Hub Security Cryptographic Module FIPS 140-2 Non-Proprietary Security Policy

Hub Security LTD

Prepared by jtsec Beyond IT Security SL

Version: 20.12.2021.1.9

jtsec
BEYOND IT SECURITY

Index

Index of tables

Index of figures

1 REVISION HISTORY

Revision	Date	Remark
01.11.2019.1.0	01/11/2019	Initial version
13.04.2020.1.1	13/04/2020	Overall document modification
13.08.2020.1.2	13/08/2020	Overall document modification
01.09.2020.1.3	01/09/2020	Updated sections 2.1, 3.2, 3.3, 5.2, 8.3 and 13.3
23.10.2020.1.4	23/10/2020	Fixed some minor issues
15.12.2020.1.5	15/12/2020	Updated sections 3.2, 3.3, 5.2, 8 and 10
21.12.2020.1.6	21/12/2020	Fixed some minor issues in sections 3.1, 3.2, 5.2, 8.4, 10, 13.1 and 15
24.11.2021.1.7	24/11/2021	Fixed some minor issues in sections 3.1, 3.2, 3.3, 5.2, 6, 8, 8.1, 8.4, 10.1.1, 10.1.2, 10.2.2 and 10.2.3
15.12.2021.1.8	15/12/2021	Fixed some minor issues in sections 5.2, 8.5, 13.1 and 15
20.12.2021.1.9	20/12/2021	Added the note below the figure 4 and updated the section 15.

Table 1: Revision history

2 INTRODUCTION

2.1 OVERVIEW

This document is the non-proprietary FIPS 140-2 Security Policy for the Hub Security Cryptographic Module V4.4.0 (firmware version V2.0002.0050.0503.0725.080 and hardware version V.0002) which will also be referred to as "HS-Module" and "the module" though this document. This Security Policy specifies the security rules under which the module should operate to meet FIPS 140-2 Level 3 requirements.

This cryptographic module is an HSM Vault developed by Hub Security LTD to be used in a security and privacy solution. It is a keys management solution for safely storing and using all the company's sensitive data.

The FIPS 140-2 security levels for the module are as follow:

Security Requirements		Security Level
1	Cryptographic Module Specification	3
2	Cryptographic Module Ports and Interfaces	3
3	Roles, Services, and Authentication	3
4	Finite State Model	3
5	Physical Security	3
6	Operational Environment	N/A
7	Cryptographic Key Management	3
8	EMI/EMC	3
9	Self-Test	3
10	Design Assurance	3
11	Mitigation of Other Attacks	N/A
Overall Level		3

Table 2: Security requirements

2.2 DOCUMENT ORGANIZATION

This security policy is one part in a FIPS 140-2 submission package. The submission package contains:

- Security Policy: This document.
- Algorithm certificates: See section "3.2 Modes of operation and security functions".
- Functional specification and design documentation: See sections "3.1 Module description and cryptographic boundary", "4 Cryptographic module ports and interfaces" and [HSFS].
- User guide: See section "13 Crypto officer and user guidance".
- Finite state machine model: See [HSFSM].
- Configuration item list: See [HSCIL].

3 MODULE SPECIFICATION

The HS-Module is a multiple-chip standalone cryptographic module composed by an FPGA connected to an MCUs that is a bare-metal device whose responsibility is providing the module with tamper monitoring and cryptographic capabilities.

3.1 MODULE DESCRIPTION AND CRYPTOGRAPHIC BOUNDARY

As detailed above, the HS-Module is composed by:

1. One FPGA whose main responsibility is running a “mailbox” interface for all incoming messages and another one for all outgoing messages. The purpose of these “mailboxes” is to store and check every message going through Data input Interface, Data Output Interface, Control Input interface and Status Output interface to delete all the messages that do not comply with the correct format. Moreover, it is the one in charge of interconnecting the components related to the cryptographic operations, which are the MCU_S and the QRNG module. The FPGA also manages the Status output interfaces and the control input interfaces.

2. One MCU named as “MCU_S” that is responsible for the processing of the incoming and data messages to carry out the required cryptographic operations using a WolfSSL library instance. MCU_S uses a dedicated memory (Flash) for its operation and it is also connected to temperature and accelerometer sensors to monitor the tamper events created after configuring the sensor threshold. The version of the firmware running in this MCU is V2.0002.0050.0503.0725.080.

3. A debouncer IC that is connected to the MCU_S to detect the transition from logical ‘0’ value to logical ‘1’ and vice versa to avoid possible false transitions.

4. One Flash memory with a FAT filesystem that is used to store the internal data required for the module operation such as COs and users identifiers and their associated encrypted keys. The module stores all sensitive data on the internal flash memory encrypted with a Local Master Key, which is stored and protected in the internal volatile memory of MCU_S.

5. One RTC dedicated IC used to keep accurate timestamp.

6. One QRNG module, that is a physical quantum random number generator that uses quantum optics process to create true quantum randomness.

7. One power distribution IC responsible for regulating the input level of voltage and distribute the power to the components of the module.

8. One internal battery that is used by the module when it is not connected to an external power source.

9. An accelerometer and a temperature sensor responsible for implementing the tamper detection detailed in section 6 Physical security.

Considering the functionality depicted above, the physical and logical boundaries encompass all the components described above because all of them are related to the secure cryptographic operation of the module.

The "Figure 1: Physical and logical boundary of the HS-Module" depicts the block diagram of the module specifying the physical and logical boundaries for the HS-Module and showing the control input interfaces, status output interfaces, the input/output interfaces and the information flow:

- The messages arrive at the FPGA through the ***Data input bus*** that is the only **Data input interface** of the module and through the ***Control input bus*** that is the ***Control input interface.***

- Once a message is inside the module it follows the information flow indicated by the blue arrows:
  a) The FPGA checks the message validity verifying the values of its fields. If the message is invalid, then it is deleted. If the message is valid, FPGA stores the message in the internal buffer for retrieval by the MCU_S and indicates MCU_S that there is a waiting message.

  b) The MCU_S is responsible for carrying out the cryptographic operation required by the message.

  c) In case the cryptographic operation produces a result, it will be output from the module through the ***Data output bus*** that is the only **data output interface** of the module.

  d) The module contains three **control input interfaces** whose functionality is described below:

     - ***Sleep/Wake Up logic input pin***: The default state of this input is logical '1'. A transition from logical '1' to logical '0' can be used to cause the module goes to "sleep state" (to reset the module in case of error or finish the initialization process). Moreover, once the module is in "sleep state", a new transition from logical '1' to logical '0' can be used to wake up the module causing it starts to perform the power-up self-tests.

     - ***Wipe logic input pin:*** The default state of this input is logical '1'. A transition from logical '1' to logical '0' can be used to indicate the module to go to "Zeroization state".

     - ***Control synchronous interface*****:** This interface can be used to introduce the correct wiping confirmation code causing the zeroization of the module, to import or export keys from the module, to perform user management and to send system request to the module. The commands are passed as specific bit sequences (messages) into the module.

  e) The module contains three **status output interfaces**:

     - **Mode logic output pin:** The default state of this output is logical '0'. The output is logical '1' only when the module is in the approved mode of operation.

     - **Error logic output pin:** The default state of this output is logical '0'. The output is logical '1' only when the module is in an error state.

     - ***Status synchronous interface:*** This status interfaces can be used to output the current state of the module as a sequence of bits (messages) when the user sends

"**GetFsmStateRequest**" request to the module. This interface is also used to output response messages to the control requests.

f) The module contains one **power interface** to power the module and the internal battery through the power distribution IC.

Cryptographic module enclosure is a sealed metal box with a single DB37 standard connector. All module external interfaces are passing through a DB37 connector to an external host system. The metal enclosure is also the physical cryptographic boundary of the module.

Figure 1: Physical and logical boundary of the HS-Module

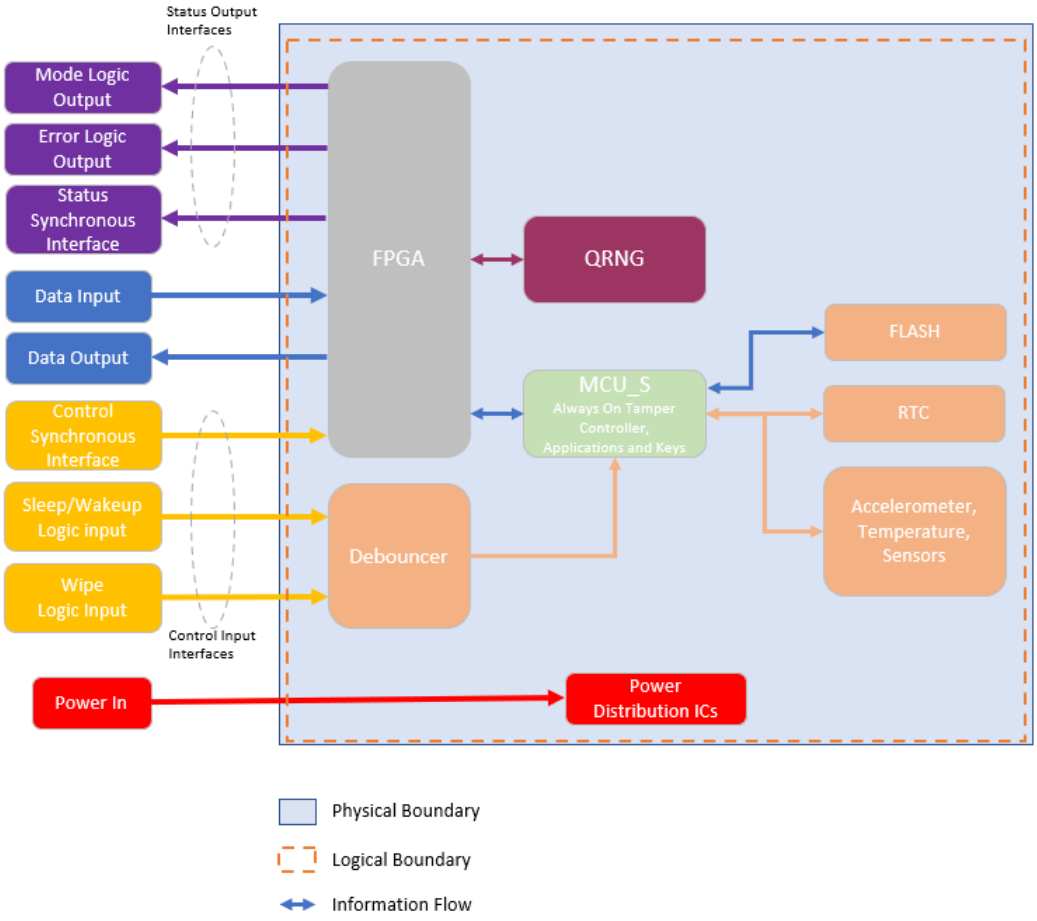

Figure 2: Physical and logical boundary of the HS-Module

3.2 MODES OF OPERATION AND SECURITY FUNCTIONS

The module only operates in FIPS 140-2 Approved mode because it implements the same FIPS-approved and vendor affirmed cryptographic operations as the WolfSSL library.

Algorithm	Mode	Key Sizes, Curves or Modulo (in bits)	Purpose	Cert
AES [FIPS 197] and KTS [SP 800-38F]	CBC [SP 800-38A]	Key sizes: 128, 192, 256	Encryption and decryption	#C2014
	CTR [SP 800-38A]	Key sizes: 128, 192, 256	Encryption and decryption	#C2014
	CCM [SP 800-38C]	Key sizes: 128, 192, 256	Authenticated encryption and decryption Message authentication	#C2014
	CMAC [SP 800-38B]	Key sizes: 128, 192, 256	Generation and Verification, Message authentication	#C2014
	ECB [SP 800-38A]	Key sizes: 128, 192, 256	Encryption and decryption	#C2014
	GCM [SP 800-38D]	Key sizes: 128, 192, 256 Tag length: 96, 104, 112, 120 and 128	Authenticated encryption and decryption Message authentication	#C2014
	KW [SP 800-38F]	Key sizes: 256	Encryption and decryption and key transport	#A876
DRBG [SP 800-90Arev1]	Hash_DRBG	SHA-256	Random bit generation	#C2014
ECDSA [FIPS 186-4]		P-224, P-256, P-384, P-521 SHA-1 (Disallowed for signature generation) SHA-224, SHA-256, SHA-384 and SHA-512	ECC key generation, Public key validation, Signature Generation, Signature Verification	#C2014
ECDSA [FIPS 186-4]		SHA3-224, SHA3-256, SHA3-384 and SHA3-512	Signature Generation, Signature Verification	#A876
HMAC [FIPS 198-1]		SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384 and SHA3-512	Generation and Verification, Message authentication	#C2014
RSA [FIPS 186-4]	PKCS v1.5 and PSS	K = 2048, 3072 SHA-1 (Disallowed for signature generation) SHA-224, SHA-256, SHA-384 and SHA-512	Key generation, signature generation and signature verification	#C2014
SHA [FIPS 180-4]		SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512	Message digest generation	#C2014

SHA-3 [FIPS 202]		SHA3-224, SHA3-256, SHA3-384 and SHA3-512	Message digest generation	#C2014
ENT (P) [SP 800-90B]	N/A	N/A	Entropy Generation for the DRBG Note: The module's entropy source produces a minimum estimated entropy of 0.82 bits for 1 of output. If the module's entropy source deteriorates to the point when the generation of a sufficient amount of entropy can no longer be guaranteed, this is detected by the module's health tests and will cause the module to enter an error state requiring a reset to recover (as detailed in [140IG] 7.18 bullet #2.	N/A

Table 3: Modes of operation and security functions

Note: Note that the module's algorithm implementation was CAVP tested for more algorithms/options than are actually implemented/used by the module. Only the algorithms listed above are actually implemented/used by the module

The vendor affirmed cryptographic operations are detailed in the following table:

Algorithm	Mode	Key Sizes, Curves or Modulo (in bits)	Purpose	Cert
CKG [SP 800-133rev2]			Cryptographic key generation	Vendor Affirmed
KAS-SSC (Key Agreement Scheme - Shared Secret Computation) [SP 800-56Arev3]	ECC (Elliptic Curve Cryptography)	P-256, P-384 and P-521	Key Agreement primitives	Vendor affirmed
KAS-KDF [SP 800-56Crev1]	One step key derivation	Key size:256 bits	Key Derivation	Vendor Affirmed

Table 4: Vendor affirmed modes of operation

Note: In accordance with FIPS 140-2 IG D.12, the module performs Cryptographic Key Generation (CKG) as per SP800-133rev2 (vendor affirmed). The resulting symmetric keys and seed used for asymmetric key generation are an unmodified output from a DRBG.

3.3 CRITICAL SECURITY PARAMETERS

This section details all the critical security parameters and public keys used by the HS-Module to be able to use the security functions and cryptographic operations detailed in the section above.

This table lists and describes the CSPs used by the HS-Module:

CSPs	Description
RSA_PK	Private component of an RSA key pair (2048 or 3072 bits) with the use determined by the caller.
EC_PK	Private component of an ECDSA key pair (P-224, P-256, P-384 and P-521) with the use determined by the caller.
ECDH_KAS_Priv_K	Private component of EC_Diffie Hellman Key Agreement with curves (P-256, P-384 and P-521).
ECDH_KAS_SS	The ECDH shared secret with 128,192 or 256 bits of security strength.
KH_Key	Keyed Hash key. Can be 128, 192 or 256 bits for CMAC and GMAC or 160, 256, 512 bits for HMAC.
RBG_Seed	The SHA-256 Hash_DRBG uses 1024 bits of entropy input, 128-256 bits of Nonce and 440 bits of Seed.
RBG _State	Hash_DRBG (SHA-256) state V(440 bits) and C(440 bits).
AES_EDK	AES (128, 192 or 256 bits) key used for symmetric encryption/decryption (including AES authenticated encryption). In case of AES-GCM, the module generates IVs internally using Approved DRBG with 96 bits in length.
LMK	AES 256 bits Local Master Key to encrypt/decrypt the keys stored in the internal flash memory
KD_DKM	Key Derivation derived keying material with 256 bits in length obtained through one step key derivation.

Table 5: List of Critical Security Parameters (CSPs) and private keys used by the HS-Module

The public keys used by the HS-Module are listed below:

CSPs	Description
RSA_Pub_K	Public component of an RSA key pair (2048 or 3072 bits) with the use determined by the caller.
EC_Pub_K	Public component of an ECDSA key pair (P-224, P-256, P-384 and P-521) with the use determined by the caller.
ECDH_KAS_Pub_K	Public component of EC_Diffie Hellman Key Agreement with curves (P-256, P-384 and P-521).

Table 6: Public keys used by the HS-Module

4 CRYPTOGRAPHIC MODULE PORTS AND INTERFACES

The table summarizes the associations between the logical interfaces required by FIPS 140-2 and the physical ports of the HS-Module shown in the following pictures:

Figure 3: Module external ports and enclosure

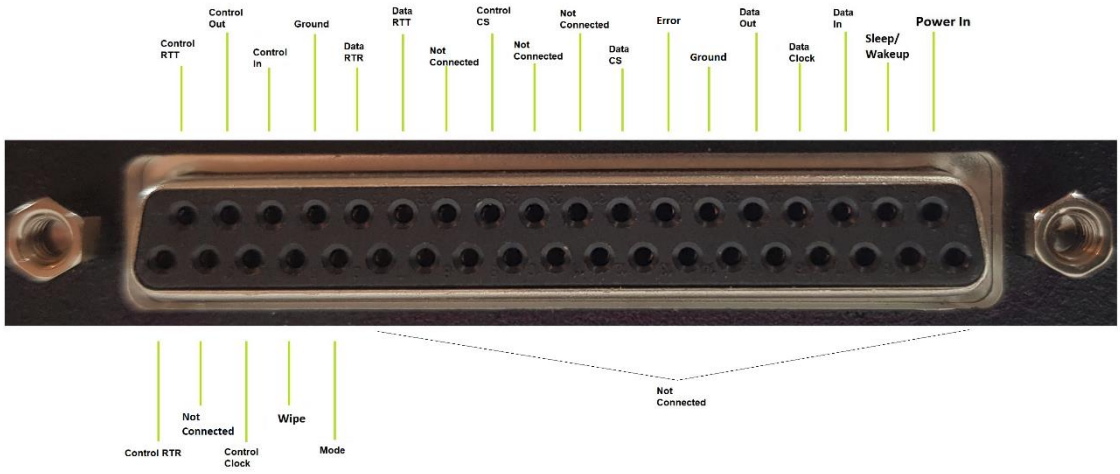

Figure 4: Detailed external DB37 connector pins

Note: The ground pin is related to power input because it is used as a reference point for measuring the voltage. It is not in itself providing any power to the module.

FIPS 140-2 logical interface	Pin ports	Cryptographic module physical port	Description and purpose
Data input interface	Data In Data Clock Data CS DATA RTR	DB37 connector	This is a synchronous digital interface for data input to the module. This interface works as a basic SPI interface, **Data CS** pin (SPI Chip Select) is set to '1' by the external host to indicate that a new request is ready to be injected. Once **Data RTR** pin (Ready to receive) that is used by the module to indicate to the external host that it is ready to receive a new request is set to '1', the module samples the information entered through the **Data In** pin synchronized to the rising edge of the **Data Clock** line. The module will disregard input data through **Data In** pin as long as **Data RTR** is '0'. The **Data In, Data Clock and Data CS** pins are input only configuration while the **Data RTR** is output only configuration. They can be used to: - Input messages to be verified and processed by the module using one of the cryptographic algorithms supported by the module. This can be done using one of the following request messages: **AesCbcRequest, AesEcbRequest, AesGcmRequest, AesCcmRequest, AesCmacRequest, DrbgRequest, EcdsaRequest,RsaRequest or ShaHmacRequest.** - Input keys and CSPs into the module when it receives one of the following request messages through the control input interface: **ImportKeyRequest, CreateKeyRequest or AddUserRequest.**
Data Output interface	Data Out Data Clock	DB37 connector	This is a synchronous digital interface for data output from the module. This

	Data CS Data RTT		interface works as a basic SPI interface, the **Data CS** pin is set to '1' by the external host to indicate that is ready to read a new response. Once the module has a data to be output, the **Data RTT** pin (Ready to Transmit) is set to '1' by the module and starts to output the information though the **Data Out** pin synchronized to the rising edge of the **Data Clock** line. The **Data Out, Data Clock and Data RTR** pins are output only configuration while the **Data CS** is input only configuration. This interface can be used to: - Output the result of the cryptographic operations carried out by the module. - Output keys and CSPs from the module when it receives one of the following request messages through the control input interface: **ExportKeyRequest or GetHSMPublicKeyRequest.**
Control input interface	Wipe	DB37 connector	This is the Wipe logic pin used to indicate the module to begin with the zeroization process if it is set to logical '0' value.
	Sleep/Wakeup	DB37 connector	This is the sleep/wakeup logic pin used to resume the normal operation of the module when it is in Error state and to power on the module when it is in sleep mode state. The sequence of changing the pin value from the default '1' to '0' will switch the module state between sleep and normal operation.
	Control In Control Clock Control CS Control RTR	DB37 connector	This is a synchronous digital interface for control input to the module. This interface works as a basic SPI interface, **Control CS** pin (SPI Chip Select) is set to '1' by the external host to indicate that a new control message is ready to be injected. Once **Control RTR** pin (Ready to receive) that is used by the module to indicate to the external host that it is ready to receive a new control message is set to '1', the module samples the

			information entered through the **Control In** pin synchronized to the rising edge of the **Control Clock** line. The module will disregard input data through **Control In** pin as long as **Control RTR** is '0'. This interface will be used as a control interface to: - Request the current FSM state using the **GetFsmStateRequest** message detailed in [HSFS] document. - Request to perform the backup of the ECDSA public key of the CO or the public/private application keys using the **ExportKeyRequest** message detailed in [HSFS] document. - Request to zeroize the Flash Memory using the **WipeRequest** message detailed in [HSFS] document. - Request to import keys public/private application keys using the **ImportKeyRequest** message detailed in [HSFS] document. - Create or delete application keys using the **CreateKeyRequest** and the **DeleteKeyRequest** messages detailed in [HSFS] document. - Add and remove user authorization to use key using the **AddUserToKeyRequest** and **DeleteUserFromKeyRequest** messages detailed in [HSFS] document. - Request the output of the HS-Module public certificate (ECC (P-521, SHA-512) using the **GetHSMPubKeyRequest** message detailed in [HSFS] document. - Request to create a CO in the HS-Module using the

			CoCreationRequest message detailed in [HSFS] document. - Request to configure the wiping code using the **WipeCodeSetRequest** message detailed in [HSFS] document. - Create or delete users from the module using the **AddUserRequest** and **DeleteUserRequest** messages detailed in [HSFS] document. - Configure the HS-module date and time using the **SetDateTimeRequest** message detailed in [HSFS] document. - Request to execute a firmware integrity test using the **RunFirmwareIntegrityRequest** message detailed in [HSFS] document. - Configure the tamper sensors and thresholds using the **SetTamperRequest** message detailed in [HSFS] document. - Request the detailed status log of the system and it firmware version using the **GetDetailedInfoRequet** message detailed in [HSFS] document. - Request the firmware version of the module using the **GetShortInforRequest** message detailed in [HSFS] document.
Status output interface	Mode	DB37 connector	This pin is used to indicate when the module is operating using a FIPS 140-2 Approved cryptographic algorithm.
	Error	DB37 connector	This pin is used to indicate when the module is in the Error state.
	Status Out Control Clock Control CS Status RTT	DB37 connector	This is a synchronous digital interface for control input to the module. This interface works as a basic SPI interface, the **Control CS** pin is set to '1' by the external host to indicate that is ready to read a new status output. Once the module has the status value to be output, the **Status RTT** pin (Ready to Transmit) is set to '1' by the module and starts to output the information status though the **Status Out** pin

			synchronized to the rising edge of the **Control Clock** line. This output interface is used to display the current status of the module based on the states defined in the [HSFSM] document.
Power in interface	Power In	DB37 connector	This pin is used to provide incoming power 5V-9V/500mA.

Table 7: HB-Module ports and interfaces

When the module is performing self-test, key zeroization, key generation or is in error state, all data output interfaces (except status output interface) are inhibited.

In addition, the HS-Module does not support the maintenance role, therefore it does not require to implement a maintenance interface.

5 ROLES, AUTHENTICATION AND SERVICES

5.1 ROLES AND AUTHENTICATION

The HS-Module supports identity-based authentication to all operators. The module supports a User and Crypto Officer roles to access to the services detailed in the table and does not provide maintenance role, nor bypass capability nor concurrent operators since it is a single thread module due to the FPGA only stores a single message at a time that is processed by the MCU_S.

The module requires authentication for each request received, therefore the module does not save authentication data between requests.

Role	Authorized services
User	- Power up the module - Call any approved cryptographic operation
Crypto Officer (CO)	- Secure initialization of the module - Secure key entry/output (Backup and restore) - Application keys management. - Key zeroization (wiping process). - Add/remove users and create keys for users.

Table 8: Users role and authorized services

Regarding the identity-based authentication, the user and crypto officer roles will be authenticated separately by the module using asymmetric key authentication (ECDSA) with 521 bits in length when accessing the specified services (P-521/SHA512). The module does not allow to assume a different role to the authenticated user. Hence, the probability that a random attempt succeeds or a false acceptance occurs is $P = \frac{1}{2^{256}}$ that is less than $\frac{1}{10^6}$. Moreover, the fastest time to reject an authentication message is 45 milliseconds, therefore, in the worst case the module will limit the user to perform a maximum of 1333 authentication attempts per minute, hence, the probability of a successful authentication attempt during one minute period is $P = \frac{1333}{2^{256}}$ that is less than $\frac{1}{10^5}$.

5.2 SERVICES

Once the crypto officer has emplaced the HS-Module in the secure room the user and crypto officer can use the following services and keys/CSPs depending on its type of access and the device functional specification.

The access types to CSPs are denoted as follows:

- 'R': Reading access
- 'W': Writing access
- 'X': Execution access

Authorized Services	Role	Description	Keys and CSPs	API function	Access
Module Initialization	CO	Secure initialization of the module	EC_Pub_K LMK	CoCreationRequest, WipeCodeSetRequest	W

		setting the wiping confirmation number and creating the COs			
Generate key	CO	Generate a symmetric key or an asymmetric key pair	AES_EDK RSA_PK, RSA_Pub_K, EC_PK, EC_Pub_K, ECDH_KAS_Priv_K, ECDH_KAS_Pub_K,	CreateKeyRequest	W
Signature generation/verification	CO User	Generate or verify a signature	RSA_PK EC_PK RSA_PUB_K EC_PUB_K	RsaRequest ECDSARequest	RW
Keyed hash	Co User	Generate or verify data integrity with CMAC, GMAC or HMAC	KH_Key	ShaHmacRequest	RX
Message Digest	CO User	Generate a message digest	N/A	ShaHmacRequest	N/A
Random	CO User	Generate random bits using the DRBG	RBG_Seed, RBG _State	DrbgRequest	RWX
Symmetric cipher	CO User	Encrypt/Decrypt data (Including authenticated encryption/decryption)	AES_EDK	AesCbcRequest, AesEcbRequest, AesGcmRequest, AesCcmRequest, AesCmacRequest	RX
Create user	CO	Create a new user in the HS-Module	EC_Pub_K	AddUserRequest	W
Delete user	CO	Delete a user in the HS-Module	All user keys	DeleteUserRequest	W
Key output (Perform backup)	CO	Output the keys backup from the module	EC_Pub_K for CO EC_Pub_K and EC_PK for application ECDH_KAS_SS KD_DKM	ExportKeyRequest	RX
Key input (Restore backup)	CO	Input keys into the module performing a backup restore	EC_Pub_K ECDH_KAS_SS KD_DKM	ImporKeyRequest	W
Zeroization	CO	Functions that destroy CSPs and public keys stored into the module. Clean up of the internal non-volatile flash	All keys and CSPs	WipeRequest	W

		memory of the MCU_S			
Tamper Configuration	CO	Enable/Disable tamper sensors and set the tamper response thresholds.	N/A	SetTamperRequest	N/A
Full System log	CO	Show detailed status of the module including the firmware version	N/A	GetDetailedInfoRequest	N/A
Configure the date and time	CO	Configure the date and time in the RTC	N/A	SetDateTimeRequest	N/A
Obtain ECDSA HS-module public key	CO User	Obtain ECDSA HS-module public key for communication purposes	EC_Pub_K	GetHSMPubKeyRequest	R
Delete application key	CO	Delete a symmetric key or an asymmetric key pair	AES_EDK RSA_PK, RSA_Pub_K, EC_PK, EC_Pub_K, ECDH_KAS_Priv_K, ECDH_KAS_Pub_K,	DeleteKeyRequest	W
Associate a key with a user	CO	Add user authorization to use a key	All type of keys	AddUserTokeyRequest	W
Disassociate a key from a user	CO	Remove user authorization to use a key	All type of keys	DeleteUserFromKeyRequest	W

Table 9: Description of authorized services

The HS-Module will provide the following services as unauthenticated services:

Authorized Services	Description	Keys and CSPs	API function	Access
Self-test	Run power-on self-test on demand	N/A	Module Reset/Reboot	N/A
Show-status	Show the current state of the module	N/A	GetFsmStateRequest	N/A
Get firmware version	Get the firmware version of the module	N/A	GetShortInfoRequest	N/A
Firmware integrity test	Perform firmware integrity check	N/A	RunFirmwareIntegrityTestRequest	N/A

	and return the result.			

Table 10: Unauthenticated services provided by the module

6 PHYSICAL SECURITY

The HS-Module is a multiple-chip standalone cryptographic module composed by an FPGA, an MCU and a QRNG physical module as detailed in “Figure 2: Physical and logical boundary of the HS-Module”. It is composed by production-grade components, it does not have any ventilation holes or slits and it is covered with a hard and opaque epoxy within the visible spectrum. Moreover, the module is contained in a sealed metal production-grade enclosure whose screws are covered with the same glue epoxy used to seal the enclosure in order to provide tamper evidence to comply with SL3 physical requirements.

Moreover, the module contains an accelerometer and temperature tamper sensors connected to the “MCU_S”. These sensors are configurable by the COs, according to the customer needs using the **SetTamperRequest** detailed in the [HSFS] document. The thresholds of each sensor can be configured to create a tamper event. If a tamper event occurs, the module will be zeroized and reset to the factory state requiring the CO to perform the initialization from the beginning before allowing access to the cryptographic functionality.

7 OPERATIONAL ENVIRONMENT

Because of the HS-Module is composed by an FPGA and an MCU in charge of providing the cryptographic functionality and the anti-tamper measures of the module, its operational environment can be classified as limited operational since it is a non-modifiable operational environment.

8 CRYPTOGRAPHIC KEY MANAGEMENT

This section encompasses the entire lifecycle of the CSPs detailed in the section "3.3 Critical security parameters" from its generation using the approved Hash_DRBG that uses QRNG module as ENT (P) to their zeroization. The HS-Module can perform symmetric and asymmetric key generation to carry out operations related to encryption/decryption, signature generation/verification, key establishment and key transport. In addition to the key generation, the module allows the CO to perform key entry and key output by performing backup and restore of the CO public keys.

In addition, the HS-Module allow the CO to delete a user and its associated keys from the system and it will zeroize all the secrets, private keys and CSPs in case of tamper event as detailed in section "6 Physical security".

8.1 RANDOM NUMBER GENERATION

The module uses an [SP 800-90Arev1] compliant Hash_DRBG for the generation of random numbers and also uses an ENT (P) for DRBG seeds.

This ENT (P) is based on the QRNG module as a source of randomness and the WolfSSL library instance run in the MCU. The module performs symmetric or asymmetric key generation after receiving a **CreateKeyRequest** message.

8.2 KEY GENERATION

The HS-Module uses the DRBG detailed in the section above for the creation of random data, which is used to generate symmetric keys and seeds for RSA, ECC key pair generation.

The module does not return intermediate key generation values.

8.3 KEY ESTABLISHMENT

The HS-Module implements key establishment protocol based on ECDH providing 128 or 256 bits of strength, in order to securely exchange secret keys.

8.4 KEY ENTRY AND OUTPUT

The module allows the CO to perform backup and restore of the public keys associated with all the crypto officers and the public and private keys application keys. The backup process consists of outputting the keys backup from the module and on the other side, the restore backup consists of entering the keys of the COs and/or application keys into the module.

The keys are electronically entered into and output from the module using a key transport method based on one of the key ECDH establishment methods (as detailed in [140IG] 7.7 that states "EXT CM Hardware to/from networked GPC", which qualifies as electronic distribution and electronic entry) detailed in the section above and ciphered with an AES 256 bits key which corresponds to the 32 bytes of the SHA-512 (KDA [SP 800-56Crev1]) of the shared secret derived by ECDH. Each key entered into or output from the module is associated with one of the COs or with one of the users created by the CO using the **AddUserToKeyRequest**.

To proceed with the entry or output of the keys from the module, the CO must first initialize the module as detailed in the section "13.4 Installation and initialization instructions" and once the modules reaches the "Approved Mode Operation" then the CO can send either the **ImportKeyRequest** or **ExportKeyRequest** which are signed by the CO P-512/SHA512 Key.

The module does not support manual key entry. Moreover, in order to comply with FIPS 140-2 standard requirements, all data output interfaces except the status output interface are inhibited during the key entry/output processes.

8.5 KEY STORAGE

The HS-Module stores entered and generated keys into the internal non-volatile flash memory. The stored keys are encrypted with the Local Master Key (LMK) that is an AES 256 key. The LMK is generated inside the MCU_S at the beginning of the initialization and never leaves the internal memory of MCU_S. Secret and private keys will not be accessible from outside of the module.

Each stored key will be associated with the allowed users using an internal table to specify the user, key usage and algorithm.

8.6 KEY ZEROIZATION

All the secret, private keys and CSPs stored into the module (including intermediate values generation keys) will be zeroized in one of the following two cases:

a) If the Wipe pin is set to logical '0' value and the wiping code is entered correctly by the CO.
b) As a tamper-event response.

All data output interfaces except the status output interface are inhibited during the key zeroization process. During this process the module overwrites the memory occupied by keys with 1s when the module receives the **WipeRequest** signed by the CO.

9 EMI/EMC

The module has been tested (at minimum) conform to the EMI/EMC requirements specified by 47 Code of Federal Regulations, Part 15, Subpart B, Unintentional Radiators, Digital Devices, Class B (i.e., for business use).

10 SELF-TEST

This section specifies the power-up and conditional self-tests performed by the HS-Module to ensure its correct functionality. If an error occurs during a self-test, the state of the module will be updated to "Error state" and indicated to the user via the status output interface. In addition, while the module is in error state, all data output interfaces except the status output interfaces are inhibited.

The self-tests will be executed in the MCU_S that provides the module with cryptographic functionalities.

10.1 POWER-UP SELF-TEST

Once the HS-Module is powered, the power-up self-tests are executed automatically updating its state from "Sleep mode" to "Self-test".

These tests consist of checking the firmware integrity of the source code contained in the module and checking the correct operativity of each cryptographic algorithm. To execute the power-up self-test, the module calls the **ResetHandler()** function.

10.1.1 FIRMWARE INTEGRITY TEST

The firmware integrity tests will check the integrity of the firmware contained in the FPGA in charge of receiving the messages to be processed by the module and will check the integrity of the MCU_S that provides the module with the cryptographic functionalities and the anti-tamper measures.

Considering the exposed above, the module will carry out the following integrity tests:

- Microcontroller Bootloader: The bootloader validates its own code integrity on each power up by verifying the error detection code (EDC) of 256 bits in length.
- FPGA code integrity verification carried out by the bootloader on each power up and based on verifying the 256 bits ECD of the FPGA image file. In addition, the FPGA itself has internal CRC32 check and will not work in case of CRC error.
- Integrity verification of the code flashed in the ROM of the MCU_S based on verifying the digital signature of the code SHA256 hash using ECDSA-SECP-256.

The function associated with this test is the **DoIntegrityTest()**.

10.1.2 CRYPTOGRAPHIC ALGORITHM TESTS

The module will perform the Known Answer Test (KAT) of the cryptographic operations implemented by the module to ensure the correct operativity of each cryptographic algorithm. These KATs involves operating the cryptographic algorithms on data for which the correct output is already known and comparing the calculated output with the previously generated output.

To execute the power-up self-test, the module calls the **DoSelfTest()** function which is responsible for calling the **DoKnownAnswerTests()** that will execute the KAS for all the cryptographic algorithms supported by the module:

Algorithm	Description
AES	KAT based on a separate encryption and decryption test using a 128 bits key and CBC mode. The function associated with this test is **AesKnownAnswerTest()**.

AES GCM	KAT based on a separate authenticated encryption and decryption test using 128 bits key and GCM mode. The function associated with this test is **AesGcm_KnownAnswerTest()**.
DRBG	KAT of the HASH_DRBG using SHA-256. The function associated with this test is **DRBG_KnownAnswerTest()**.
ECC PCT	Performs an ECC PCT (Pairwise Consistency Test) using the P-256 curve. The function associated with this test is **ECDSA_PairwiseAgreeTest()**.
HMAC	HMAC-SHA-1 (160-bit-key), HMAC-SHA-512 (512-bit key) and HMAC-SHA3-256 (256-bit key) KATs. The function associated with this test is **HMAC_KnownAnswerTest()**.
KAS ECC	[SP 800-56Arev3] Section 5.7.1.2. primitive "Z" computation KAT (per[140IG] 9.6), using P-256. The function associated with this test is **EccPrimitiveZ_KnownAnswerTest()**.
RSA	Separate Signature generation and signature verification KATs (k=2048), inclusive of the embedded SHA-256 (per FIPS 140-2 IG 9.2). The function associated with this test is **RsaSignPKCS1v15_KnownAnswerTest()**.
ENT (P)	Performs the start-up health tests. The function associated with this test is **QRNG_HEALTH_TEST().**

Table 11: Cryptographic algorithms power-up self-test description

Power-up self-test function returns 0 if all self-test succeeds, and 1 if not. If a self-test fails, the type of error is written in the log file and the module enters the error state and all data output interfaces except status output interface will be inhibited.

The cryptographic operations will not be able to be used until the self-tests are finished successfully. If a self-test fails, the invocation of any cryptographic operation will fail. The only way to resume the normal operation of the module from a self-test failure is by resetting the module and setting the Sleep/Wakeup pin to logical '0' value.

10.2 CONDITIONAL SELF-TEST

Because the module implements the public/private key generation and employs continuous random bit generation, it is necessary to implement the conditional self-tests specified in this section.

10.2.1 PAIRWISE CONSISTENCY TEST

The following pairwise consistency tests will be executed in order to check the correct operativity of the module during the public/private keys generation:

Algorithm	Description
ECC PCT	ECC key Generation Pairwise Consistency Test performed on ECC key pair generation. The function associated with this test is **wc_ecc_check_key().**
RSA PCT	RSA key Generation Pairwise Consistency Test performed on RSA key pair generation and signature generation and verification. The function associated with this test is **wc_CheckRsaKey ().**

Table 12: Pairwise consistency tests

10.2.2 CONTINUOUS RANDOM GENERATOR TEST

The module will execute the following tests continuous random generator test for ENT and DRBG algorithms as required by the FIPS 140-2 standard:

Algorithm	Description
DRBG	[SP 800-90Arev1] section 11.3 instantiates, generate, reseed health tests for SHA-256 Hash_DRBG. The function associated with this test is **RNG_HealthTest_fips ()**.
ENT (P)	Continuous Random Generation Test of 64 bits block on the output. The function associated with this test is **wc_RNG_TestSeed**.

Table 13: Continuous random generator test

10.2.3 CONTINUOUS HEALTH-TESTS

The module will execute the following continuous Health-tests for the ENT (P) in compliance with the [SP 800-90B] document:

Algorithm	Description
RCT	The Repetition Count Test is performed as specified in section 4.4.1 of the [SP 800-90B]. The function associated with this test is **Repetition_Count_Test().**
APT	The Adaptative Proportion test is performed as specified in section 4.4.2 of the [SP 800-90B]. The function associated with this test is **Adaptive_Proportion_Test().**

Table 14: Continuous Health-tests

10.2.4 CONDITIONAL TEST FOR ASSURANCE

The module will execute the following conditional test for assurance as stated in [IG 9.6] and [SP 800-56Arev3]:

Algorithm	Description
ECC conditional Test for assurance	Conditional Tests for Assurances are performed following sections 5.5.2, 5.6.2 and 5.6.3 of [SP 800-56Arev3].

Table 15: Conditional test for assurance

11 MITIGATION OF OTHER ATTACKS

The module is not designed to mitigate other attacks which are outside of the scope of FIPS 140-2.

12 DESIGN ASSURANCE

12.1 CONFIGURATION MANAGEMENT

The configuration management list is composed by the Configuration Items version control, change control, flaw remediation tracking and the source code revision which are managed by Hub Security LTD in a private Gitlab repository with write access restricted to the authorized developers and multifactor authentication processes.

12.2 CONFIGURATION ITEMS IDENTIFICATION METHOD

The internal versioning of the VHDL FPGA source code and C/C++ MCU source code is performed automatically and the assigned version and revision are used internally to control the code development. However, this version and revision are not the same as the final released version of the VHDL and C/C++ codes assigned manually using a comment.

The firmware version of the MCU will be assigned with the following format "MODULE.HHHH.FFFFF.BBBB.VVVV.SSS", where:

- MODULE - unique module name.
- HHHH - hardware version number between 0000 and 9999.
- FFFFF - FPGA version number between 0000 and 9999.
- BBBB – MCU_S Bios version number between 000 and 9999.
- VVVV - MCU_S bootloader version number between 0000 and 9999.
- SSS - MCU_S FIPS application version number between 000 and 999.

Moreover, the HS-Module will be able to return the source code version after receiving the **GetShortInfoRequest** message.

The associated module documentation is manually versioned by appending the date, the major version and minor version on their name as following "NAME.dd.mm.yyyy.A.B" with the following naming convention:

- NAME - unique document name.
- dd - day number between 01 and 31.
- mm - month number between 01 and 12.
- yyyy - year number between 0000 and 9999.
- A - major version number.
- B - minor version number.

The configuration item list can be consulted in the [HSCIL] document.

13 CRYPTO OFFICER AND USER GUIDANCE

13.1 OPERATION RULES

When the module is installed and securely initialized as detailed in the section "13.4 Installation and initialization instructions", it is initialized in FIPS mode that is the only mode of operation that complies with the following rules:

1. The HS-Module is initialized in FIPS mode of operation automatically after the self-test are completed successfully.
2. The replacement or modification of the module by unauthorized users is prohibited.
3. Once the module is powered, the power-up self-tests are executed automatically without requiring any operator additional action to be executed.
4. All data output interfaces except status output interfaces will be inhibited during the key entry, power-up and conditional self-tests, keys zeroization and error state.
5. The module does not implement critical security functions to be tested during the self-test.
6. The zeroization affects the internal volatile and non-volatile flash memory of the MCU_S that is in charge of executing the instance of the WolfSSL library in order to provide the module with cryptographic operativity.
7. The HS-Module does not support a maintenance interface nor maintenance role.
8. The HS-Module does not provide bypass capability.
9. The module provides identity-based authentication using asymmetric key authentication (ECDSA) with 521 bits in length to comply with the authentication requirements for Security Level 3.
10. The module does not support manual key entry, because all the keys will be entered using electronic input and output methods as stated in [140IG] 7.7.
11. The module implements power-up self-tests to ensure the integrity of the firmware and the correct cryptographic operation of each approved cryptographic algorithm supported by the module. The power-up self-test can be performed on demand by resetting the module.
12. The module implements conditional self-test to ensure the correctness of the random number generation process, the asymmetric key generation process and key establishment process.
13. The public and private keys are entered into or output from the module using a valid key establishment method via software ciphered with an AES 256 bits key.
14. All the keys are stored into the internal non-volatile flash memory encrypted using the LMK and specifying the type of key and associated with its owner using the user ID.
15. The crypto officer is in charge of uploading the keys backup of the CO during the module initialization and inserting the wiping confirmation code to perform the key zeroization.
16. If the module is in error state, the cryptographic operativity of the module will be disabled.
17. The normal operation of the HS-Module can be resumed from an Error state by resetting the module by setting the Sleep/Wakeup pin to logical '0' value.
18. The CO is responsible for performing periodic inspection of the epoxy applied to the screws of the module in order to ensure the physical security maintenance of the module. Moreover, if evidence of tamper, the CO should contact Hub Security immediately.

13.2 SECURE DISTRIBUTION

The module will be shipped to the costumers via certified courier service by Hub Security LTD, and it will be shipped in boxes and contained in bags with Hub Security tamper-evident and labelled with a serial

number that is provided to the customer when the module is acquired; therefore, the customer will be able to check the integrity box and the tamper seals before opening it.

13.3 INTEGRITY AND CONFIDENTIALITY ASSURANCE

When the module is received, the Crypto officer can verify the HS-module integrity verifying that the tamper seals of the shipping box,bag and the HS-Module has not been tampered. The CO can also compare the serial numbers on the tamper evident bags with the information as provided by Hub Security.

If the epoxy applied to the screws of the enclosure is intact, the serial numbers are correct, the module metal enclosure is intact and the module is operative, the CO can verify firmware version of the module by following the installation and initialization instructions detailed in the following section.

Otherwise, if the Crypto Officer notices that the evident bag or the epoxy that protects the screws of the enclosure of the HS-Module have been manipulated, then the HS-Module should not be initiated nor used for secure operation. Hub Security should be contacted to arrange the return of the module to Hub Security facilities for inspection and any necessary repair.

13.4 INSTALLATION AND INITIALIZATION INSTRUCTIONS

Once the shipping box or bag and the module have been physically inspected by the CO to ensure they have not tampered (the tamper-evident is intact), the CO must use the following instructions in order to emplace, install and initialize the module in a secure manner:

a) Plug the module to the electric network and connect it to the host PC (or equivalent appliance with the necessary SPI ports and GPIO ports).
b) Set the Sleep/Wakeup pin in sequence ‘1’-‘0’-‘1’ logical value in order to make the module, to begin with, the power-up self-tests.
c) Enter the **COCreationRequest** through the Data Input Interface once the power-up self-tests have been finished with success.
d) The first CO is added using the manufacturer supplied initialization code. The additional two COs are added by the majority of digital signatures of existing COs.
e) Set the Wipe code using the **SetWipeCodeRequest**.
f) Set the Sleep/Wakeup pin from logical '1' to logical '0' (falling edge) to switch the module from Sleep state to Powerup state and wait until the power-up self-tests are finished with success.
g) Once the power-up self-tests are successfully passed, the module is set in Approved Mode of Operation.
h) Sent to the module the **GetShortInfoRequest** to verify the correct version of the firmware (V2.0002.0050.0503.0725.080)

13.5 SECURE OPERATION

After the CO installs and initializes the module as described in the previous section, the module will be in the **ApprovedMode** state and will allow the user and crypto officer to use the authorized services and API functions detailed in section “5.2 Services” related to any cryptographic operation, key zeroization, module configuration, user management, obtain the module status or perform a self-test.

To interact with the module, the user and the CO can use the ports defined in section “4 Cryptographic module ports and interfaces” without any additional measure or special behavior, due to the module is always operating in FIPS mode.

14 GLOSSARY AND ABBREVIATIONS

AES	Advanced Encryption Standard
CBC	AES Cipher Bloc Chaining
CCM	Counter with CBC-MAC
CMAC	Cipher-based Message Authentication Code
CO	Crypto Officer
CRC32	Cyclic Redundancy check of 32 bits
CRNGT	Continuous Random Number Generator Test
CTR	AES Counter Mode
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code block
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EDC	Error Detection Code
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FIPS	Federal Information Processing Standard
FPGA	Field Programmable Gate Array
GCM	Galois Counter Mode
GPIO	General Purpose Input/Output
HMAC	Hash-Based Message Authentication Code
HSM	Hardware Security Module
HS-Module	Hub Security Cryptographic Module V4.4.0
IG	Implementation Guidance
IUT	Instrument Under Test
KAT	Known Answer Test
LAN	Local Area Network
LMK	Local Mater Key
MCU	Microcontroller Unit
NDNRNG	Non-Deterministic Random Number Generator
NIST	National Institute of Standard and Technology
OTP	One Time Programable
PCT	Pairwise Consistency Test
PKCS	Public-Key Cryptographic Standards
PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir y Adleman
SHA	Secure Hash Algorithm
SP	Security Policy
USB	Universal Serial Bus
FIPS	Federal Information Processing Standard
FPGA	Field Programmable Gate Array

15 REFERENCE DOCUMENT

140IG	Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program
HSSP	Hub Security FIPS 140-2 Security Policy.20.12.2021.1.9
HSCIL	Hub Security FIPS 140-2 Configuration Item List.20.12.2021.1.9
HSFSM	Hub Security FIPS 140-2 Finite State Model.20.12.2021.1.9
HSFS	Hub Security FIPS 140-2 Functional Specification.20.12.2021.1.9
FIPS 197	Advanced Encryption Standard (AES), Federal Information Processing standards publication 197
SP 800-38A	Recommendation for Block Cipher Modes of Operation, Methods and techniques
SP 800-38B	Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication
SP 800-38C	Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality
SP 800-38D	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC
SP 800-90Arev1	Recommendation for Random Number Generation Using Deterministic Random Bit Generators
SP 800-56Arev3	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography
FIPS 186-4	Digital Signature Standard (DSS)
FIPS 198-1	The Keyed-hash Message Authentication Code
FIPS 180-4	Secure Hash Standard (SHS)
SP 800-133rev2	Recommendation for Cryptographic Key Generation
SP 800-56Crev1	Recommendation for key-Derivation Methods in Key-Establishment Schemes
FIPS 202	SHA-3 Standard: Permutation-Based hash and Extendable-Output Functions