as TLS and IPsec, a user shall generate the IV to meet the requirement per the FIPS 140-2 IG A.5 case 1. In this case, the IVs shall be constructed in compliance with the provisions of a peer-to-peer industry standard protocol.

  And if the IVs are generated regardless of the protocols, the IVs shall be generated as required per IG A.5 Case 3. In this case, the 96 bits IV provided from inside of the single chip but the outside of the module shall include an encoding of the module name. The name field shall allow for at least 2^{32} different names.
- If the module loses power and then it is restored, a new AES-GCM key shall be generated to use.
- The size of the nonce that is input to the HASH_DRBG should be more than half of the size of the entropy input.

11. Mitigation of Other Attacks

The module has not been designed to mitigate any specific attacks outside the scope of FIPS 140-2 requirements.