xFusion Digital Technologies Co., Ltd.

xFusion Cryptographic Library

FIPS 140-3 Non-Proprietary Security Policy

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy of the xFusion Cryptographic Library v2.0.0. For the purpose of the FIPS 140-3 validation, the module is a software cryptographic module. It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-3 (Federal Information Processing Standards Publication 140-3) for an overall Security Level 1 module.

1.2 Security Levels

Section	Title	Security Level
1	General	1
2	Cryptographic module specification	1
3	Cryptographic module interfaces	1
4	Roles, services, and authentication	1
5	Software/Firmware security	1
6	Operational environment	1
7	Physical security	N/A
8	Non-invasive security	N/A
9	Sensitive security parameter management	1
10	Self-tests	1
11	Life-cycle assurance	1
12	Mitigation of other attacks	1
	Overall Level	1

Table 1: Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The xFusion Cryptographic Library v2.0.0 (hereafter referred to as “the module”) is a Software Multichip standalone cryptographic module. The module provides cryptographic services to applications running in the user space of the underlying operating system through a C language Application Program Interface (API).

The module is composed by the shared library fips.so, which is an OpenSSL provider. Providers are containers for algorithm implementations. Whenever a cryptographic algorithm is used via OpenSSL high level APIs, a provider is selected.

The cryptographic boundary of the Module is the provider itself, a dynamically loadable library. The module performs no communication other than with the calling application via APIs that are invoked by the module.

Module Type: Software

Module Embodiment: MultiChipStand

Cryptographic Boundary:

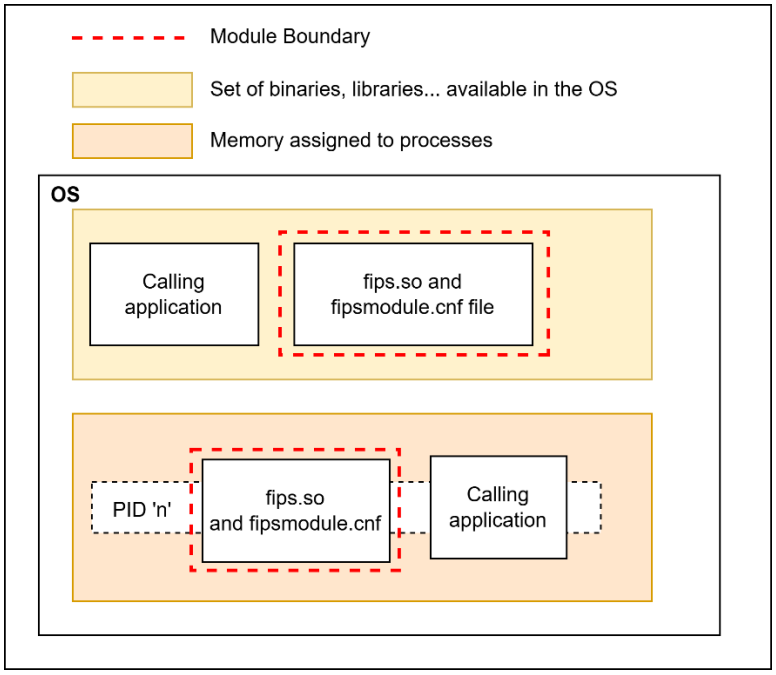
The cryptographic boundary consists of the following shared library and integrity check file. The following table enumerates the files that comprise the module (surrounded with red lines in Figure 1: Cryptographic Boundary).

Component	Description
fips.so	Shared library for the provider implementation v2.0.0
fipsmodule.cnf	File with HMAC authentication code for integrity for the fips.so shared library

Table 2: Cryptographic Module Components

The image below illustrates the high-level software architecture of the module. The block diagram below shows the module and the delimitation of the cryptographic module boundary (dotted red line).

Figure 1: Cryptographic Boundary



Tested Operational Environment's Physical Perimeter (TOEPP):

The block diagram in Figure 2 shows the cryptographic boundary of the module (blue square), its interfaces with the operational environment and the flow of information between the module and operator (depicted through the arrows). The physical perimeter of the general-purpose computing system comprises the module's TOEPP. The Baseboard Management Controller (BMC) is an embedded server management subsystem that provides out-of-band platform management capabilities.

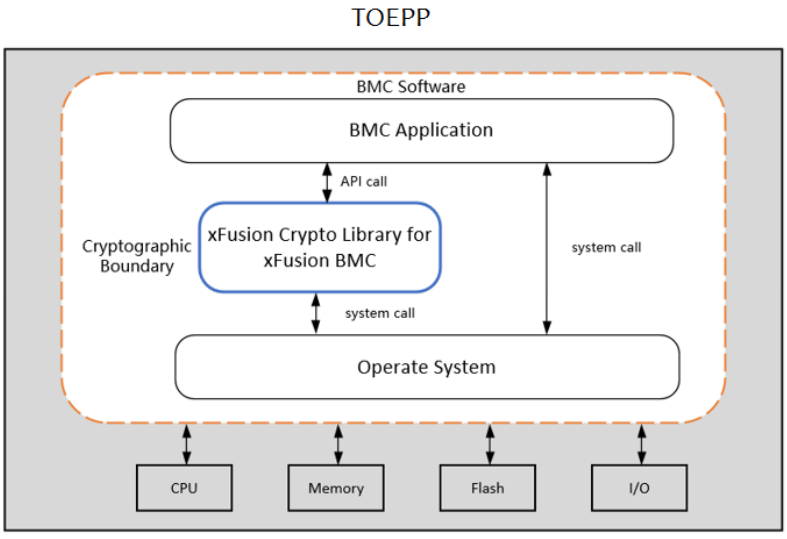


Figure 2 – Block Diagram depicting the cryptographic boundary (within the highlighted blue rectangle) and data flow between the module interfaces and operator. The boundary also includes the instantiation of the cryptographic module in memory.

The TOEPP is the physical perimeter of both hardware platforms listed in Table 4.

The Module performs no communications other than with the calling application (the process that invokes the Module services) and the OS syslog. The boundary also includes the instantiation of the module saved in memory

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Hardware:

N/A for this module.

Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets):

Package or File Name	Software/ Firmware Version	Features	Integrity Test
fips.so	2.0.0	None	Message authentication with HMAC-SHA2-256

Table 3: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)

The module operates in a modifiable operational environment. The module runs on a commercially available general-purpose operating system. The module executes on the hardware specified Table 4. The module does not support concurrent operators.

Tested Module Identification – Hybrid Disjoint Hardware:

N/A for this module.

Tested Operational Environments - Software, Firmware, Hybrid:

The module has been tested on the platforms indicated in the following table, with the corresponding module variants and configuration options.

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Linux 5.10	2288H V7	Hi1711	No	None	2.0.0
Linux 5.10	2258 V7	Hi1711	No	None	2.0.0

Table 4: Tested Operational Environments - Software, Firmware, Hybrid

Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid:

The vendor claims the following platforms to be vendor affirmed - that is, the module functions the same way and provides the same services on the following systems, for which operational testing and algorithm testing was not performed

Operating System	Hardware Platform
Linux 5.10	2288H V8
Linux 5.10	2188H V8

Operating System	Hardware Platform
Linux 5.10	2288 V8
Linux 5.10	2258H V8
Linux 5.10	2158H V8
Linux 5.10	1288 V8
Linux 5.10	1158H V8
Linux 5.10	1258H V8
Linux 5.10	5288 V8
Linux 5.10	G8600 V8
Linux 5.10	G6500E V8
Linux 5.10	G6500 V8
Linux 5.10	G6550 V8
Linux 5.10	1158H V7
Linux 5.10	1288H V7
Linux 5.10	1258H V7
Linux 5.10	2258H V7
Linux 5.10	2288 V7
Linux 5.10	2488H V7
Linux 5.10	5288 V7
Linux 5.10	5885H V7
Linux 5.10	5298 V7
Linux 5.10	1288H V6
Linux 5.10	2288H V6
Linux 5.10	2288E V6
Linux 5.10	2488H V6
Linux 5.10	5288 V6
Linux 5.10	5885H V6
Linux 5.10	G5200 V7
Linux 5.10	G5500 V7
Linux 5.10	G8600 V7
Linux 5.10	G5500 V6

Operating System	Hardware Platform
Linux 5.10	GN560E V7
Linux 5.10	X6000 V6
Linux 5.10	CX5200 V5
Linux 5.10	1288H V5
Linux 5.10	2288H V5
Linux 5.10	5288 V5

Table 5: Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid

CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

2.3 Excluded Components

There are no excluded components for the module.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved	The module will be in Approved mode when all pre-operational self-tests have been completed successfully, and only Approved security functions can be invoked.	Approved	Pre-operational self-tests successfully passed (SELF_TEST_post() =1)

Table 6: Modes List and Description

The module supports only one mode of operation: Approved. The module will be in Approved mode when all pre-operational and conditional self-tests have been completed successfully, and only Approved security functions can be invoked.

Degraded Mode Description:

The module does not support degraded mode of operation.

2.5 Algorithms

Approved Algorithms:

The table below lists the approved security functions (or cryptographic algorithms) of the module, including specific properties.

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A6648	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC-CS1	A6648	Direction - decrypt, encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC-CS2	A6648	Direction - decrypt, encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC-CS3	A6648	Direction - decrypt, encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A6648	Key Length - 128, 192, 256	SP 800-38C
AES-CFB1	A6648	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB128	A6648	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A6648	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A6648	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B
AES-CTR	A6648	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A6648	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A6648	Direction - Decrypt, Encrypt IV Generation - External, Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-GMAC	A6648	Direction - Decrypt, Encrypt IV Generation - External, Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-KW	A6648	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KWP	A6648	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-OFB	A6648	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-XTS Testing Revision 2.0	A6648	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E

Algorithm	CAVP Cert	Properties	Reference
Counter DRBG	A6648	Prediction Resistance - Yes Mode - AES-128, AES-192, AES-256 Derivation Function Enabled - No, Yes	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-5)	A6648	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Secret Generation Mode - testing candidates	FIPS 186-5
ECDSA KeyVer (FIPS186-5)	A6648	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A6648	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Component - No, Yes	FIPS 186-5
ECDSA SigVer (FIPS186-5)	A6648	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	FIPS 186-5
Hash DRBG	A6648	Prediction Resistance - Yes Mode - SHA-1, SHA2-256, SHA2-512, SHA3-256, SHA3-512	SP 800-90A Rev. 1
HMAC DRBG	A6648	Prediction Resistance - Yes Mode - SHA-1, SHA2-256, SHA2-512, SHA3-256, SHA3-512	SP 800-90A Rev. 1
HMAC-SHA-1	A6648	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A6648	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A6648	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A6648	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A6648	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512/224	A6648	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512/256	A6648	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-224	A6648	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-256	A6648	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-384	A6648	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1

Algorithm	CAVP Cert	Properties	Reference
HMAC-SHA3-512	A6648	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KAS-ECC CDH-Component SP800-56Ar3 (CVL)	A6648	Function - Full Public Key Validation, Key Pair Generation, Partial Public Key Validation Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521	SP 800-56A Rev. 3
KAS-ECC-SSC Sp800-56Ar3	A6648	Domain Parameter Generation Methods - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-FFC-SSC Sp800-56Ar3	A6648	Domain Parameter Generation Methods - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Scheme - dhEphem - KAS Role - initiator, responder	SP 800-56A Rev. 3
KDA HKDF SP800-56Cr2	A6648	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-8192 Increment 8 HMAC Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	SP 800-56C Rev. 2
KDA OneStep SP800-56Cr2	A6648	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-8192 Increment 8	SP 800-56C Rev. 2
KDF ANS 9.42 (CVL)	A6648	KDF Type - DER Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Key Data Length - Key Data Length: 8-4096 Increment 8	SP 800-135 Rev. 1
KDF ANS 9.63 (CVL)	A6648	Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Key Data Length - Key Data Length: 128, 4096	SP 800-135 Rev. 1
KDF SP800-108	A6648	KDF Mode - Counter, Feedback Supported Lengths - Supported Lengths: 8, 72, 128, 776, 3456, 4096	SP 800-108 Rev. 1

Algorithm	CAVP Cert	Properties	Reference
KDF SSH (CVL)	A6648	Cipher - AES-128, AES-192, AES-256 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
KMAC-128	A6648	Message Length - Message Length: 0-65536 Increment 8 Key Data Length - Key Data Length: 128-1024 Increment 8	SP 800-185
KMAC-256	A6648	Message Length - Message Length: 0-65536 Increment 8 Key Data Length - Key Data Length: 128-1024 Increment 8	SP 800-185
KTS-IFC	A6648	Modulo - 2048, 3072, 4096, 6144 Key Generation Methods - rsakpg1-basic, rsakpg1-crt, rsakpg1-prime-factor, rsakpg2-basic, rsakpg2-crt, rsakpg2-prime-factor Scheme - KTS-OAEP-basic - KAS Role - initiator, responder Key Transport Method - Key Length - 1024	SP 800-56B Rev. 2
PBKDF	A6648	Iteration Count - Iteration Count: 1-10000000 Increment 1 Password Length - Password Length: 8-128 Increment 8	SP 800-132
RSA KeyGen (FIPS186-5)	A6648	Key Generation Mode - probableWithProbableAux Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 Modulo - 2048, 3072, 4096, 6144, 8192 Primality Tests - 2powSecStr Private Key Format - standard	FIPS 186-5
RSA SigGen (FIPS186-5)	A6648	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
RSA SigVer (FIPS186-4)	A6648	Signature Type - PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-5)	A6648	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
Safe Primes Key Generation	A6648	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3
Safe Primes Key Verification	A6648	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
SHA-1	A6648	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-224	A6648	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A6648	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A6648	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A6648	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512/224	A6648	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512/256	A6648	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA3-224	A6648	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-256	A6648	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-384	A6648	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-512	A6648	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHAKE-128	A6648	Output Length - Output Length: 16-65536 Increment 8	FIPS 202
SHAKE-256	A6648	Output Length - Output Length: 16-65536 Increment 8	FIPS 202
TLS v1.2 KDF RFC7627 (CVL)	A6648	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
TLS v1.3 KDF (CVL)	A6648	HMAC Algorithm - SHA2-256, SHA2-384 KDF Running Modes - DHE, PSK, PSK-DHE	SP 800-135 Rev. 1

Table 7: Approved Algorithms

The table above lists the approved security functions (or cryptographic algorithms) of the module, including specific key lengths employed for approved services, and implemented modes or methods of operation of the algorithms.

Note: When the HMAC algorithm is used, the "*hmac_flag*" parameter shall be set to 1.

Note 2: RSA FIPS 186-4 signature verification is only allowed for legacy use.

Note 3: KAS-ECC CDH-Component (CVL) shall only be used within the context of an SP 800-56Arev3 KAS.

Vendor-Affirmed Algorithms:

The table below lists the vendor-affirmed algorithms that are allowed in the approved mode of operation.

Name	Properties	Implementation	Reference
CKG	Key Type:Symmetric and Asymmetric	N/A	SP 800-133r2 and IG D.H: Per Section 4, example 1

Table 8: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

The module does not implement any Non-Approved Algorithms Allowed in the Approved Mode of Operation.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

The module does not implement any Non-Approved Algorithms Allowed in the Approved Mode of Operation with no security claimed.

Non-Approved, Not Allowed Algorithms:

N/A for this module.

The module does not implement either non-approved allowed algorithms with no security claimed or non-approved and not allowed algorithms in the Approved mode of operation.

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Asymmetric Key Generation	AsymKeyPair-DomPar AsymKeyPair-KeyGen CKG	Asymmetric Key Pair/domain Parameter Generation performed by DH, ECDSA or RSA.		RSA KeyGen (FIPS186-5): (A6648) Modulo: 2048, 3072, 4096, 6144, 8192 ECDSA KeyGen (FIPS186-5): (A6648) Curve: B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Safe Primes Key Generation: (A6648) Safe Prime Groups: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Counter DRBG:

Name	Type	Description	Properties	Algorithms
				(A6648) Mode: AES-128, AES-192, AES-256 Hash DRBG: (A6648) Mode: SHA-1, SHA2-256, SHA2-512, SHA3-256, SHA3-512 HMAC DRBG: (A6648) Mode: SHA-1, SHA2-256, SHA2-512, SHA3-256, SHA3-512 CKG: () Key Type: Symmetric and Asymmetric
Asymmetric Key Verification	AsymKeyPair-KeyVer	Asymmetric Key Pair Verification for ECDSA or DH		ECDSA KeyVer (FIPS186-5): (A6648) Curve: B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Safe Primes Key Verification: (A6648) Safe Prime Groups: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192
Authenticated Decryption	BC-Auth	Block Cipher Symmetric		AES-CCM: (A6648) Key Length: 128,

Name	Type	Description	Properties	Algorithms
		Decryption Authenticated		192, 256 AES-GCM: (A6648) Key Length: 128, 192, 256
Authenticated Encryption	BC-Auth	Block Cipher Symmetric Encryption Authenticated		AES-CCM: (A6648) Key Length: 128, 192, 256 AES-GCM: (A6648) Key Length: 128, 192, 256
Decryption	BC-UnAuth	Block Cipher Symmetric Decryption Non-Authenticated		AES-CBC: (A6648) Key Length: 128, 192, 256 AES-CBC-CS1: (A6648) Key Length: 128, 192, 256 AES-CBC-CS2: (A6648) Key Length: 128, 192, 256 AES-CBC-CS3: (A6648) Key Length: 128, 192, 256 AES-CFB1: (A6648) Key Length: 128, 192, 256 AES-CFB128: (A6648) Key Length: 128, 192, 256 AES-CFB8: (A6648) Key Length: 128, 192, 256 AES-CTR: (A6648) Key Length: 128, 192, 256 AES-ECB: (A6648) Key Length: 128, 192, 256 AES-OFB: (A6648) Key Length: 128,

Name	Type	Description	Properties	Algorithms
				192, 256 AES-XTS Testing Revision 2.0: (A6648) Key Length: 128, 256
Digital Signature Generation	DigSig-SigGen	Digital Signature Generation using ECDSA or RSA		ECDSA SigGen (FIPS186-5): (A6648) Curve: B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Hash Algorithm: SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 RSA SigGen (FIPS186-5): (A6648) Signature Type: PKCS 1.5 and PSS Modulo: 2048, 3072, 4096 Hash Algorithm: SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 SHA2-224: (A6648) SHA2-256: (A6648) SHA2-384: (A6648)

Name	Type	Description	Properties	Algorithms
				SHA2-512: (A6648) SHA2-512/224: (A6648) SHA2-512/256: (A6648) SHA3-224: (A6648) SHA3-256: (A6648) SHA3-384: (A6648) SHA3-512: (A6648)
Digital Signature Verification	DigSig-SigVer	Digital Signature Verification using ECDSA or RSA		ECDSA SigVer (FIPS186-5): (A6648) Curve: B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Hash Algorithm: SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 RSA SigVer (FIPS186-5): (A6648) Signature Type: PKCS 1.5 and PSS Modulo: 2048, 3072, 4096 Hash Algorithm: SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256,

Name	Type	Description	Properties	Algorithms
				SHA3-384, SHA3-512 RSA SigVer (FIPS186-4): (A6648) Signature Type: PKCS 1.5 and PSS Modulo: 1024, 2048, 3072, 4096 Hash Algorithm: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 SHA-1: (A6648) SHA2-224: (A6648) SHA2-256: (A6648) SHA2-384: (A6648) SHA2-512: (A6648) SHA2-512/224: (A6648) SHA2-512/256: (A6648) SHA3-224: (A6648) SHA3-256: (A6648) SHA3-384: (A6648) SHA3-512: (A6648)
ECC CDH Primitive	KAS-SSC	ECC CDH Primitive in Shared Secret Computation		KAS-ECC CDH-Component SP800-56Ar3: (A6648)
Encryption	BC-UnAuth	Block Cipher Symmetric Encryption Non-Authenticated		AES-CBC: (A6648) Key Length: 128, 192, 256 AES-CBC-CS1: (A6648)

Name	Type	Description	Properties	Algorithms
				Key Length: 128, 192, 256 AES-CBC-CS2: (A6648) Key Length: 128, 192, 256 AES-CBC-CS3: (A6648) Key Length: 128, 192, 256 AES-CFB1: (A6648) Key Length: 128, 192, 256 AES-CFB128: (A6648) Key Length: 128, 192, 256 AES-CFB8: (A6648) Key Length: 128, 192, 256 AES-CTR: (A6648) Key Length: 128, 192, 256 AES-ECB: (A6648) Key Length: 128, 192, 256 AES-OFB: (A6648) Key Length: 128, 192, 256 AES-XTS Testing Revision 2.0: (A6648) Key Length: 128, 192, 256
KAS-ECC	KAS-SSC	Uses the KAS_ECC_SSC shared secret computation which is then fed into one of the module's SP 800-135 compliant KDFs (TLS 1.2 and 1.3, SSHv2, ANSI	Reference:IG D.F scenario 2, path (1), no key confirmation, key derivation per IG 2.4.B Caveat:Key establishment methodology provides between	KAS-ECC-SSC Sp800-56Ar3: (A6648) Curves: B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 TLS v1.2 KDF

Name	Type	Description	Properties	Algorithms
		X9.63-2001 and ANSI X9.42-2001) to derive keys for their respective industry standard protocols	112 and 256 encryption strength	RFC7627: (A6648) Hash Algorithm: SHA2-256, SHA2-384, SHA2-512 TLS v1.3 KDF: (A6648) HMAC Algorithm: SHA2-256, SHA2-384 KDF ANS 9.42: (A6648) KDF ANS 9.63: (A6648) KDF SSH: (A6648)
KAS-FFC	KAS-SSC	Uses the KAS_FFC_SSC shared secret computation which is then fed into one of the module's SP 800-135 compliant KDFs (TLS 1.2 and 1.3, SSHv2, ANSI X9.63-2001 and ANSI X9.42-2001) to derive keys for their respective industry standard protocols	Reference:IG D.F scenario 2, path (1), no key confirmation, key derivation per IG 2.4.B Caveat:Key establishment methodology provides between 112 and 200 bits of encryption strength	KAS-FFC-SSC Sp800-56Ar3: (A6648) Domain Parameter Generation Methods: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 TLS v1.2 KDF RFC7627: (A6648) Hash Algorithm: SHA2-256, SHA2-384, SHA2-512 TLS v1.3 KDF: (A6648) HMAC Algorithm: SHA2-256, SHA2-384 KDF SSH: (A6648) KDF ANS 9.42: (A6648)

Name	Type	Description	Properties	Algorithms
				KDF ANS 9.63: (A6648)
KBKDF	KBKDF	Key Based Derivation Function		KDF SP800-108: (A6648) KDF Mode: Counter, Feedback MAC Mode: CMAC-AES128, CMAC-AES192, CMAC-AES256, HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA2-512/224, HMAC-SHA2-512/256, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512
KDF-TLS	KAS-135KDF	Key derivation for TLS		TLS v1.2 KDF RFC7627: (A6648) Hash Algorithm: SHA2-256, SHA2-384, SHA2-512 TLS v1.3 KDF: (A6648) HMAC Algorithm: SHA2-256, SHA2-384
Key Derivation SP800-135	KAS-135KDF	SP 800-135 Key Derivation		KDF ANS 9.42: (A6648) KDF ANS 9.63: (A6648) KDF SSH: (A6648)
Key Derivation 56Crev2	KAS-56CKDF	Key Derivation using SP 800-56Crev2		KDA HKDF SP800-56Cr2: (A6648) HMAC Algorithm: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512,

Name	Type	Description	Properties	Algorithms
				SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 KDA OneStep SP800-56Cr2: (A6648) Auxiliary Function: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512, HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA2-512/224, HMAC-SHA2-512/256, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512, KMAC-128, KMAC-256
MAC1	MAC	Message Authentication Computation with AES CMAC or AES GMAC		AES-CMAC: (A6648) Key Length: 128, 192, 256 AES-GMAC: (A6648) Key Length: 128, 192, 256
MAC2	MAC	Message Authentication Computation with HMAC or KMAC		HMAC-SHA-1: (A6648) HMAC-SHA2-224: (A6648) HMAC-SHA2-256: (A6648) HMAC-SHA2-384:

Name	Type	Description	Properties	Algorithms
				(A6648) HMAC-SHA2-512: (A6648) HMAC-SHA2-512/224: (A6648) HMAC-SHA2-512/256: (A6648) HMAC-SHA3-224: (A6648) HMAC-SHA3-256: (A6648) HMAC-SHA3-384: (A6648) HMAC-SHA3-512: (A6648) KMAC-128: (A6648) KMAC-256: (A6648)
Message Digest	SHA XOF	Secure Hash		SHA-1: (A6648) SHA2-224: (A6648) SHA2-256: (A6648) SHA2-384: (A6648) SHA2-512: (A6648) SHA2-512/224: (A6648) SHA2-512/256: (A6648) SHA3-224: (A6648) SHA3-256: (A6648) SHA3-384: (A6648) SHA3-512: (A6648) SHAKE-128: (A6648) SHAKE-256: (A6648)

Name	Type	Description	Properties	Algorithms
PBKDF2	PBKDF	PBKDF2 Key Derivation		PBKDF: (A6648) HMAC Algorithm: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Iteration Count: 1-10000000 Increment 1 Salt Length: 128-4096 Increment 8
Random Number Generation	DRBG	Generation of random numbers		Counter DRBG: (A6648) Mode: AES-128, AES-192, AES-256 Hash DRBG: (A6648) Mode: SHA-1, SHA2-256, SHA2-512, SHA3-256, SHA3-512 HMAC DRBG: (A6648) Mode: SHA-1, SHA2-256, SHA2-512, SHA3-256, SHA3-512
CKG	CKG	Direct output of DRBGs may be used for symmetric key generation per SP 800-133r2.		CKG: () Counter DRBG: (A6648) Mode: AES-128, AES-192, AES-256 Hash DRBG: (A6648) Mode: SHA-1, SHA2-256, SHA2-512, SHA3-256, SHA3-512 HMAC DRBG: (A6648) Mode: SHA-1,

Name	Type	Description	Properties	Algorithms
				SHA2-256, SHA2-512, SHA3-256, SHA3-512
Key Wrapping	BC-AuthDecrypt BC-AuthEncrypt	Key wrapping using AES KW or AES KWP		AES-KW: (A6648) Key Length: 128, 192, 256 AES-KWP: (A6648) Key Length: 128, 192, 256
Key Encapsulation	AsymKeyPair-Decap AsymKeyPair-Encap	Key Encapsulation using RSA		KTS-IFC: (A6648) Moduli: 2048, 3072, 4096, 6144

Table 9: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 AES GCM Usage

The Module supports internal AES GCM IV generation compliant to [IG] C.H *Key/IV Pair Uniqueness Requirements* from [SP800-38D] Scenario 1(a), tested per option (ii) under C.H TLS/DTLS 1.2 protocol IV generation, and Scenario 5 TLS 1.3 per [RFC8446].

The Module does not implement the TLS protocol itself; however, it provides the cryptographic functions required for implementing the protocols. AES GCM encryption is used in the context of the TLS protocol versions 1.2 and 1.3. For TLS v1.2, the mechanism for IV generation is compliant with [RFC5288]. The module provides the primitives to support the AES GCM ciphersuites from [SP800-52r2] Section 3.3.1. The counter portion of the IV is strictly increasing. When the IV exhausts the maximum number of possible values for a given session key, this results in a failure in encryption and a handshake to establish a new encryption key will be required. It is the responsibility of the user of the module, i.e., the first party, client or server, to encounter this condition, to trigger this handshake in accordance with [RFC5246]. For TLS v1.3, the mechanism for IV generation is compliant with [RFC8446].

The module also supports internal IV generation using the module's approved DRBG. The IV is at least 96-bits in length per [SP800-38D], Section 8.2.1. Per [IG] C.H Scenario 2 and [SP800-38D], the approved DRBG generates outputs such that the (key, IV) pair collision probability is less than 2^{-32}.

In each case, in the event that the module power is lost and restored, the user must ensure that the AES GCM encryption/decryption keys are re-distributed. The module does not support persistent storage of SSPs.

The Module also supports importing of GCM IVs when an IV is not generated within the Module. In the approved mode, an IV must not be imported for encryption from outside the cryptographic boundary of the Module as this will result in a non-conformance.

2.7.2 AES-XTS

The AES algorithm in XTS mode shall only be used for confidentiality on storage devices, as specified in [SP800-38E]. The length of a single data unit encrypted with the AES-XTS shall not exceed 220 AES blocks that is 16 MB of data.

The module implements a check to ensure that the two AES keys used in AES-XTS algorithm are not identical. This check is performed before using the keys in the AES-XTS algorithm to process data with them.

The user is responsible for generating both keys used for the AES-XTS independently, as required by the [IG] C.I.

Based on the previous information, the module AES XTS implementation is compliant with [IG] C.I.

2.7.3 SHA-3 Family

The module is compliant to the [IG] C.C regarding the SHA-3 family algorithms.

All the SHA-3 functions have been tested and validated with the CAVP tool (#A6648), as it is indicated in Table 7. In addition, every higher-level algorithm that uses a SHA-3 family algorithm, are also validated with the CAVP, together in the same CAVP certificate (#A6648).

2.7.4 RSA Digital Signature

The module provides different algorithms for digital signature. Among them is the RSA [FIPS186-5], which is compliant with [IG] C.F.

RSA digital signature generation can be performed with 2048-, 3072- or 4096- bits modulus length. As it is indicated in Table 7, all the RSA signature algorithm implementations are tested with the CAVP (Cert. #A6648). In the CAVP certificate, it is indicated that the modulus used by the RSA signature generation are 2048, 3072 and 4096. Therefore, all different modulus length used by the RSA signature generation are tested by the CAVP.

For the signature verification, the module has tested and validated all the different RSA [FIPS186-5] modulus length implemented by the module: 2048, 3072 and 4096, in the CAVP certificate Cert. #A6648, and RSA [FIPS186-4] signature verification with modulus length: 1024, 2048, 3072 and 4096, also in CAVP certificate Cert. #A6648.

2.7.5 SHA-1 Usage:

SHA-1 is deprecated through December 31, 2030, for non-digital signature applications. After December 31, 2030, any use of SHA-1 will be disallowed.

2.7.6 Legacy Use:

The module supports the following implementation for legacy use:

- RSA FIPS 186-4 (modulus 1024 bits) digital signature verification providing less than 112 bits of security strength per IG C.K.
- RSA FIPS 186-4 digital signature verification with SHA-1 used as the underlying hash algorithm per IG C.K.

2.8 RBG and Entropy

The module employs a Deterministic Random Bit Generator (DRBG) for the creation of cryptographic key material. The module contains the following Random Bit Generator (DRBG) compliant with the [SP800-90Arev1]:

- CTR-DRBG (with and without Derivation Function)
- HASH-DRBG
- HMAC-DRBG

The module does not implement or actively call Non-Deterministic Random Bit Generator or entropy source. The module receives entropy passively and uses 128, 192 or 256 bits of entropy to seed the DRBG. Full entropy source must be used for the default DRBG.

The calling application is responsible for use of a [SP800-90B] compliant entropy source with at least 256 bits of security strength. Entropy is supplied to the Module via callback functions. The callback functions shall return an error if the minimum entropy strength cannot be met.

2.9 Key Generation

For generation of RSA and ECDSA key pairs, the module implements approved key generation services compliant with [FIPS 186-5] where the key material is directly obtained from approved [SP 800-90Arev1] DRBGs according to the [SP 800-133rev2].

The public and private key pairs used in the Diffie-Hellman and EC Diffie-Hellman KAS are generated internally. They are compliant with NIST [SP 800-56Arev3].

Cryptographic Key Generation: the module uses an Approved Hash, CTR or/and HMAC DRBG specified in [SP800-90Arev1] to generate random cryptographic material. The resulting generated material are unmodified outputs from the DRBG.

According to FIPS 140-3 Implementation Guidance [IG] D.H. a component key generation (CKG) using the unmodified output of an approved DRBG can be used to generate cryptographic material for:

- Direct Generation of symmetric keys per section 6.1 of the SP 800-133rev2.
- Derivation of symmetric keys per section 6.2 of the SP 800-133rev2.

During the SSP generation, services are not available, and input and output are inhibited.

2.10 Key Establishment

For the key establishment, the module provides different methods of key agreement, key derivation and key transport.

2.10.1 Key Agreement

The module provides Diffie-Hellman, EC Diffie-Hellman key agreement shared secret computation to obtain “shared secret” values.

The security strength of the preceding algorithms is as follows:

1. Diffie-Hellman key agreement provides between 112 and 200 bits of encryption strength.
2. EC Diffie-Hellman key agreement provides between 112 and 256 bits of encryption strength.

Diffie-Hellman and EC Diffie-Hellman are under scenario 2, path 1 of [IG] D.F.

Moreover, the module provides SSP derivation and SSP transport methods which are described in the following sections.

2.10.2 Key Derivation

The module provides the following SSP Derivation methods:

- [SP800-108rev1] Key-Based Key Derivation (KBKDF algorithm).
- Password-Based Key Derivation (PBKDF2 algorithm) based on [SP800-132] option 1a.
- Protocol-Suite Key Derivation: TLS v1.2 KDF, TLS v1.3 KDF, ANSI X9.42, ANSI X9.63 and SSHv2 KDF as key derivation functions.
- Key Derivation based on SP 800-56Crev2: HKDF and KDA.

For the protocol-suite key derivation functions TLS v1.2 KDF (CVL), TLS v1.3 KDF (CVL), ANSI X9.42 (CVL), ANSI X9.63 (CVL) and SSHv2 KDF (CVL); they shall only be used within the context of the respective protocol.

Regarding PBKDF2, in line with the requirements for [SP800-132], keys generated using the approved PBKDF2 must only be used for storage applications. Any other use of the approved PBKDF2 is non-conformant. The security strength of the derived key is at least 112 bits.

As the module is a general-purpose software module, it is not possible to anticipate all the levels of use for the PBKDF2, however a user of the module should also note that a password should at least contain enough security strength to be unguessable and also contain enough strength to reflect the security strength required for the key being generated. The supported lengths of a password/passphrase used can range between 8 and 128 bits.

For the iteration count, as the functionality of the module relies on the usage that a user performs with the module, it is recommended a minimum of 1.000 iterations. The iteration count values used range from 1 to 10000 per [SP800-132] Section 5.2 whereby the iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users.

In addition, users are referred to Appendix A, “Security Considerations” in [SP800-132] for further information on password, salt, and iteration count selection

2.10.3 Key Transport

The module does not establish SSPs using an approved key transport scheme (KTS). However, it does offer approved authenticated algorithms that can be used by an external operator/application as part of an approved KTS. The key transport methods are provided by:

1. Key wrapping. The module implements three different options:

   a. Using an approved key wrapping algorithm (e.g., AES in KW/KWP mode).
   b. An approved authenticated symmetric encryption mode (for example, AES GCM, AES CCM).
   c. An approved symmetric encryption mode (e.g., AES ECB/CBC ...) together with an approved authentication method (e.g., HMAC or AES CMAC).
2. Key Encapsulation: For this method, the module implements an approved RSA-based key transport scheme (KTS-RSA based on SP 800-56Brev2).

According to Table 2: Comparable strengths in [SP800-57 Part1 Rev5], the key sizes of AES and RSA provides the following security strength:

- AES key wrapping provides between 128 and 256 bits of encryption strength.
- Approved authenticated encryption mode (AES-GCM, AES-CCM) provides between 128 and 256 bits of encryption strength.
- Combination of any approved AES encryption mode with HMAC/AES CMAC authentication provides between 128 and 256 bits of encryption strength.
- RSA key encapsulation provides between 112 and 176 bits of encryption strength.

2.11 Industry Protocols

The module implements TLS versions 1.2 and 1.3. The AES GCM supported ciphersuites for each version are listed below:

Supported AES GCM ciphersuites for TLS v1.2:

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

Supported AES GCM ciphersuites for TLS v1.3:

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384

No parts of the TLS v1.2/v1.3 protocol, other than the approved cryptographic algorithms and the KDFs, have been tested by the CAVP or CMVP.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	API input parameters for data
N/A	Data Output	API output parameters for data
N/A	Control Input	API function calls, API input parameters for control
N/A	Status Output	API return codes

Table 10: Ports and Interfaces

As a software module, it does not have physical ports. For the purpose of the FIPS 140-3 validation, the module interfaces are defined as Software or Firmware Module Interfaces (SFMI), and the physical ports are interpreted to be physical ports of the hardware platform on which the module runs.

The logical interface is a C language Application Program Interface (API) through which calling application request services. The interfaces are mapped to the API provided by the module, through which the operator can interact. The interfaces are listed in the table above.

All data output via data output interface is inhibited under the following circumstances:

- When the module performing pre-operational and conditional self-tests.
- During zeroisation of data such as CSPs.
- When the module enters error state.

The module does not implement either control output interface or power interface (it is not required since the cryptographic module is a software module).

4 Roles, Services, and Authentication

4.1 Authentication Methods

N/A for this module.

The module does not support authentication mechanisms.

4.2 Roles

The Module only supports the Cryptographic Officer (CO) role (implicitly identified). The module does not allow concurrent operators. The module does not support a maintenance role or bypass capability.

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 11: Roles

4.3 Approved Services

All services implemented by the module are listed in the tables below. The approved services are shown in Table 12. Please note that the Sensitive Security Parameters (SSPs) listed below indicate the type of access required using the following notation:

- G – Generate: The SSP is generated or derived.
- R – Read: The SSP is read from the module.
- W – Write: The SSP is updated, imported, or written to the module.
- E – Execute: The SSP is used within an Approved security function.
- Z – Zeroize: The SSP is zeroized.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Initialization	Perform the initialization of the module	None	Module Default Entry Point	None	None	Crypto Officer
Show Status	Provide module status	None	API call parameters	Current Status	None	Crypto Officer
Show Version	Display the module name and version	None	API call parameters	Module Name and Version	None	Crypto Officer
Self-test	Perform pre-operational and conditional self-tests	FIPS_OK	Power cycle and/or API call parameters	Status	Authenticated Decryption Authenticated Encryption Decryption Digital Signature Generation Digital Signature Verification Encryption KAS-ECC KAS-FFC KBKDF KDF-TLS Key Derivation SP800-135 Key Derivation 56Crev2 MAC1 MAC2 Message Digest	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					PBKDF2 Random Number Generation Key Encapsulation	
Asymmetric Key Generation and Verification	Generate asymmetric key pairs (RSA, ECDSA, DH, ECDH)	FIPS_OK	API call parameters: Curve, modulus, group, key length. Key pair to be verified	Status, Generated private and public key pair	Asymmetric Key Generation Asymmetric Key Verification CKG	Crypto Officer - ECDSA Public Key: G,R,E - ECDSA Private Key: G,R,E - Diffie-Hellman Public Key: G,R,E - Diffie-Hellman Private Key: G,R,E - ECDH Public Key: G,R,E - ECDH Private Key: G,R,E - RSA Public Key: G,R,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- RSA Private Key: G,R,E
Digital Signature	Generate or verify RSA and ECDSA digital signatures.	FIPS_OK	API call parameters: key pair, message, signature (for verification)	Status, signature	Digital Signature Generation Digital Signature Verification	Crypto Officer - ECDSA Public Key: W,E - ECDSA Private Key: W,E - RSA Public Key: W,E - RSA Private Key: W,E
Key Agreement	Perform key agreement primitives on behalf of the calling application (does not establish keys into the module)	FIPS_OK	API call parameters: private key, counter public key	Status, key components, key	ECC CDH Primitive KAS-ECC KAS-FFC	Crypto Officer - Diffie-Hellman Public Key: W,E - Diffie-Hellman Private Key: W,E - ECDH Public Key: W,E - ECDH

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Private Key: W,E - Shared Secret: G
Key Derivation	Derive keying material using KBKDF, PBKDF, HKDF, SP800-56Crev2 One-Step KDF (KDA), SP800-135rev1 TLS 1.2, SSHv2, ANSI X9.42-2001, ANSI X9.63-2001 KDFs and TLS 1.3 KDF.	FIPS_OK	API call parameters: Shared Secret, additional info depending on the algorithm used.	Status and derived keying material	CKG KBKDF KDF-TLS Key Derivation SP800-135 Key Derivation 56Crev2 PBKDF2	Crypto Officer - Shared Secret: W,E - Keying material: G - PBKDF2 password: W,E - PBKDF2 salt: W,E
Key Encapsulation	Encrypt or Decrypt a key value on behalf of the calling application (does not establish keys into the module)	FIPS_OK	API call parameters: Key-encryption key, Key to be encapsulated.	Status and Encapsulated key	Key Encapsulation	Crypto Officer - RSA Private Key: W,E - RSA Public Key: W,E
Key Wrapping	Encrypt or decrypt a key value on behalf of the calling application (does not establish keys into the module)	FIPS_OK	API call parameters: Key-encryption key, Key to be wrapped	Status and Wrapped key	Key Wrapping	Crypto Officer - AES Keys: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Keyed Hash	Generate or verify data integrity with HMAC/KMAC.	FIPS_OK	API call parameters: HMAC/KMAC key, message, keyed hash value (for verification)	Status, Keyed Hash value (Generation), True or False (Verification)	MAC2	Crypto Officer - HMAC Key: W,E - KMAC Key: W,E
Message Authentication Code	Generation or verify data integrity with CMAC and GMAC	FIPS_OK	API call parameters: key, data, authenticated message digest to verify.	Status, authenticated message digest.	MAC1	Crypto Officer - AES Keys: W,E
Message Digest	Compute and return a message digest using SHS and SHA-3 algorithms	FIPS_OK	API call parameters: message	Status, hash value	Message Digest	Crypto Officer
Random Number Generation	Used for random number and symmetric key generation using HMAC, HASH or CTR DRBG	FIPS_OK	API call parameters: number of bits to be generated.	Status and random bitstring	Random Number Generation CKG	Crypto Officer - DRBG Entropy Input: W,E,Z - DRBG Seed: G,E,Z - DRBG C Value: G,E - DRBG V Value: G,E - DRBG Key Value: G,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Symmetric Encryption/Decryption	Encrypt/Decrypt plaintext/ciphertext using supplied key	FIPS_OK	API call parameters: key, IV, plaintext/ciphertext	Status, ciphertext/plaintext	Encryption Decryption Authenticated Encryption Authenticated Decryption	Crypto Officer - AES Keys: W,E - AES IV: W,E - AES XTS Key: W,E - AES XTS IV: W,E
Zeroization	Zeroize and deallocate memory containing sensitive data	None	Memory pointer	None	None	Crypto Officer - AES Keys: Z - AES IV: Z - AES XTS Key: Z - AES XTS IV: Z - Diffie-Hellman Public Key: Z - Diffie-Hellman Private Key: Z - DRBG Entropy Input: Z - DRBG

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Seed: Z - DRBG C Value: Z - DRBG V Value: Z - DRBG Key Value: Z - ECDH Public Key: Z - ECDH Private Key: Z - ECDSA Public Key: Z - ECDSA Private Key: Z - HMAC Key: Z - KMAC Key: Z - RSA Public Key: Z - RSA Private Key: Z - Shared Secret: Z - HKDF

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						salt: Z - Keying material: Z - PBKDF2 password: Z - PBKDF2 salt: Z

Table 12: Approved Services

Service Indicator

Regarding the Indicator of approved security services, the Module conforms to [IG] 2.4.C Approved Security Service Indicator, similar to example 2. Each service provides context sensitive status responses as described in the OpenSSL 3 API manual pages; generally, functions of return type int return the value 1 (FIPS_OK) for success with other error codes as appropriate for the call.

4.4 Non-Approved Services

N/A for this module.

The module does not provide any non-approved services.

4.5 External Software/Firmware Loaded

The module does not implement a software/firmware loading capability.

5 Software/Firmware Security

5.1 Integrity Techniques

The cryptographic module is composed of a software shared library as well as the file containing the pre-computed HMAC-SHA2-256 value.

The module uses HMAC-SHA2-256 as the approved integrity technique. Before the integrity test, the module performs the HMAC-SHA2-256 self-test. At the compilation and build time performed by the vendor, HMAC-SHA2-256 is used to generate a MAC value, which is stored in /data/opt/pme/conf/openssl/fipsmodule.cnf.

Every time the module starts, it loads the module and the HMAC-SHA2-256 is used to recalculate the MAC value of the current running binary file of the module. If it matches, the module integrity is successful, and it starts normally. If the MAC value does not match, the module library exits with error. If failure occurs during self-test, all crypto functionality is disabled.

5.2 Initiate on Demand

The module also provides on-demand integrity test. The integrity test is performed by the Self-Test On demand service by invoking the SELF_TEST_post() function or reloading the cryptographic module. This test is performed as part of the pre-operational self-tests as well.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

The Module is a software module. It is operated in a modifiable operational environment per FIPS 140-3 level 1 specifications. The module executes within the BMC operating system as an independent process. The BMC operational system segregates processes into separate process spaces. Other processes and operators cannot control the module's process and storage areas.

All SSPs are under the control of the OS, which protects its CSPs against unauthorized disclosure, modification, and substitution and PSPs against unauthorized modification and substitution. Additionally, the OS provides dedicated process space to each executing process, and the module operates entirely within the calling application's process space. The module only allows access to SSPs through its well-defined API. The module does not have the ability of spawning new processes.

The operating system is restricted to a single operator. Concurrent operators are explicitly excluded.

7 Physical Security

The module is a software module; therefore, this section does not apply.

8 Non-Invasive Security

The module does not implement any non-invasive attack mitigation technique to protect itself and the module's unprotected SSPs from non-invasive attacks.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
RAM	System Memory	Dynamic

Table 13: Storage Areas

SSPs are provided to the module by the calling process and are destroyed when released by the appropriate zeroisation function calls. The module does not perform persistent storage of SSPs.

No physical storage is offered within the cryptographic boundary, and therefore the module does not store any SSPs persistently beyond the lifetime of the API call. Any persistent key storage occurs outside the module's cryptographic boundary but within the physical perimeter and the management of these keys is responsibility of the calling application.

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
SSP Input	App via TOEPP path	RAM	Plaintext	Manual	Electronic	
SSP Output	RAM	App via TOEPP path	Plaintext	Manual	Electronic	

Table 14: SSP Input-Output Methods

The keys and SSPs to be entered or exited are provided to the module via API input/output parameters in plaintext form and output via API output parameters in plaintext form.

This is allowed per section 7.9.5 of the [ISO19790] since all CSPs or key components are maintained within the environment and the requirements from section 7.6.3 are met.

The module does not support either manual key entry or intermediate key generation values.

The module does not output intermediate key generation values.

Additionally, the module implements two independent internal actions in order to prevent the inadvertent output of any plaintext SSP. The mechanism implemented by the module is described below:

1. Memory allocation of the necessary context to request the service.
2. Process the service request which outputs CSPs using the created context.

When these two actions are completed without errors, the Key/SSP exits the module in plaintext.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
API call	The zeroization functions overwrite the memory occupied by SSP with 'zeros' and deallocate the memory with the regular memory deallocation operating system call.	The zeroization of the SSPs starts just after the invocation of the zeroization command. Once invoked, these techniques take effect immediately and do not allow sufficient time to compromise any plaintext secret, private keys and CSPs.	By invocation through API call
Power Cycle	The zeroisation is performed by erasing the memory location occupied by the SSP and further deallocating that area. In case of abnormal termination, or swap in/out of a memory page of a process, the keys in memory are overwritten by the Linux kernel before the memory is allocated to another process.	The keys in memory are overwritten by the Linux kernel before the memory is allocated to another process.	Restart the module

Table 15: SSP Zeroization Methods

SSP zeroisation functions are implemented in the module and they depend on the SSP to be zeroized. The zeroisation of the SSPs starts just after the invocation of the zeroisation command. Once invoked, these techniques take immediate effect and do not allow sufficient time to compromise any plaintext secret, private keys or CSPs.

The zeroisation functions overwrite the memory occupied by the SSP with ‘zeros’ and deallocate the memory with the regular memory deallocation operating system call. In case of abnormal termination or swap in/out of a memory page of a process, the keys in memory are overwritten by the Linux kernel before the memory is allocated to another process.

During the zeroisation process, services are not available, and input and output interfaces are inhibited.

9.4 SSPs

The following section includes all the information related to the Sensitive Security Parameters (SSPs) and its management. The tables below identify all the SSPs handled by the cryptographic module as well as their purpose, generation method, input and output methods, where they are stored and how they are zeroised.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES Keys	AES Keys	128, 192, 256 - 128, 192, 256	Symmetric Keys - CSP			Authenticated Decryption Authenticated Encryption Decryption

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
						Encryption MAC1 Key Wrapping
AES IV	AES Initialization Vector	128 - 128	Initialization Vectors - CSP			Authenticated Decryption Authenticated Encryption Decryption Encryption MAC1
AES XTS Key	AES Keys	128, 256 - 128, 256	Symmetric Keys - CSP			Encryption Decryption
AES XTS IV	AES Initialization Vector	128 - 128	Initialization Vectors - CSP			Encryption Decryption
Diffie-Hellman Public Key	Diffie-Hellman Public Key	From 2048-bit to 8192-bit - 112, 128, 152 and 200 bits	Asymmetric Key - PSP	Asymmetric Key Generation		KAS-FFC
Diffie-Hellman Private Key	Diffie-Hellman Private Key	From 2048-bit to 8192-bit - 112, 128, 152, 200 bits	Asymmetric Key - CSP	Asymmetric Key Generation		KAS-FFC
DRBG Entropy Input	Entropy material for DRBG	More than 128 bits - More than 256 bits	DRBG material - CSP			Random Number Generation
DRBG Seed	Seeding material for DRBG	256 - 256	DRBG material - CSP			Random Number Generation
DRBG C Value	Used for DRBG	Internal state value - Internal state value	DRBG material - CSP	Hash DRBG (A6648)		Random Number Generation
DRBG V Value	Used for DRBG	Internal state value -	DRBG material - CSP	Random Number Generation		Random Number Generation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		Internal state value				
DRBG Key Value	Used for DRBG	256 - 256	DRBG material - CSP	Counter DRBG (A6648) HMAC DRBG (A6648)		Random Number Generation
ECDH Public Key	ECDH Public Key	From 224 to 576 bits - 128-256 bits	Asymmetric key - PSP	Asymmetric Key Generation		KAS-ECC ECC CDH Primitive
ECDH Private Key	ECDH Private Key	From 224 to 576 bits - 128-256 bits	Asymmetric key - CSP	Asymmetric Key Generation		KAS-ECC ECC CDH Primitive
ECDSA Public Key	ECDSA Public Key	From 192 to 576 bits - 192-576 bits	Asymmetric key - PSP	Asymmetric Key Generation		Asymmetric Key Verification Digital Signature Verification
ECDSA Private Key	ECDSA Private Key	From 224 to 576 bits - 224-576 bits	Asymmetric key - CSP	Asymmetric Key Generation		Digital Signature Generation
HKDF salt	HKDF salt	Salt bitstring - Salt bitstring	Bitstring - CSP			Key Derivation 56Crev2
HMAC Key	HMAC Key used for message authentication	112 bits or greater - 112 bits or greater	HMAC Key - CSP			MAC2
Keying material	Keying material derived from key derivation function (SP 800-108rev1 KBKDF, SP 800-132, HKDF,	Keying material bitstring - Keying material bitstring	Bitstring - CSP	Key Derivation SP800-135 Key Derivation 56Crev2 PBKDF2		Key Derivation SP800-135 Key Derivation 56Crev2 PBKDF2

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	PBKDF, KDA, SP 800-135rev1 KDFs, TLS 1.3 KDF).			KBKDF KDF-TLS		KBKDF KDF-TLS
KMAC Key	KMAC Key used for message authentication	128 bits or greater - 128 bits or greater	KMAC Key - CSP			MAC2
PBKDF2 password	PBKDF2 password	From 8 to 128 characters (bytes) with increments of 8 characters - From 8 to 128 characters (bytes) with increments of 8 characters	Password - CSP			PBKDF2
PBKDF2 salt	PBKDF2 salt	From 128 to 4096-bit salt bitstring with increment of 8-bits - From 128 to 4096-bit salt bitstring with increment of 8-bits	Bitstring - CSP			PBKDF2
RSA Public Key	RSA Public Key	2048, 3072, 4096, 6144, 8192-bit - 112, 128, 152, 176, 200 bits	Asymmetric Key - PSP	Asymmetric Key Generation		Digital Signature Verification Key Encapsulation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
RSA Private Key	RSA Private Key	2048, 3072, 4096, 6144, 8192-bit - 112, 128, 152, 176, 200 bits	Asymmetric Key - CSP	Asymmetric Key Generation		Digital Signature Generation Key Encapsulation
Shared Secret	Shared Secret	112 or greater - 112 or greater	Bitstring - CSP		KAS-FFC KAS-ECC	KBKDF KDF-TLS Key Derivation SP800-135 Key Derivation 56Crev2

Table 16: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES Keys	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	AES IV:Used With
AES IV	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	AES Keys:Used With
AES XTS Key	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	AES XTS IV:Used With
AES XTS IV	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	AES XTS Key:Used With
Diffie-Hellman Public Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	Diffie-Hellman Private Key:Paired With
Diffie-Hellman Private Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	Diffie-Hellman Public Key:Paired With
DRBG Entropy Input	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	DRBG Seed:Used With DRBG C Value:Used With DRBG V Value:Used With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
					DRBG Key Value:Used With
DRBG Seed	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	DRBG Entropy Input:Used With DRBG C Value:Used With DRBG V Value:Used With DRBG Key Value:Used With
DRBG C Value		RAM:Plaintext	Ephemeral	API call Power Cycle	DRBG Entropy Input:Used With DRBG Seed:Used With DRBG V Value:Used With
DRBG V Value		RAM:Plaintext	Ephemeral	API call Power Cycle	DRBG Entropy Input:Used With DRBG Seed:Used With DRBG C Value:Used With DRBG Key Value:Used With
DRBG Key Value		RAM:Plaintext	Ephemeral	API call Power Cycle	DRBG Entropy Input:Used With DRBG Seed:Used With DRBG V Value:Used With
ECDH Public Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	ECDH Private Key:Paired With
ECDH Private Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	ECDH Public Key:Paired With
ECDSA Public Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	ECDSA Private Key:Paired With
ECDSA Private Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	ECDSA Public Key:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
HKDF salt	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	Keying material:Used With
HMAC Key	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	
Keying material	SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	HKDF salt:Used With PBKDF2 password:Used With PBKDF2 salt:Used With
KMAC Key	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	
PBKDF2 password	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	PBKDF2 salt:Used With Keying material:Used With
PBKDF2 salt				API call Power Cycle	PBKDF2 password:Used With Keying material:Used With
RSA Public Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	RSA Private Key:Paired With
RSA Private Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	RSA Public Key:Paired With
Shared Secret	SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	Diffie-Hellman Public Key:Derived From Diffie-Hellman Private Key:Derived From ECDH Public Key:Derived From ECDH Private Key:Derived From

Table 17: SSP Table 2

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A6648)	Key length: 256-bit hardcoded key	KAT	SW/FW Integrity	Stdout	Integrity self-test performed at the initialization of the module in order to verify its integrity

Table 18: Pre-Operational Self-Tests

The module performs pre-operational tests automatically when the module is loaded into memory, without operator intervention.

The pre-operational self-tests ensure that the module is not corrupted and that the cryptographic algorithms work as expected. The module transitions to the operational state only after the pre-operational self-tests (and the cryptographic algorithm self-tests, which in this module are executed automatically after the pre-operational self-tests) are passed successfully.

The types of pre-operational self-tests are described in the next sub-section.

Pre-Operational Software Integrity Test

At runtime and prior to checking the module integrity, a Conditional Cryptographic Algorithm Self-Test (CAST) is performed. If the CAST of the HMAC-SHA2-256 is successful, the module computes the HMAC of the module. This value is compared with the one stored in fipsmodule.cnf file, and if they are the same value, the test is passed. Otherwise, the test fails, and the module enters in Critical Error state (Section 10.4).

While the module is executing the pre-operational self-tests, services are not available, and input and output are inhibited. The module is not available for use by the calling application until the pre-operational self-tests are completed successfully.

Pre-Operational Bypass and Critical Functions Tests

The module does not implement pre-operational bypass or critical functions tests. We note that the entropy source is not within the cryptographic boundary of the module, instead passively receiving entropy from the external entropy source. Thus, its critical functions tests are not included in the module.

10.2 Conditional Self-Tests

Conditional self-tests are performed by the cryptographic module when conditions specified for the following tests occurs: Cryptographic Algorithm Self-Tests, Pair-Wise Consistency Test and Critical Function Tests.

The module does not implement any functions requiring a Software/Firmware Load Test, Manual Entry Tests nor Conditional Bypass Test; therefore, these tests are not performed by the module.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-256 (A6648)	128-bit hardcoded key	KAT	CAST	verify_integrity() = 1	Used for module integrity test	Power-up and On-demand
SHA-1 (A6648)	Generation	KAT	CAST	SELF_TEST_kats() = 1	Message Digest Generation	Power-up and On-demand
SHA2-512 (A6648)	Generation	KAT	CAST	SELF_TEST_kats() = 1	Message Digest Generation	Power-up and On-demand
SHA3-256 (A6648)	Generation	KAT	CAST	SELF_TEST_kats() = 1	Message Digest Generation	Power-up and On-demand
AES-ECB Decrypt (Inverse Cipher Function) (A6648)	128-bit key	KAT	CAST	SELF_TEST_kats() = 1	Decrypt	Power-up and On-demand
AES-GCM Authenticated Encrypt (Forward Cipher Function) (A6648)	256-bit key	KAT	CAST	SELF_TEST_kats() = 1	Encrypt	Power-up and On-demand
AES-GCM Authenticated Decrypt (Forward Cipher Function) (A6648)	256-bit key	KAT	CAST	SELF_TEST_kats() = 1	Decrypt	Power-up and On-demand
RSA SigGen (FIPS186-5) (A6648)	2048 bits key size with SHA2-256 and mode PKCS#1v1.5	KAT	CAST	SELF_TEST_kats() = 1	Signature Generation	Power-up and On-demand

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
RSA SigVer (FIPS186-5) (A6648)	2048 bits key size with SHA2-256 and mode PKCS#1v1.5	KAT	CAST	SELF_TEST_kats() = 1	Signature Verification	Power-up and On-demand
ECDSA SigGen P-224 (FIPS186-5) (A6648)	P-224 curve with SHA2-256	KAT	CAST	SELF_TEST_kats() = 1	Signature Generation	Power-up and On-demand
ECDSA SigGen K-233 (FIPS186-5) (A6648)	K-233 curve with SHA2-256	KAT	CAST	SELF_TEST_kats() = 1	Signature Generation	Power-up and On-demand
ECDSA SigVer P-224 (FIPS186-5) (A6648)	P-224 curve with SHA2-256	KAT	CAST	SELF_TEST_kats() = 1	Signature Verification	Power-up and On-demand
ECDSA SigVer K-233 (FIPS186-5) (A6648)	K-233 curve with SHA2-256	KAT	CAST	SELF_TEST_kats() = 1	Signature Verification	Power-up and On-demand
TLS v1.3 KDF (A6648)	TLS v1.3 (Extract) KDF (per Section 7.1 of RFC 8446) with SHA2-256	KAT	CAST	SELF_TEST_kats() = 1	Key Derivation	Power-up and On-demand
TLS v1.2 KDF RFC7627 (A6648)	TLS v1.2 KDF (SHA2-256)	KAT	CAST	SELF_TEST_kats() = 1	Key Derivation	Power-up and On-demand
KDF SSH (A6648)	SSHv2 KDF	KAT	CAST	SELF_TEST_kats() = 1	Key Derivation	Power-up and On-demand

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KDF ANS 9.42 (A6648)	Key Derivation	KAT	CAST	SELF_TEST_kats() = 1	Key Derivation	Power-up and On-demand
KDF ANS 9.63 (A6648)	Key Derivation	KAT	CAST	SELF_TEST_kats() = 1	Key Derivation	Power-up and On-demand
PBKDF (A6648)	PBKDF2	KAT	CAST	SELF_TEST_kats() = 1	Derivation of the Master Key (MK) (per Section 5.3 of SP 800-132)	Power-up and On-demand
KDF SP800-108 (A6648)	KBDKF with Counter Mode (HMAC-SHA2-256)	KAT	CAST	SELF_TEST_kats() = 1	Key Derivation	Power-up and On-demand
KDA HKDF SP800-56Cr2 (A6648)	Key Derivation	KAT	CAST	SELF_TEST_kats() = 1	Key Derivation	Power-up and On-demand
KDA OneStep SP800-56Cr2 (A6648)	One-step KDF	KAT	CAST	SELF_TEST_kats() = 1	Key Derivation	Power-up and On-demand
Hash DRBG (A6648)	SHA2-256	KAT	CAST	SELF_TEST_kats() = 1	Random Bit Generation	Power-up and On-demand
Counter DRBG (A6648)	AES-128 bit key with derivation function	KAT	CAST	SELF_TEST_kats() = 1	Random Bit Generation	Power-up and On-demand
HMAC DRBG (A6648)	SHA-1	KAT	CAST	SELF_TEST_kats() = 1	Random Bit Generation	Power-up and On-demand

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KAS-FFC-SSC Sp800-56Ar3 (A6648)	ffdhe2048 safe prime group with dhEphem scheme	KAT	CAST	SELF_TEST_kats() = 1	Shared Secret Computation (per Section 6 of SP 800-56Arev3)	Power-up and On-demand
KAS-ECC-SSC Sp800-56Ar3 (A6648)	P-256 curve with Ephemeral Unified scheme	KAT	CAST	SELF_TEST_kats() = 1	Shared Secret Computation (per Section 6 of SP 800-56Arev3)	Power-up and On-demand
KTS-IFC Encrypt for Basic (A6648)	2048 bits key size	KAT	CAST	SELF_TEST_kats() = 1	Encrypt for Basic (per IG D.G and SP 800-56Brev2)	Power-up and On-demand
KTS-IFC Decrypt for Basic (A6648)	2048 bits key size	KAT	CAST	SELF_TEST_kats() = 1	Decrypt for Basic (per IG D.G and SP 800-56Brev2)	Power-up and On-demand
KTS-IFC Decrypt for CRT(A6648)	2048 bits key size	KAT	CAST	SELF_TEST_kats() = 1	Decrypt for CRT (per IG D.G and SP 800-56Brev2)	Power-up and On-demand
ECDSA KeyGen (FIPS186-5) (A6648)	Generated curve	PCT	Critical Function	ecdsa_keygen_pairwise_test() = 1	Pairwise Consistency Test on Generation of an ECDSA Key Pair	Key Generation
RSA KeyGen (FIPS186-5) (A6648)	Generated key size	PCT	PCT	rsa_keygen_pairwise_test() = 1	Pairwise Consistency Test on Generation	Key Generation

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
					of an RSA Key Pair.	
KAS-ECC-SSC Sp800-56Ar3 Assurances (A6648)	SP800-56Arev3 assurances	KAT	Critical Function	ossl_ec_key_public_check() = 1	Assurance per Section 5.6.2 of SP 800-56Arev3 required per [IG] D.F	Power-up and On-demand
KAS-FFC-SSC Sp800-56Ar3 Assurances (A6648)	SP800-56Arev3 assurances	KAT	Critical Function	DH_check_pub_key() = 1	Assurances per Section 5.6.2 of SP 800-56Arev3, required per [IG] D.F	Power-up and On-demand
DRBG	DRBG Health Checks	Health Checks	Critical Function	self_test_drbg() = 1	DRBG health checks: DRBG Instantiate, Generate and Reseed Tests	Power-up and On-demand

Table 19: Conditional Self-Tests

In addition to the pre-operational self-tests, the module performs self-tests on Approved cryptographic algorithms supported in the approved mode of operation, using the tests shown in (and indicated as CASTs) and using the provision of IG 10.3.A and IG 10.3.B for optimization of the number of self-tests. These CASTs are performed during the module initialization, prior to the first operational use of each cryptographic algorithm.

Data output through the data output interface is inhibited during the self-tests.

The cryptographic algorithm self-tests are performed in the form of Known Answer Tests (KATs), in which the calculated output is compared with the expected known answer (that are hardcoded in the module). A failed match causes a failure of the self-test. If any of these self-tests fails, the module transitions to error state and is aborted.

The SELF_TEST_kats() API function is in charge of performing all the KAT self-tests for each algorithm, without any operator intervention. This function is called from the SELF_TEST_post() API function, called during the initialization of the module. The API function returns a '1' if all pre-operational and KAT self-tests succeed, and a '0' otherwise.

The Pairwise Conditional Self-tests are run when an asymmetric key pair is generated. In case of failure the module enters in Critical Error state and needs to be reloaded to clear the error. For the DRBG health checks, if any fails, the module will enter in a critical error state.

While the module is executing the conditional self-tests, services are not available, and input and output are inhibited. The module is not available for use by the calling application until these tests are completed successfully. If any of these tests fail, the module enters in Error state.

10.3 Periodic Self-Test Information

Periodic self-tests are performed on demand by the user. The user can follow the instructions in section 10.5, to initiate the periodic self-tests.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A6648)	KAT	SW/FW Integrity	On Demand	Automatic

Table 20: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A6648)	KAT	CAST	On Demand	Programmatically
SHA-1 (A6648)	KAT	CAST	On Demand	Programmatically
SHA2-512 (A6648)	KAT	CAST	On Demand	Programmatically
SHA3-256 (A6648)	KAT	CAST	On Demand	Programmatically
AES-ECB Decrypt (Inverse Cipher Function) (A6648)	KAT	CAST	On Demand	Programmatically
AES-GCM Authenticated Encrypt (Forward Cipher Function) (A6648)	KAT	CAST	On Demand	Programmatically
AES-GCM Authenticated Decrypt (Forward Cipher Function) (A6648)	KAT	CAST	On Demand	Programmatically
RSA SigGen (FIPS186-5) (A6648)	KAT	CAST	On Demand	Programmatically
RSA SigVer (FIPS186-5) (A6648)	KAT	CAST	On Demand	Programmatically

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
ECDSA SigGen P-224 (FIPS186-5) (A6648)	KAT	CAST	On Demand	Programmatically
ECDSA SigGen K-233 (FIPS186-5) (A6648)	KAT	CAST	On Demand	Programmatically
ECDSA SigVer P-224 (FIPS186-5) (A6648)	KAT	CAST	On Demand	Programmatically
ECDSA SigVer K-233 (FIPS186-5) (A6648)	KAT	CAST	On Demand	Programmatically
TLS v1.3 KDF (A6648)	KAT	CAST	On Demand	Programmatically
TLS v1.2 KDF RFC7627 (A6648)	KAT	CAST	On Demand	Programmatically
KDF SSH (A6648)	KAT	CAST	On Demand	Programmatically
KDF ANS 9.42 (A6648)	KAT	CAST	On Demand	Programmatically
KDF ANS 9.63 (A6648)	KAT	CAST	On Demand	Programmatically
PBKDF (A6648)	KAT	CAST	On Demand	Programmatically
KDF SP800-108 (A6648)	KAT	CAST	On Demand	Programmatically
KDA HKDF SP800-56Cr2 (A6648)	KAT	CAST	On Demand	Programmatically
KDA OneStep SP800-56Cr2 (A6648)	KAT	CAST	On Demand	Programmatically
Hash DRBG (A6648)	KAT	CAST	On Demand	Programmatically
Counter DRBG (A6648)	KAT	CAST	On Demand	Programmatically
HMAC DRBG (A6648)	KAT	CAST	On Demand	Programmatically
KAS-FFC-SSC Sp800-56Ar3 (A6648)	KAT	CAST	On Demand	Programmatically

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
KAS-ECC-SSC Sp800-56Ar3 (A6648)	KAT	CAST	On Demand	Programmatically
KTS-IFC Encrypt for Basic (A6648)	KAT	CAST	On Demand	Programmatically
KTS-IFC Decrypt for Basic (A6648)	KAT	CAST	On Demand	Programmatically
KTS-IFC Decrypt for CRT(A6648)	KAT	CAST	On Demand	Programmatically
ECDSA KeyGen (FIPS186-5) (A6648)	PCT	Critical Function	On Demand	Programmatically
RSA KeyGen (FIPS186-5) (A6648)	PCT	PCT	On Demand	Programmatically
KAS-ECC-SSC Sp800-56Ar3 Assurances (A6648)	KAT	Critical Function	On Demand	Programmatically
KAS-FFC-SSC Sp800-56Ar3 Assurances (A6648)	KAT	Critical Function	On Demand	Programmatically
DRBG	Health Checks	Critical Function	On Demand	Programmatically

Table 21: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator

Critical Error	Failure of pre-operational self-tests, cryptographic algorithm self-tests, pairwise consistency tests and critical security function tests.	Failure of pre-operational self-tests, cryptographic algorithm self-tests, pairwise consistency test, critical security functions tests.	Power Cycle	verify_integrity() = 0;
				SELF_TEST_kats() = 0;
				rsa_keygen_pairwise_test() = 0;
				ecdsa_keygen_pairwise_test() = 0;
				ossl_ec_key_public_check() = 0;
				DH_check_pub_key() = 0;
				self_test_drbg ()= 0

Name	Description	Conditions	Recovery Method	Indicator
Soft Error	AES XTS check failure.	AES XTS check failure (Key1 == Key2).	The module clears the error automatically	aes_xts_check_keys_differ() = 0

Table 22: Error States

If any of the self-tests described in sections 10.1, 10.2 and 10.3 fail, the module enters in Critical Error state and needs to be reloaded to exercise any cryptographic service. In the Critical Error State, no cryptographic services are provided, and data output is prohibited. In this state, an internal flag is set to prevent subsequent invocation of any cryptographic calls.

The only method to recover from the Critical Error state is to power cycle the device which results in the module being reloaded into memory and performing the pre-operational software integrity test and the Conditional CASTs. The module will only enter the operational state after successfully passing the pre-operational software integrity test and the Conditional CASTs.

In addition, if the AES XTS check fails during the CSP entry, the module will flow to Soft Error state, not allowing any cryptographic service and no data output or input until the error is cleared.

The table below shows the different causes that lead to the error states and the status indicators reported.

10.5 Operator Initiation of Self-Tests

Pre-operational self-tests and Conditional Cryptographic Algorithms self-test performed by the module during its initialization, are available on demand by resetting the module or by calling the SELF_TEST_post() API function. During the execution of the on-demand self-tests, services are not available, and no data output or input is possible

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

The module is pre-installed and configured. When the module starts, complete the integrity check and startup self-check. After there is no abnormality, the module startups successfully and provides cryptographic function calls for other functional modules of the BMC.

The approved mode is enabled by default. There are no additional installation, configuration, or usage instructions for operators intending to use the Module.

11.2 Administrator Guidance

The Cryptographic Officer and Administrator shall use the provided BMC User Guide. The module always operates in the Approved mode. The module shall be operated using the approved services, with their corresponding approved and allowed cryptographic algorithms provided in this Security Policy (see section 4.3). Key size must comply with [SP800-131Arev2].

There are no additional installation, configuration, or usage instructions for operators intending to use the xFusion Cryptographic Library.

For ensure the correct SSP Zeroization, the user shall ensure that all the contexts that use that SSP shall also be zeroized.

11.3 Non-Administrator Guidance

The Module only supports the Cryptographic Officer operator role and does not support non-administrator operator roles.

Therefore, there is no non-administrator guidance.

11.4 Design and Rules

The inherent properties of the Module are:

1. Manual key entry is not supported.
2. Data output is inhibited during self-tests, zeroization, SSP generation and error states.
3. The Module does not perform any cryptographic function if any self-test has failed.

11.5 End of Life

To cease using the module, power off the module. The module does not possess persistent storage of SSPs. The SSP value only exists in volatile memory and that value vanishes when the module is powered off. So as a first step for the secure sanitization, the module needs to be powered off.

12 Mitigation of Other Attacks

The module implements two mitigations against timing-based side-channel attacks, namely Constant-time Implementations and Blinding.

Constant-time Implementations protect cryptographic implementations in the Module against timing analysis since such attacks exploit differences in execution time depending on the cryptographic operation, and constant-time implementations ensure that the variations in execution time cannot be traced back to the key, CSP or secret data.

Numeric Blinding protects the RSA and ECDSA algorithms from timing attacks. These algorithms are vulnerable to such attacks since attackers can measure the time of signature operations or RSA decryption. To mitigate this the Module generates a random blinding factor which is provided as an input to the decryption/signature operation and is discarded once the operation has completed and resulted in an output. This makes it difficult for attackers to attempt timing attacks on such operations without the knowledge of the blinding factor and therefore the execution time cannot be correlated to the RSA/ECDSA key.

13 References and Definitions

List with the different references and definitions used in this document.

Abbreviation	Full Specification Name
[NIST]	National Institute of Standards and Technology.
[FIPS140-3]	Security Requirements for Cryptographic Modules, March 22, 2019.
[IG]	Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program.
[ISO19790]	Information technology – Security techniques – Security requirements for cryptographic modules, 2012(2014).
[FIPS186-4]	Digital Signature Standard (DSS), July 2013.
[FIPS186-5]	Digital Signature Standard (DSS), February 3rd, 2023.
[SP800-38D]	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, November 2007.
[SP800-38E]	Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices, January 2010.
[SP800-52r2]	Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations, August 2019.
[SP800-56Arev3]	Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography, April 2018.
[SP800-57 Part1 Rev5]	Recommendation for Key Management: Part 1 – General, May 2020.
[SP800-90Arev1]	Recommendation for Random Number Generation Using Deterministic Random Bit Generators, June 2015.
[SP800-90B]	Recommendation for the Entropy Sources Used for Random Bit Generation.
[SP800-108rev1]	Recommendation for Key Derivation Using Pseudorandom Functions
[SP800-131Arev2]	Transitioning the Use of Cryptographic Algorithms and Key Lengths, March 2019.
[SP800-132]	Recommendation for Password-Based Key Derivation - Part 1: Storage Applications, December 2010.
[SP800-133rev2]	Recommendation for Cryptographic Key Generation, June 2020.

[RFC5288]	AES Galois Counter Mode (GCM) Cipher Suites for TLS, August 2008.
[RFC5246]	The Transport Layer Security (TLS) Protocol Version 1.2, August 2008.
[RFC8446]	The Transport Layer Security (TLS) Protocol Version 1.3, August 2018.

Table 23: References