KAYTUS SYSTEMS PTE. LTD.

Linux OpenSSL FIPS Provider

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 Linux OpenSSL FIPS Provider cryptographic module. For the purpose of the FIPS 140-3 validation, the module is a software cryptographic module validated at overall security level 1. The following table shows the claimed security level for each of the twelve sections that comprise the FIPS 140-3 standard.

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:

Linux OpenSSL FIPS Provider is a software library that provides a C-language Application Program Interface (API) for use by application that require cryptographic functionality.

In the OpenSSL, providers are containers for algorithm implementations. Whenever a cryptographic algorithm is used via the OpenSSL high level APIs a provider is selected. It is that provider implementation that actually does the required work. Based on the OpenSSL new approach, the Linux OpenSSL FIPS Provider is a provider that can be used with any OpenSSL version from 3.0.4.

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: Multi-Chip Standalone

Module Characteristics:

Cryptographic Boundary:

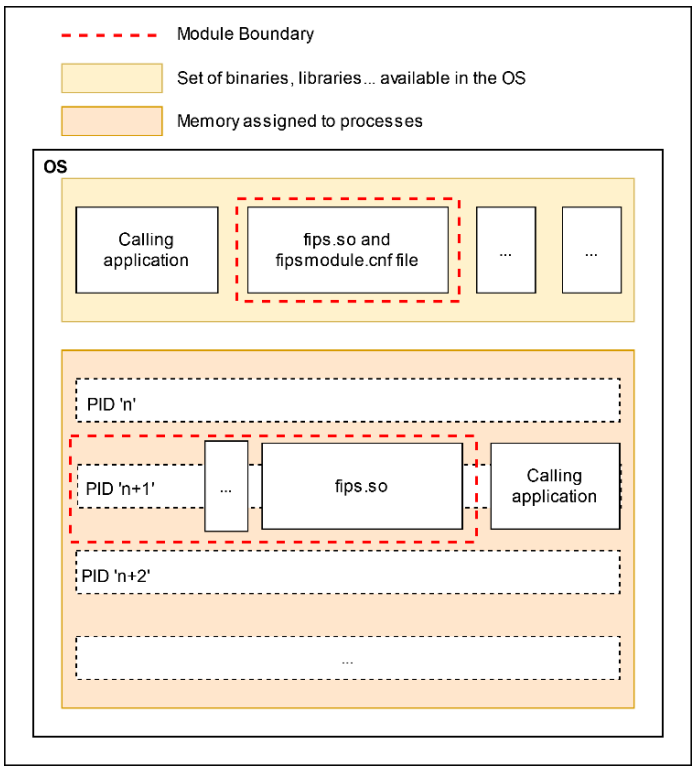
The cryptographic boundary consists of the following shared library and the integrity check file used for the integrity test. The following table enumerates the files that comprise the module (surrounded with red lines in Figure 1: Cryptographic Boundary).

Component	Description
fips.so	Shared library for the provider implementation v3.1
fipsmodule.cnf	File with HMAC authentication code for integrity

Table 2 - Cryptographic Module Components

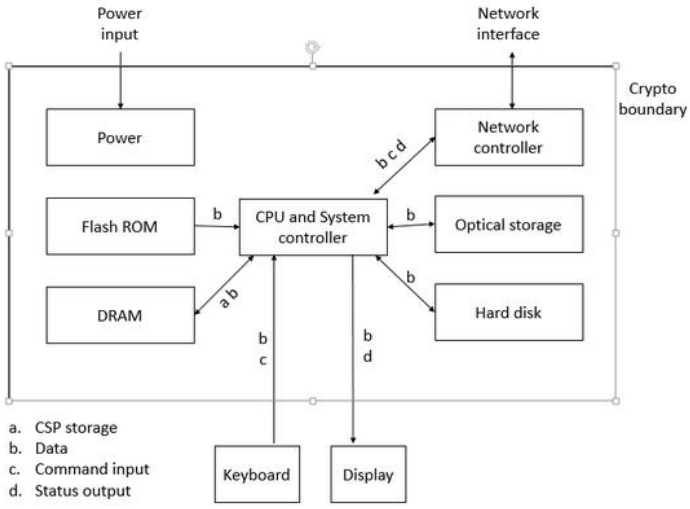
The image below illustrates the high-level software architecture of the module. The block diagram below shows the module and the delimitation of the cryptographic module boundary (dotted red line).

Figure 1: Cryptographic Boundary



The module is aimed to run on a general-purpose computer (GPC); the physical perimeter of the module is the tested platforms. Figure 2 shows the major components of a GPC.

Figure 2: Cryptographic Module Physical Perimeter



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	v3.1	None	Message authentication with HMAC-SHA2-256

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

Tested Module Identification – Hybrid Disjoint Hardware:

N/A for this module.

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Linux 5.4.85	KR2280V2 Rack Server	AST2600 (ARM Cortex A7)	No	None	v3.1
Linux 5.15.50	KR2280V2 Rack Server	AST2600 (ARM Cortex A7)	No	None	v3.1

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

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

N/A for this module.

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

N/A. The module does not have excluded components

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 an allowed security functions can be invoked.	Approved	Pre-operational self-test successfully passed (SELF_TEST_post() = 1)

Table 5: Modes List and Description

The module supports only one mode of operation: 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.

The module does not support degraded operation.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A4198	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC-CS1	A4198	Direction - decrypt, encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 128-512 Increment 8	SP 800-38A
AES-CBC-CS2	A4198	Direction - decrypt, encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 128-512 Increment 8	SP 800-38A
AES-CBC-CS3	A4198	Direction - decrypt, encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 136-512 Increment 8	SP 800-38A
AES-CCM	A4198	Key Length - 128, 192, 256 Tag Length - 112, 128, 32, 48, 64, 80, 96 IV Length - IV Length: 56, 64, 72, 80, 88, 96, 104 Payload Length - Payload Length: 0-256 Increment 8 AAD Length - AAD Length: 0-524288 Increment 8	SP 800-38C
AES-CFB1	A4198	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB128	A4198	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A4198	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A4198	Direction - Generation, Verification Key Length - 128, 192, 256 MAC Length - MAC Length: 128 Message Length - Message Length: 0-524288 Increment 8	SP 800-38B
AES-CTR	A4198	Direction - Decrypt, Encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 8-128 Increment 8 Supports Counter larger than maximum value - Yes Incremental Counter - Yes Counter Tests Performed - Yes	SP 800-38A
AES-ECB	A4198	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
AES-GCM	A4198	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96-1024 Increment 8 Payload Length - Payload Length: 8-65536 Increment 8 AAD Length - AAD Length: 0-65536 Increment 8	SP 800-38D
AES-GMAC	A4198	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 AAD Length - AAD Length: 0-65536 Increment 8	SP 800-38D
AES-KW	A4198	Direction - Decrypt, Encrypt Cipher - Cipher, Inverse Key Length - 128, 192, 256 Payload Length - Payload Length: 128-4096 Increment 128	SP 800-38F
AES-KWP	A4198	Direction - Decrypt, Encrypt Cipher - Cipher, Inverse Key Length - 128, 192 Payload Length - Payload Length: 8-4096 Increment 8	SP 800-38F
AES-OFB	A4198	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-XTS Testing Revision 2.0	A4198	Direction - Decrypt, Encrypt Key Length - 128, 256 Payload Length - Payload Length: 128-65536 Increment 128 Tweak Mode - Hex Data Unit Length Matches Payload Length - Yes	SP 800-38E
Counter DRBG	A4198	Prediction Resistance - Yes Supports Reseed - Yes Mode - AES-128, AES-192, AES-256 Derivation Function Enabled - No, Yes Additional Input - Additional Input: 0-256 Increment 256, Additional Input: 256, Additional Input: 320, Additional Input: 384 Entropy Input - Entropy Input: 128-256 Increment 128, Entropy Input: 256, Entropy Input: 256-512 Increment 128, Entropy Input: 320, Entropy Input: 384 Nonce - Nonce: 0, Nonce: 128 Personalization String Length - Personalization String Length: 0-256 Increment 256, Personalization String Length: 256, Personalization String Length: 320, Personalization	SP 800-90A Rev. 1

Algorithm	CAVP Cert	Properties	Reference
		String Length: 384 Returned Bits - 256	
DSA KeyGen (FIPS186-4)	A4198	L - 2048, 3072 N - 224, 256	FIPS 186-4
DSA PQGGen (FIPS186-4)	A4198	P/Q Generation Methods - Probable G Generation Methods - Canonical, Unverifiable L - 2048, 3072 N - 224, 256 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	FIPS 186-4
DSA PQGVer (FIPS186-4)	A4198	P/Q Generation Methods - Probable G Generation Methods - Canonical, Unverifiable L - 1024, 2048, 3072 N - 160, 224, 256 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	FIPS 186-4
DSA SigGen (FIPS186-4)	A4198	L - 2048, 3072 N - 224, 256 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	FIPS 186-4
DSA SigVer (FIPS186-4)	A4198	L - 1024, 2048, 3072 N - 160, 224, 256 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	FIPS 186-4
ECDSA KeyGen (FIPS186-4)	A4198	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 186-4
ECDSA KeyVer (FIPS186-4)	A4198	Curve - B-163, B-233, B-283, B-409, B-571, K-163, K-233, K-283, K-409, K-571, P-192, P-224, P-256, P-384, P-521	FIPS 186-4
ECDSA SigGen (FIPS186-4)	A4198	Component - No 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	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A4198	Component - No, Yes Curve - B-163, B-233, B-283, B-409, B-571, K-163, K-233, K-283, K-409, K-571, P-192, P-224, P-256, P-384, P-521 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	FIPS 186-4
Hash DRBG	A4198	Prediction Resistance - Yes Supports Reseed - Yes Mode - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 Entropy Input - Entropy Input: 128-256 Increment 64, Entropy Input: 192-256 Increment 64, Entropy Input: 256-	SP 800-90A Rev. 1

Algorithm	CAVP Cert	Properties	Reference
		320 Increment 64 Nonce - Nonce: 128-160 Increment 32, Nonce: 96-128 Increment 32 Personalization String Length - Personalization String Length: 0-256 Increment 128 Additional Input - Additional Input: 0-256 Increment 128 Returned Bits - 160, 224, 256, 384, 512	
HMAC DRBG	A4198	Prediction Resistance - Yes Supports Reseed - Yes Mode - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 Entropy Input - Entropy Input: 160-256 Increment 32, Entropy Input: 192-256 Increment 64, Entropy Input: 256-512 Increment 64, Entropy Input: 384-512 Increment 64, Entropy Input: 512-1024 Increment 64 Nonce - Nonce: 128, Nonce: 64, Nonce: 96 Personalization String Length - Personalization String Length: 0-192 Increment 64, Personalization String Length: 0-256 Increment 128 Additional Input - Additional Input: 0-256 Increment 128, Additional Input: 192 Returned Bits - 160, 224, 256, 384, 512	SP 800-90A Rev. 1
HMAC-SHA-1	A4198	MAC - MAC: 32-160 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A4198	MAC - MAC: 32-224 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A4198	MAC - MAC: 32-256 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A4198	MAC - MAC: 32-384 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A4198	MAC - MAC: 32-256 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512/224	A4198	MAC - MAC: 32-224 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512/256	A4198	MAC - MAC: 32-256 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-224	A4198	MAC - MAC: 32-224 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-256	A4198	MAC - MAC: 32-256 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-384	A4198	MAC - MAC: 32-384 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-512	A4198	MAC - MAC: 32-512 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1

Algorithm	CAVP Cert	Properties	Reference
KAS-ECC CDH-Component SP800-56Ar3 (CVL)	A4198	Function - Full Public Key Validation, Key Pair Generation 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	A4198	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	A4198	Domain Parameter Generation Methods - FB, FC, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 Scheme - dhEphem - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-IFC-SSC	A4198	Modulo - 2048, 3072, 4096, 6144, 8192 Key Generation Methods - rsakpg1-basic, rsakpg1-crt, rsakpg1-prime-factor, rsakpg2-basic, rsakpg2-crt, rsakpg2-prime-factor Scheme - KAS1 - KAS Role - initiator, responder KAS2 - KAS Role - initiator, responder Fixed Public Exponent - 010001	SP 800-56A Rev. 3
KDA HKDF SP800-56Cr2	A4198	Fixed Info Pattern - algorithmId||l||uPartyInfo||vPartyInfo Fixed Info Encoding - concatenation 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 Perform Multiple Expansion Tests - No Uses Hybrid Shared Secret - No	SP 800-56C Rev. 2
KDA OneStep SP800-56Cr2	A4198	Auxiliary Function Methods - Auxiliary Function Name - HMAC-SHA2-224 MAC Salting Methods - default, random Fixed Info Pattern - algorithmId||l||uPartyInfo||vPartyInfo Fixed Info Encoding - concatenation Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-8192 Increment 8	SP 800-56C Rev. 2
KDA TwoStep SP800-56Cr2	A4198	MAC Salting Methods - default, random Fixed Info Pattern - algorithmId||l||uPartyInfo||vPartyInfo	SP 800-56C Rev. 2

Algorithm	CAVP Cert	Properties	Reference
		Fixed Info Encoding - concatenation KDF Mode - feedback MAC Modes - 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 Fixed Data Order - after fixed data Counter Lengths - 8 The KDF supports an empty IV - Yes The KDF requires an empty IV - Yes Supported Lengths - Supported Lengths: 2048 Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-8192 Increment 8 Perform Multiple Expansion Tests - No Uses Hybrid Shared Secret - No	
KDF ANS 9.42 (CVL)	A4198	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 Other Info Length - Other Info Length: 0-4096 Increment 8 zz Length - zz Length: 8-4096 Increment 8 Key Data Length - Key Data Length: 8-4096 Increment 8 Supplemental Information Length - Supplemental Information Length: 8-120 Increment 8 OID - AES-128-KW, AES-192-KW, AES-256-KW	SP 800-135 Rev. 1
KDF ANS 9.63 (CVL)	A4198	Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512 Field Size - 224, 571 Shared Info Length - Shared Info Length: 0, 1024 Key Data Length - Key Data Length: 128, 4096	SP 800-135 Rev. 1
KDF SP800-108	A4198	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 Supported Lengths - Supported Lengths: 8, 72, 128, 776, 3456, 4096 Fixed Data Order - Before Fixed Data Counter Length - 32 Supports Empty IV - No Requires Empty IV - No Custom Key In Length - 0	SP 800-108 Rev. 1
KDF SSH (CVL)	A4198	Cipher - AES-128, AES-192, AES-256 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1

Algorithm	CAVP Cert	Properties	Reference
KMAC-128	A4198	Message Length - Message Length: 0-65536 Increment 8 MAC Length - MAC Length: 32-65536 Increment 8 Key Data Length - Key Data Length: 128-1024 Increment 8 Hex Customization - No Supports eXtendable-Output Functions - No, Yes	SP 800-185
KMAC-256	A4198	Message Length - Message Length: 0-65536 Increment 8 MAC Length - MAC Length: 32-65536 Increment 8 Key Data Length - Key Data Length: 128-1024 Increment 8 Hex Customization - No Supports eXtendable-Output Functions - No, Yes	SP 800-185
KTS-IFC	A4198	Function - keyPairGen IUT ID - abcd1234 Modulo - 2048, 3072, 4096, 6144 Key Generation Methods - rsakpg1-basic, rsakpg1-crt, rsakpg1-prime-factor, rsakpg2-basic, rsakpg2-crt, rsakpg2-prime-factor Fixed Public Exponent - 010001 Scheme - KTS-OAEP-basic - KAS Role - initiator, responder Key Transport Method - Hash Algorithms - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Supports Null Associated Data - Yes Associated Data Encoding - concatenation Key Length - 1024	SP 800-56B Rev. 2
PBKDF	A4198	Iteration Count - Iteration Count: 1-100000 Increment 1 Password Length - Password Length: 8-128 Increment 8	SP 800-132
RSA KeyGen (FIPS186-4)	A4198	Key Generation Mode - B.3.6 Modulo - 2048, 3072, 4096 Primality Tests - Table C.2 Private Key Format - Standard	FIPS 186-4
RSA SigGen (FIPS186-4)	A4198	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096	FIPS 186-4
RSA Signature Primitive (CVL)	A4198	Private Key Format - crt Public Exponent Mode - fixed Fixed Public Exponent - 010001	FIPS 186-4
RSA SigVer (FIPS186-4)	A4198	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA-1 Public Exponent Mode - Random	FIPS 186-4
Safe Primes Key Generation	A4198	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
Safe Primes Key Verification	A4198	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	SP 800-56A Rev. 3
SHA-1	A4198	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-224	A4198	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A4198	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A4198	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A4198	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512/224	A4198	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512/256	A4198	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA3-224	A4198	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-256	A4198	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-384	A4198	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-512	A4198	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHAKE-128	A4198	Supports Bit-Oriented Messages - No Supports Empty Message - Yes Supports Bit-Oriented Output - No Output Length - Output Length: 16-65536 Increment 8	FIPS 202
SHAKE-256	A4198	Supports Bit-Oriented Messages - No Supports Empty Message - Yes Supports Bit-Oriented Output - No Output Length - Output Length: 16-65536 Increment 8	FIPS 202
TLS v1.2 KDF RFC7627 (CVL)	A4198	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512 Key Block Length - Key Block Length: 1024	SP 800-135 Rev. 1
TLS v1.3 KDF (CVL)	A4198	HMAC Algorithm - SHA2-256, SHA2-384 KDF Running Modes - DHE, PSK, PSK-DHE	SP 800-135 Rev. 1

Table 6: Approved Algorithms

Note: Per [IG] 9.3.A, scenario 2b, no assurances of minimum strength of generated SSP are declared. The minimum entropy strength expected is 112 bits.

Note: The PBKDF2 algorithm is only used for internal pins and keys storage according FIPS SP 800-132. The module implements PBKDF2 based on SP 800-132 option 1a.

Vendor-Affirmed Algorithms:

The module includes the following vendor-affirmed security method in the next table:

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 7: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

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

The table below lists the Security Function Implementations supported by the module.

Name	Type	Description	Properties	Algorithms
KAS-ECC	KAS-SSC/KDF	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 X9.63-2001 and ANSI X9.42-2001) to derive keys for their respective industry standard protocols	IG:IG D.F scenario 2, path (2), split Key confirmation:No Key derivation:IG 2.4.B SP 800-135rev1 CVL and KDA (separately tested) Caveat:Key establishment methodology provides between 112 and 256 bits of encryption strength	KAS-ECC-SSC Sp800-56Ar3: (A4198) Curves: P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B- 233, B-283, B-409, B-571 TLS v1.2 KDF RFC7627: (A4198) Hash Algorithm: SHA2-256, SHA2-384, SHA2-512 Key Block Length: 1024 KDF SSH: (A4198) Cipher: AES-128, AES-192, AES-256 Hash Algorithm: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512 KDF ANS 9.63: (A4198) Hash Algorithm: SHA2-224, SHA2-256, SHA2-384, SHA2-512 Key Data Length: 128, 4096 KDF ANS 9.42: (A4198) Hash Algorithm: HA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224,

Name	Type	Description	Properties	Algorithms
				SHA3-256, SHA3-384, SHA3-512 Key Data Length: 8-4096 Increment 8 TLS v1.3 KDF: (A4198) HMAC Algorithm: SHA2-256, SHA2-384 KDF Running Modes: DHE, PSK, PSK-DHE KDA HKDF SP800-56Cr2: (A4198) 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 KDA OneStep SP800-56Cr2: (A4198) Auxiliary Function : HMAC-SHA2-224, KMAC-128, SHA2-512 KDA TwoStep SP800-56Cr2: (A4198) MAC Modes: 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,

Name	Type	Description	Properties	Algorithms
				HMAC-SHA3-384, HMAC-SHA3-512
KAS-FFC	KAS-SSC/KDF	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	IG:IG D.F Scenario 2, path (2), split Key confirmation:No Key derivation:IG 2.4.B SP 800-135rev1 CVL and KDA (separately tested) Caveat:Key establishment methodology provides between 112 and 200 bits of encryption strength	KAS-FFC-SSC Sp800-56Ar3: (A4198) Domain Parameter Generation Methods: FB, FC, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 TLS v1.2 KDF RFC7627: (A4198) Hash Algorithm: SHA2-256, SHA2-384, SHA2-512 Key Block Length: 1024 KDF SSH: (A4198) Cipher: AES-128, AES-192, AES-256 Hash Algorithm: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512 KDF ANS 9.63: (A4198) Hash Algorithm: SHA2-224, SHA2-256, SHA2-384, SHA2-512 Key Data Length: 128, 4096 KDF ANS 9.42: (A4198) Hash Algorithm: HA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384,

Name	Type	Description	Properties	Algorithms
				SHA3-512 Key Data Length: 8-4096 Increment 8 TLS v1.3 KDF: (A4198) HMAC Algorithm: SHA2-256, SHA2-384 KDF Running Modes: DHE, PSK, PSK-DHE KDA HKDF SP800-56Cr2: (A4198) HMAC Algorithms: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 KDA OneStep SP800-56Cr2: (A4198) Auxiliary Function: HMAC-SHA2-224, KMAC-128, SHA2-512 KDA TwoStep SP800-56Cr2: (A4198) MAC Modes: 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,

Name	Type	Description	Properties	Algorithms
				HMAC-SHA3-384, HMAC-SHA3-512
KAS-RSA	KAS-SSC/KDF	Uses the KAS_RSA_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	IG:IG D.F Scenario 1, path (2), split Key confirmation:No Key derivation:IG 2.4.B SP 800-135rev1 CVL and KDA (separately tested) Caveat:Key establishment methodology provides between 112 and 200 bits of encryption strength	KAS-IFC-SSC: (A4198) Moduli: 2048, 3072, 4096, 6144, 8192 TLS v1.2 KDF RFC7627: (A4198) Hash Algorithm: SHA2-256, SHA2-384, SHA2-512 Key Block Length: 1024 KDF SSH: (A4198) Cipher: AES-128, AES-192, AES-256 Hash Algorithm: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512 KDF ANS 9.63: (A4198) Hash Algorithm: SHA2-224, SHA2-256, SHA2-384, SHA2-512 Key Data Length: 128, 4096 KDF ANS 9.42: (A4198) Hash Algorithm: HA-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: 8-4096 Increment 8 TLS v1.3 KDF: (A4198)

Name	Type	Description	Properties	Algorithms
				HMAC Algorithm: SHA2-256, SHA2-384 KDF Running Modes: DHE, PSK, PSK-DHE KDA HKDF SP800-56Cr2: (A4198) HMAC Algorithms: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 KDA OneStep SP800-56Cr2: (A4198) Auxiliary Function: HMAC-SHA2-224, KMAC-128, SHA2-512 KDA TwoStep SP800-56Cr2: (A4198) MAC Modes: 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
Key Wrapping	BC-AuthDecrypt BC-AuthEncrypt	Key wrapping using AES KW or AES KWP.		AES-KW: (A4198) Key Length: 128, 192, 256

Name	Type	Description	Properties	Algorithms
				AES-KWP: (A4198) Key Length: 128, 256
Key Encapsulation	AsymKeyPair-Decap AsymKeyPair-Encap	Key encapsulation using RSA		KTS-IFC: (A4198) Moduli: 2048, 3072, 4096, 6144
Asymmetric Key Generation	AsymKeyPair-DomPar AsymKeyPair-KeyGen CKG	Asymmetric Key Pair/Domain Parameter Generation performed by DH, DSA, ECDSA or RSA		DSA KeyGen (FIPS186-4): (A4198) (L,N) value: (2048, 224), (2048, 256), (3072, 256) ECDSA KeyGen (FIPS186-4): (A4198) 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 RSA KeyGen (FIPS186-4): (A4198) Moduli: 2048, 3072, 4096 Counter DRBG: (A4198) Mode: AES-128, AES-192, AES-256 Hash DRBG: (A4198) Mode: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 HMAC DRBG: (A4198) Mode: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 DSA PQGGen (FIPS186-4): (A4198)

Name	Type	Description	Properties	Algorithms
				(L, N) value: (2048, 224), (2048, 256), (3072, 256) Safe Primes Key Generation: (A4198) Safe Primes Group: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 CKG: ()
Asymmetric Key Verification	AsymKeyPair-KeyVer	Asymmetric Key Verification by DH, DSA or ECDSA		DSA PQGVer (FIPS186-4): (A4198) (L, N) value: (1024, 160), (2048, 224), (2048, 256), (3072, 256) ECDSA KeyVer (FIPS186-4): (A4198) Curve: B-163, B-233, B-283, B-409, B-571, K-163, K-233, K-283, K-409, K-571, P-192, P-224, P-256, P-384, P-521 Safe Primes Key Verification: (A4198) Safe Prime Groups: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192
Digital Signature Generation	DigSig-SigGen	Digital Signature Generation using DSA, ECDSA or RSA		DSA SigGen (FIPS186-4): (A4198) (L, N) value: (2048, 224), (2048, 256), (3072, 256) Hash Algorithm: SHA2-224, SHA2-256, SHA2-384,

Name	Type	Description	Properties	Algorithms
				SHA2-512, SHA2-512/224, SHA2-512/256 ECDSA SigGen (FIPS186-4): (A4198) 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 RSA SigGen (FIPS186-4): (A4198) Signature Type: ANSI X9.31, PKCS 1.5 and PKCSPSS Moduli: 2048, 3072, 4096 Hash Pair for ANSI X9.31: SHA2-256, SHA2-384, SHA2-512 Hash Pair for PKCS 1.5: SHA2-224, SHA2-256, SHA2-384, SHA2-512 Hash Pair for PKCSPSS: SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 SHA2-224: (A4198) SHA2-256: (A4198) SHA2-384: (A4198) SHA2-512: (A4198)

Name	Type	Description	Properties	Algorithms
				SHA2-512/224: (A4198) SHA2-512/256: (A4198)
Digital Signature Verification	DigSig-SigVer	Digital Signature Verification using DSA, ECDSA or RSA		DSA SigVer (FIPS186-4): (A4198) (L,N) value: (1024, 160), (2048, 224), (2048, 256), (3072, 256) Hash Algorithm: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 ECDSA SigVer (FIPS186-4): (A4198) Curve: B-163, B-233, B-283, B-409, B-571, K-163, K-233, K-283, K-409, K-571, P-192, P-224, P-256, P-384, P-521 Hash Algorithm: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 RSA SigVer (FIPS186-4): (A4198) Signature Type: ANSI X9.31, PKCS 1.5 and PKCSPSS Moduli: 1024, 2048, 3072, 4096 Hash Pair for ANSI X9.31: SHA-1, SHA2-256, SHA2-

Name	Type	Description	Properties	Algorithms
				384, SHA2-512 Hash Pair for PKCS1.5 and PKCSPSS: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 SHA-1: (A4198) SHA2-224: (A4198) SHA2-256: (A4198) SHA2-384: (A4198) SHA2-512: (A4198) SHA2-512/224: (A4198) SHA2-512/256: (A4198)
Encryption	BC-UnAuth	Block Cipher Symmetric Encryption Non-Authenticated		AES-CBC: (A4198) Key Length: 128, 192, 256 AES-CBC-CS1: (A4198) Key Length: 128, 192, 256 AES-CBC-CS2: (A4198) Key Length: 128, 192, 256 AES-CBC-CS3: (A4198) Key Length: 128, 192, 256 AES-CFB1: (A4198) Key Length: 128, 192, 256 AES-CFB128: (A4198) Key Length: 128, 192, 256 AES-CFB8: (A4198) Key Length: 128, 192, 256 AES-CTR: (A4198) Key Length: 128, 192, 256 AES-ECB: (A4198) Key Length: 128, 192, 256 AES-OFB: (A4198) Key Length: 128, 192, 256

Name	Type	Description	Properties	Algorithms
				AES-XTS Testing Revision 2.0: (A4198) Key Length: 128, 256
Decryption	BC-UnAuth	Block Cipher Symmetric Decryption		AES-CBC: (A4198) Key Length: 128, 192, 256 AES-CBC-CS1: (A4198) Key Length: 128, 192, 256 AES-CBC-CS2: (A4198) Key Length: 128, 192, 256 AES-CBC-CS3: (A4198) Key Length: 128, 192, 256 AES-CFB1: (A4198) Key Length: 128, 192, 256 AES-CFB128: (A4198) Key Length: 128, 192, 256 AES-CFB8: (A4198) Key Length: 128, 192, 256 AES-CTR: (A4198) Key Length: 128, 192, 256 AES-ECB: (A4198) Key Length: 128, 192, 256 AES-OFB: (A4198) Key Length: 128, 192, 256 AES-XTS Testing Revision 2.0: (A4198) Key Length: 128, 192, 256
Authenticated Encryption	BC-Auth	Block Cipher Symmetric Authenticated Encryption		AES-CCM: (A4198) Key Length: 128, 192, 256 AES-GCM: (A4198) Key Length: 128, 192, 256
Authenticated Decryption	BC-Auth	Block Cipher Symmetric Authenticated Decryption		AES-CCM: (A4198) Key Length: 128, 192, 256

Name	Type	Description	Properties	Algorithms
				AES-GCM: (A4198) Key Length: 128, 192, 256
MAC1	MAC	Message Authentication Computation with AES CMAC or AES GMAC		AES-CMAC: (A4198) MAC Length: 128 Key Length: 128, 192, 256 AES-GMAC: (A4198) Key Length: 128, 192, 256
MAC2	MAC	Generate or verify data integrity with HMAC or KMAC		HMAC-SHA-1: (A4198) MAC Length: 32-160 Increment 8 Key Length: 8-524288 Increment 8 HMAC-SHA2-224: (A4198) MAC Length: 32-224 Increment 8 Key Length: 8-524288 Increment 8 HMAC-SHA2-256: (A4198) MAC Length: 32-256 Increment 8 Key Length: 8-524288 Increment 8 HMAC-SHA2-512: (A4198) MAC Length: 32-256 Increment 8 Key Length: 8-524288 Increment 8 HMAC-SHA2-512/224: (A4198) MAC Length: 32-224 Increment 8 Key Length: 8-524288 Increment 8

Name	Type	Description	Properties	Algorithms
				HMAC-SHA2-512/256: (A4198) MAC Length: 32-256 Increment 8 Key Length: 8-524288 Increment 8 HMAC-SHA3-224: (A4198) MAC Length: 32-224 Increment 8 Key Length: 8-524288 Increment 8 HMAC-SHA3-256: (A4198) MAC Length: 32-256 Increment 8 Key Length: 8-524288 Increment 8 HMAC-SHA3-384: (A4198) MAC Length: 32-384 Increment 8 Key Length: 8-524288 Increment 8 HMAC-SHA3-512: (A4198) MAC Length: 32-512 Increment 8 Key Length: 8-524288 Increment 8 KMAC-128: (A4198) MAC Length: 32-65536 Increment 8 Key Data Length: 128-1024 Increment 8 KMAC-256: (A4198) MAC Length: 32-65536

Name	Type	Description	Properties	Algorithms
				Increment 8 Key Data Length: 128-1024 Increment 8 HMAC-SHA2-384: (A4198) MAC Length: 32-384 Increment 8 Key Length: 8-524288 Increment 8
Message Digest	SHA	Message Digest		SHA2-224: (A4198) SHA2-256: (A4198) SHA2-384: (A4198) SHA2-512: (A4198) SHA2-512/224: (A4198) SHA2-512/256: (A4198) SHA3-224: (A4198) SHA3-256: (A4198) SHA3-384: (A4198) SHA3-512: (A4198) SHAKE-128: (A4198) SHAKE-256: (A4198) SHA-1: (A4198)
Key Derivation SP 800-135	KAS-135KDF	SP 800-135 Key derivation		KDF ANS 9.42: (A4198) OID: AES-128-KW, AES-192-KW, AES-256-KW 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 KDF ANS 9.63: (A4198) Hash Algorithm: SHA2-

Name	Type	Description	Properties	Algorithms
				224, SHA2-256, SHA2-384, SHA2-512
KDF-SSH	KAS-135KDF	Key derivation for SSH		KDF SSH: (A4198) Cipher: AES-128, AES-192, AES-256 Hash Algorithm: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512
Key Derivation 56Crev2	KAS-56CKDF	Key derivation using SP 800-56Crev2 implementation		KDA HKDF SP800-56Cr2: (A4198) 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 KDA OneStep SP800-56Cr2: (A4198) Auxiliary Function: HMAC-SHA2-224, KMAC-128, SHA2-512 KDA TwoStep SP800-56Cr2: (A4198) MAC Modes: 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,

Name	Type	Description	Properties	Algorithms
				HMAC-SHA3-384, HMAC-SHA3-512
PBKDF2	PBKDF	PBKDF2 Key derivation		PBKDF: (A4198) Option: 1a HMAC Algorithm: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 Iteration Count: 1-100000 Increment 1 Salt Length: 128-4096 Increment 8
KBKDF	KBKDF	KBKDF		KDF SP800-108: (A4198) 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
KDF-TLS	KAS-135KDF	Key Derivation for TLS		TLS v1.2 KDF RFC7627: (A4198) Hash Algorithm: SHA2-256, SHA2-384, SHA2-512 TLS v1.3 KDF: (A4198) HMAC Algorithm: SHA2-256, SHA2-384 KDF Running Modes: DHE, PSK, PSK-DHE

Name	Type	Description	Properties	Algorithms
Random Number Generation	DRBG	Random number generation		Counter DRBG: (A4198) Mode: AES-128, AES-192, AES-256 Hash DRBG: (A4198) Mode: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 HMAC DRBG: (A4198) Mode: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256
Digital Signature Generation Primitive	DigSig-SigGen	RSA Digital Signature Generation Primitive		RSA Signature Primitive: (A4198) Moduli: 2048
ECC CDH Primitive	KAS-SSC	ECC CDH Primitive in Shared Secret Computation		KAS-ECC CDH-Component SP800-56Ar3: (A4198) Curve: P-224, P-256, P-384, P-521 , K- K-233, K-283, K-409, K-571, B- 233, B-283, B-409, B-571
CKG	CKG	Direct output of DRBGs may be used for symmetric key generation per SP 800-133r2.		CKG: () Counter DRBG: (A4198) Mode: AES-128, AES-192, AES-256 HMAC DRBG: (A4198) Mode: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 Hash DRBG: (A4198) Mode: SHA-1, SHA2-224,

Name	Type	Description	Properties	Algorithms
				SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 AES GCM Usage

The module does not implement the TLS protocol itself, however, it provides the cryptographic functions required for implementing the protocol. AES GCM encryption is used in the context of the TLS protocol versions 1.2 and 1.3 (per Scenario 1 and Scenario 5 in [IG] C.H, respectively). For TLS v1.2, the mechanism for IV generation is compliant with RFC 5288. 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 RFC 5246. For TLS v1.3, the mechanism for IV generation is compliant with RFC 8446.

The module also supports internal IV generation using the module's Approved DRBG. The IV is at least 96-bits in length per NIST SP 800-38D, Section 8.2.1. Per [IG] C.H Scenario 2 and NIST SP 800-38D, the approved DRBG generates outputs such that the (key, IV) pair collision probability is less than 2-32.

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.

2.7.2 AES-XTS

The AES algorithm in XTS mode can be used for the cryptographic protection of data 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.

AES-XTS keys (i.e., Key_1 and Key_2) entered into the module shall be generated and/or established independently according to NIST SP 800-133rev2, Section 6.3. for an approved use of AES-XTS.

Based on the previous information, the module AES XTS implementation is compliant with FIPS 140-3 IG C.I.

2.7.3 SHA-3 Family

The module is compliant to the FIPS 140-3 IG C.C regarding the SHA-3 family algorithms.

All the SHA-3 and SHAKE functions have been tested and validated with the CAVP tool (#A4198), as it is indicated in Table 6. 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 (#A4198).

2.7.4 RSA Digital Signature

The module provides different algorithms for digital signature. Among them is the RSA FIPS 186-4, which is compliant with FIPS 140-3 IG C.F.

RSA digital signature generation can be performed with 2048-, 3072- or 4096-bits modulus length. As it is indicated in Table 6, all the RSA signature algorithm implementations are tested with the CAVP (#A4198). 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.

Regarding the Miller-Rabin tests round, the module performs 128 rounds when the modulus is higher than 2048 bits.

For the signature verification, the module has tested and validated all the different RSA modulus length implemented by the module: 1024, 2048, 3072 and 4096. This is also indicated in CAVP Cert. #A4198.

2.7.5 SHA-1 Usage

SHA-1 is non-approved for digital signature generation and deprecated through December 31, 2030, for non-digital signature applications.

2.8 RBG and Entropy

N/A for this module.

N/A for this module.

The module employs a Deterministic Random Bit Generator (DRBG) for the creation cryptographic key material. The module contains the following Deterministic Random Bit Generator (DRBG) compliant with the [SP 800-90Arev1]:

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

The module does not implement Non-Determinist Random Bit Generator or entropy source.

A minimum of 112-bits of entropy must be supplied. This entropy is supplied by means of callback functions. Those functions must return an error if the minimum entropy strength cannot be met.

2.9 Key Generation

For generation of RSA, DSA and ECDSA key pairs, the module implements approved key generation services compliant with [FIPS 186-4] 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 NIST SP 800-90Arev1 to generate random cryptographic material. The resulting generated material are unmodified outputs from the DRBG.

According to FIPS 140-3 Implementation Guidance 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

The module does not establish SSPs using an approved key agreement scheme (KAS). However, it does offer some or all of the underlying KAS cryptographic functionality to be used by an external operator/application as part of an approved KAS.

2.10.1 Key Agreement

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

The security strength of the preceding algorithms is as follows:

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

The Diffie-Hellman and EC Diffie-Hellman are under scenario 2 of [IG] D.F, and the RSA key agreement method is under scenario 1 of the same [IG].

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:

- Key-Based Key Derivation (KBKDF algorithm)
- Password-Based Key Derivation (PBKDF2 algorithm)
- Protocol-Suite Key Derivation: TLS 1.2 KDF, TLS 1.3 KDF, ANSI X9.42, ANSI X9.63 and SSHv2 KDF as key derivation functions.
- Key Derivation based on SP 800-56Crev2: HKDF and KDA.

Regarding PBKDF2, in line with the requirements for SP 800-132, keys generated using the approved PBKDF2 must only be used for storage applications. Any other use of the approved PBKDF2 is non-conformant.

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 password length should be between 8 and 128 bytes with an increment of 8 bytes.

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 bits.

In addition, users are referred to Appendix A, “Security Considerations” in SP 800-132 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 either 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 encapsulation based on SP 800-56Brev2.

According to Table 2: Comparable strengths in [SP 800-57 Part 1 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
- TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
- TLS_DHE_DSS_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	Control Input	API function calls, API input parameters for control.
N/A	Data Input	API input parameters, kernel I/O files on file system.
N/A	Data Output	API output parameters, kernel I/O files on file system.
N/A	Status Output	API return codes

Table 9: Ports and Interfaces

As a software module, the provider 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 (SMFI), and the physical ports are interpreted to be the 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.

In addition, all data and control output interfaces are inhibited when the module is performing pre-operational and conditional tests, zeroisation or when the module is in the 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 supports the following roles: Crypto Officer role, which performs the module installation and configuration; and User role, which performs all services with the exception of module installation and configuration.

Name	Type	Operator Type	Authentication Methods
User	Role	Other	None
Crypto Officer	Role	CO	None

Table 10: Roles

The operator implicitly assumes the set of roles consisting of the CO and User when accessing the module services. The module does not support concurrent operators.

4.3 Approved Services

All services implemented by the module are listed in tables below. The approved services are shown in Table 11. 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 initialization of the module	None	Module Default Entry Point	None	None	Crypto Officer
Show Status	Return the status of the module	None	API call parameters	Module Status	None	User
Show version information	Display the module name and version	None	API call parameters	Module Name and Version	None	User
On-Demand Self-Tests	Perform pre-operational and conditional self-tests	Return code from function call	Power cycle and/or API call parameters	Status	KAS-ECC KAS-FFC KAS-RSA Digital Signature Generation Digital Signature Verification Encryption Decryption Authenticated Encryption Authenticated Decryption MAC1 MAC2 Message Digest Key Derivation SP 800-135 KDF-SSH Key Derivation 56Crev2 PBKDF2 KBKDF	User

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					KDF-TLS Random Number Generation	
Zeroization	Zeroize and deallocate memory containing sensitive data	None	Power cycle or using provided "free" APIs	Status	None	User - AES ECB, CBC, CBC-CS1, CBC-CS2, CBC-CS3, CFB1, CFB8, CFB128, OFB, CTR Keys : Z - AES CBC, CBC-CS1, CBC-CS2, CBC-CS3, CFB1, CFB8, CFB128, OFB, CTR IVs: Z - AES XTS Key: Z - AES XTS IV: Z - AES KW/KWP Key: Z - AES CCM/GCM Key: Z - AES CCM IV: Z - AES GCM IV: Z - AES CMAC/GMAC

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Key: Z - HMAC Key: Z - KMAC Key: Z - DSA Public Key: Z - DSA Private Key: Z - ECDSA Public Key: Z - ECDSA Private Key: Z - RSA Private Key: Z - RSA Public Key: Z - Diffie-Hellman Public Key: Z - Diffie-Hellman Private Key: Z - ECDH Public Key: Z - ECDH Private Key: Z - Shared Secret: Z - HKDF salt: Z - Keying material: Z - PBKDF2 password: Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- PBKDF2 salt: Z - Entropy input string: Z - DRBG seed: Z - DRBG 'C' value: Z - DRBG 'V' value: Z - DRBG 'Key' value: Z
Symmetric Encryption/Decryption	Encrypt/Decrypt plaintext/ciphertext using supplied key	Return code from function call	API call parameters: key, IV, plaintext/ciphertext	Status, ciphertext/plaintext	Encryption Decryption Authenticated Encryption Authenticated Decryption	User - AES ECB, CBC, CBC-CS1, CBC-CS2, CBC-CS3, CFB1, CFB8, CFB128, OFB, CTR Keys : W,E - AES CBC, CBC-CS1, CBC-CS2, CBC-CS3, CFB1, CFB8, CFB128, OFB, CTR IVs: W,E - AES XTS Key: W,E - AES XTS IV: W,E - AES CCM/GCM Key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- AES CCM IV: W,E - AES GCM IV: W,E
Message Authentication Code	Generate or verify data integrity with CMAC and GMAC.	Return code from function call	API call parameters: key, data, authenticated message digest to verify.	Status, authenticated message digest	MAC1	User - AES CMAC/GMAC Key: R,E
Asymmetric Key Generation	Generate RSA, DSA, ECDSA, ECDH, DH keys.	Return code from function call	API call parameters: Curve, modulus, group, key length (N, L). Key pair to be verified	Status, Generated private and public key pair	Asymmetric Key Generation Asymmetric Key Verification	User - DSA Public Key: G,R,E - DSA Private Key: G,R,E - ECDSA Public Key: G,R,E - ECDSA Private Key: G,R,E - RSA Private Key: G,R,E - RSA Public Key: G,R,E - Diffie-Hellman Public Key: G,R,E - Diffie-Hellman Private Key: G,R,E - ECDH Public Key: G,R,E

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

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- Shared Secret: G
Message digest	Compute and return a message digest using SHS and SHA-3 algorithms	Return code from function call	API call parameters: message	Status, hash	Message Digest	User
Keyed Hash	Generate or verify data integrity with HMAC or KMAC.	Return code from function call	API call parameters: HMAC/KMAC key, message, keyed hash value (for verification)	Status, Keyed Hash value (Generation), True or false (Verification)	MAC2	User - HMAC Key: R,E - KMAC Key: R,E
Key Derivation	Derive keying material using KBKDF, PBKDF, HKDF, SP 800-56Crev2 One-Step KDF (KDA), SP 800-135rev1 TLS 1.2, SSHv2, ANSI X9.6-2001, ANSI X9.42-2001 KDFs and TLS 1.3 KDF.	Return code from function call	API call parameters: Shared Secret, salt, additional info depending on the algorithm used.	Status and derived keying material	Key Derivation SP 800-135 KDF-SSH Key Derivation 56Crev2 PBKDF2 KBKDF KDF-TLS	User - Shared Secret: R,E - Keying material: G - HKDF salt: R,E - PBKDF2 password: R,E - PBKDF2 salt: R,E
Key Encapsulation	Encrypt or Decrypt a key value on behalf of the calling application (does not establish keys into the module)	Return code from function call	API call parameters: Key-encryption key, Key to be encapsulated	Status and Encapsulated key	Key Encapsulation	User - RSA Private Key: R,E - RSA Public Key: R,E
Key Wrapping	Encrypt or Decrypt a key value on behalf of the calling application	Return code from	API call parameters: Key-encryption key, Key to be wrapped	Status and Wrapped key	Key Wrapping	User - AES KW/KWP Key: R,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	(does not establish keys into the module)	function call				
Random Number Generation	Used for random number and symmetric key generation using HMAC, HASH or CTR DRBG	Return code from function call	API call parameters: number of bits to be generated	Status and random bitstring	Random Number Generation CKG	User - Entropy input string: R,E - DRBG seed: W,E - DRBG 'C' value: W,E - DRBG 'V' value: W,E - DRBG 'Key' value: W,E

Table 11: Approved Services

4.4 Non-Approved Services

N/A for this module.

4.5 External Software/Firmware Loaded

The module does not implement the capability of software/firmware loading.

5 Software/Firmware Security

5.1 Integrity Techniques

The cryptographic module is composed of a software shared library as well as file where a pre-computed HMAC-SHA-256 signature is stored.

The integrity of the software module is achieved by comparing the pre-computed HMAC value to the one calculated at run time.

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 rebooting 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 operates in a modifiable operational environment per FIPS 140-3 level 1 specifications. The module runs on a commercially available general-purpose operating system.

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

The application that requests cryptographic services is the single user of the module. 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.

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 12: Storage Areas

Symmetric keys, HMAC keys, public and private keys are provided to the module by the calling application via API input parameters and are destroyed by the module when invoking the appropriate API function calls.

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	Automated	Electronic	
SSP Output	RAM	App via TOEPP Path	Plaintext	Automated	Electronic	

Table 13: 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 ISO/IEC 19790:2012 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 SSPs. 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 physical memory page of a process, the keys in physical memory are overwritten by the Linux kernel before the physical memory is allocated to another process.	The keys in physical memory are overwritten by the Linux kernel before the physical memory is allocated to another process.	Restart the module

Table 14: SSP Zeroization Methods

SSP zeroisation functions are implemented in the fips.so library, and they depend on the SSP to be zeroized.

The zeroization functions overwrite the memory occupied by 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.

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.

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

9.4 SSPs

The following section includes all the information related to the Sensitive Security Parameters (SSPs) and its management. Tables 17 and 18 below identify all the SSPs handled by the cryptographic module as well as its purpose, generation method, input and output method, where they are stored and how they are zeroized.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES ECB, CBC, CBC-CS1, CBC-CS2, CBC-CS3, CFB1, CFB8, CFB128, OFB, CTR Keys	AES Keys for Symmetric Encryption and Decryption	128, 192, 256 - 128, 192, 256	Symmetric Keys - CSP	External		Encryption Decryption
AES CBC, CBC-CS1, CBC-CS2, CBC-CS3, CFB1, CFB8, CFB128, OFB, CTR IVs	AES Initialization Vectors for Symmetric Encryption and Decryption	128 - 128	Initialization Vectors - CSP	External		Encryption Decryption
AES XTS Key	AES XTS Key for Symmetric Encryption and Decryption	128, 256 - 128, 256	Symmetric Keys - CSP	External		Encryption Decryption
AES XTS IV	AES XTS Initialization Vector	128 - 128	Initialization Vector - CSP	External		Encryption Decryption
AES KW/KWP Key	Key Wrapping	128, 192, 256 - 128, 192, 256	Symmetric Keys - CSP	External		Key Wrapping
AES CCM/GCM Key	AES Keys for Symmetric Authenticated Encryption and Decryption and Key Transport	128, 192, 256 - 128, 192, 256	Symmetric Keys - CSP	External		Authenticated Encryption Authenticated Decryption
AES CCM IV	AES Initialization Vector for Symmetric	56, 64, 72, 80, 88, 96, 104 - 56, 64, 72, 80, 88, 96, 104	Initialization Vector - CSP	External		Authenticated Encryption

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	Authenticated Encryption and Decryption and Key Transport					Authenticated Decryption
AES GCM IV	AES Initialization Vector for Symmetric Authenticated Encryption and Decryption and Key Transport	96 - 96	Initialization Vector - CSP	External		Authenticated Encryption Authenticated Decryption
AES CMAC/GMAC Key	AES Key for Message Authentication	128, 192, 256 - 128, 192, 256	Symmetric Keys - CSP	External		MAC1
HMAC Key	HMAC Key used for message authentication	112-bits or greater - 112-bits or greater	HMAC Key - CSP	External		MAC2
KMAC Key	KMAC Key used for message authentication	128-bits or greater - 128-bits or greater	KMAC Key - CSP	External		MAC2
DSA Public Key	DSA Public Key	1024, 2048, 3072-bit key - 112, 128, 192	Asymmetric key - PSP	Asymmetric Key Generation External		Asymmetric Key Verification Digital Signature Verification
DSA Private Key	DSA Private Key	224 and 256-bit key - 224 and 256-bit key	Asymmetric key - CSP	Asymmetric Key Generation External		Digital Signature Generation
ECDSA Public Key	ECDSA Public Key	From 192-576 bits - From 192-576 bits	Asymmetric key - PSP	Asymmetric Key Generation External		Asymmetric Key Verification Digital

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
						Signature Verification
ECDSA Private Key	ECDSA Private Key	From 224-576 bits - From 224-576 bits	Asymmetric key - CSP	Asymmetric Key Generation External		Digital Signature Generation
RSA Private Key	RSA Private Key	2048, 3072, 4096, 6144-bit key - 112, 128, 152, 176 bits	Asymmetric key - CSP	Asymmetric Key Generation External		KAS-RSA Key Encapsulation Digital Signature Generation
RSA Public Key	RSA Public Key	1024, 2048, 3072, 4096, 6144-bit key - 80, 112, 128, 152, 176 bits	Asymmetric key - PSP	Asymmetric Key Generation External		KAS-RSA Key Encapsulation Digital Signature Verification
Diffie-Hellman Public Key	Diffie-Hellman Public Key	From 2048-bit to 8192-bit key - 112, 128, 152, 200 bits	Asymmetric key - PSP	Asymmetric Key Generation External		KAS-FFC
Diffie-Hellman Private Key	Diffie-Hellman Private Key	From 2048-bit to 8192-bit key - 112, 128, 152, 200 bits	Asymmetric key - CSP	Asymmetric Key Generation External		KAS-FFC
ECDH Public Key	ECDH Public Key	From 224-576 bits - 112-256 bits	Asymmetric key - PSP	Asymmetric Key Generation External		KAS-ECC
ECDH Private Key	ECDH Private Key	From 224-576 bits - 112-256 bits	Asymmetric key - CSP	Asymmetric Key		KAS-ECC

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
				Generation External		
Shared Secret	Shared Secret	112 or greater - 112 or greater	Bitstring - CSP		KAS-ECC KAS-FFC KAS-RSA	
HKDF salt	HKDF salt	Salt bitstring - Salt bitstring	Bitstring - CSP	External		Key Derivation 56Crev2
Keying material	Keying material derived from key derivation function (SP 800-108rev1 KBKDF, SP 800-132 PBKDF, HKDF, KDA, SP 800-135rev1 KDFs, TLS 1.3 KDF)	Keying material bitstring - Keying material bitstring	Bitstring - CSP	Key Derivation SP 800-135 KDF-SSH Key Derivation 56Crev2 PBKDF2 KBKDF KDF-TLS		Key Derivation SP 800-135 KDF-SSH Key Derivation 56Crev2 PBKDF2 KBKDF KDF-TLS
PBKDF2 password	PBKDF2 password	From 8 to 128 characters (bytes) with increment of 8 characters. - From 8 to 128 characters (bytes) with increment of 8 characters.	Password - CSP	External		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	External		PBKDF2
Entropy input string	Entropy material for DRBG	For CTR_DRBG: 128, 256-bits for AES-128 DF 256, 384, 512-bits for AES-192/256 DF 256-bits for AES-128 NODF 320-bits for AES-192 NODF 384-bits for AES-256 NODF For HASH DRBG: 128, 192, 256-bits for SHA-1 192, 256-bits for	DRBG material - CSP	External		Random Number Generation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		SHA-224 256, 320-bits for SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256 For HMAC DRBG: 160-256 bits in increment of 32 bits for HMAC-SHA-1 192, 256-bits for HMAC-SHA-224 256, 384, 448, 512-bits for HMAC-SHA-256 384, 448, 512-bits for HMAC-SHA-384 512-1024-bits in increment of 64 bits for HMAC-SHA-512 - For CTR_DRBG: 128, 256-bits for AES-128 DF 256, 384, 512-bits for AES-192/256 DF 256-bits for AES-128 NODF 320-bits for AES-192 NODF 384-bits for AES-256 NODF For HASH DRBG: 128, 192, 256-bits for SHA-1 192, 256-bits for SHA-224 256, 320-bits for SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256 For HMAC DRBG: 160-256 bits in increment of 32 bits for HMAC-SHA-1 192, 256-bits for HMAC-SHA-224 256, 384, 448, 512-bits for HMAC-SHA-256 384, 448, 512-bits for HMAC-SHA-384 512-1024-bits in increment of 64 bits for HMAC-SHA-512				
DRBG seed	DRBG seed	For HASH DRBG and HMAC DRBG: 440-bits random data for SHA-1, SHA-224 and SHA-256;	DRBG material - CSP	External		Random Number Generation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		888-bits random data for SHA-384, SHA-512. For CTR DRBG: 256-bits random data for CTR-128-DF/CTR-128-NODF 320-bits random data for CTR-192-DF/ CTR-192-NODF 384-bits random data for CTR-256-DF/ CTR-256-NODF - For HASH DRBG and HMAC DRBG: 440-bits random data for SHA-1, SHA-224 and SHA-256; 888-bits random data for SHA-384, SHA-512. For CTR DRBG: 256-bits random data for CTR-128-DF/CTR-128-NODF 320-bits random data for CTR-192-DF/ CTR-192-NODF 384-bits random data for CTR-256-DF/ CTR-256-NODF				
DRBG 'C' value	Used for DRBG	Internal state value - Internal state value	DRBG material - CSP	Hash DRBG (A4198)		Random Number Generation
DRBG 'V' value	Used for DRBG	Internal state value - Internal state value	DRBG material - CSP	Random Number Generation		Random Number Generation
DRBG 'Key' value	Used for DRBG	128, 192, 256-bit key for CTR-DRBG 160, 224, 256, 384, 512 for HMAC-DRBG - 128, 192, 256-bit key for CTR-DRBG 160, 224, 256, 384, 512 for HMAC-DRBG	DRBG material - CSP	Counter DRBG (A4198) HMAC DRBG (A4198)		Random Number Generation

Table 15: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES ECB, CBC, CBC-CS1, CBC-CS2, CBC-CS3, CFB1, CFB8, CFB128, OFB, CTR Keys	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	AES CBC, CBC-CS1, CBC-CS2, CBC-CS3, CFB1, CFB8, CFB128, OFB, CTR IVs:Used With
AES CBC, CBC-CS1, CBC-CS2, CBC-CS3, CFB1, CFB8, CFB128, OFB, CTR IVs	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	AES ECB, CBC, CBC-CS1, CBC-CS2, CBC-CS3, CFB1, CFB8, CFB128, OFB, CTR 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
AES KW/KWP Key	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	
AES CCM/GCM Key	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	AES CCM IV:Used With AES GCM IV:Used With
AES CCM IV	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	AES CCM/GCM Key:Used With
AES GCM IV	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	AES CCM/GCM Key:Used With
AES CMAC/GMAC Key	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	
HMAC Key	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
KMAC Key	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	
DSA Public Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	DSA Private Key:Paired With
DSA Private Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	DSA 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
RSA Private Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	RSA Public Key:Paired With
RSA Public Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	RSA Private Key:Paired With
Diffie-Hellman Public Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	Diffie-Hellman Private Key:Paired With

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

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Entropy input string	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	DRBG seed:Used With DRBG 'C' value:Used With DRBG 'V' value:Used With DRBG 'Key' value:Used With
DRBG seed	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	Entropy input string: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	Entropy input string:Used With DRBG seed:Used With DRBG 'V' value:Used With
DRBG 'V' value		RAM:Plaintext	Ephemeral	API call Power Cycle	Entropy input string: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 'V' value:Used With DRBG seed:Used With Entropy input string:Used With

Table 16: 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 (A4198)	256-bit hardcoded key	KAT	SW/FW Integrity	None	Integrity self-test perform at the initialization of the module in order to verify its integrity.

Table 17: Pre-Operational Self-Tests

The module executes the following pre-operational self-tests when it is loaded into memory, without operator intervention.

During compilation, the module compiles the HMAC key into fips.so. Once compiled, during the installation, the module calculates the HMAC-SHA2-256 for fips.so and stores in fipsmodule.cnf file.

At runtime and prior to using HMAC-SHA-256, a Conditional Cryptographic Algorithm Self-Test (CAST) is performed. If the CAST of the HMAC-SHA-256 is successful, the module computes the HMAC of fips.so. 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. In this state, an internal flag is set to prevent subsequent invocation of any cryptographic calls.

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 tests are completed successfully.

The verify_integrity() API function is in charge of performing the pre-operational self-tests. This API function is called from the SELF_TEST_post() API function, which also performs the KAT tests for each algorithm. The API function returns a ‘1’ if all pre-operational and KAT self-tests succeed, and a ‘0’ otherwise.

10.2 Conditional Self-Tests

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

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-256 (A4198)	128-bit hardcoded key	KAT	CAST	verify_integrity() = 1	Used for module integrity test	Power-up and On-demand

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA-1 (A4198)	Generation	KAT	CAST	SELF_TEST_kats() = 1	Message Digest Generation	Power-up and On-demand
SHA2-512 (A4198)	Generation	KAT	CAST	SELF_TEST_kats() = 1	Message Digest Generation	Power-up and On-demand
SHA3-256 (A4198)	Generation	KAT	CAST	SELF_TEST_kats() = 1	Message Digest Generation	Power-up and On-demand
AES-ECB Decrypt (Inverse Function) (A4198)	128-bit key	KAT	CAST	SELF_TEST_kats() = 1	Decrypt	Power-up and On-demand
AES-GCM Authenticated Encrypt (Forward Cipher) (A4198)	256 bits key size	KAT	CAST	SELF_TEST_kats() = 1	Encrypt	Power-up and On-demand
AES-GCM Authenticated Decrypt (Forward Cipher) (A4198)	256 bits key size	KAT	CAST	SELF_TEST_kats() = 1	Decrypt	Power-up and On-demand
RSA SigGen (FIPS186-4) (A4198)	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
RSA SigVer (FIPS186-4) (A4198)	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
DSA SigGen (FIPS186-4) (A4198)	2048, 224 bits key size with SHA2-256	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
DSA SigVer (FIPS186-4) (A4198)	2048, 224 bits key size with SHA2-256	KAT	CAST	SELF_TEST_kats() = 1	Signature Verification	Power-up and On-demand
ECDSA SigGen (FIPS186-4) (A4198)	P-224 and K-233 curves with SHA2-256	KAT	CAST	SELF_TEST_kats() = 1	Signature Generation	Power-up and On-demand
ECDSA SigVer (FIPS186-4) (A4198)	P-224 and K-233 curves with SHA2-256	KAT	CAST	SELF_TEST_kats() = 1	Signature Verification	Power-up and On-demand
Counter DRBG (A4198)	AES 128-bit key with derivation function.	KAT	CAST	SELF_TEST_kats() = 1	Random Bit Generation	Power-up and On-demand
Hash DRBG (A4198)	SHA2-256	KAT	CAST	SELF_TEST_kats() = 1	Random Bit Generation	Power-up and On-demand
HMAC DRBG (A4198)	SHA-1	KAT	CAST	SELF_TEST_kats() = 1	Random Bit Generation	Power-up and On-demand
KDF SP800-108 (A4198)	Counter Mode (HMAC-SHA2-256).	KAT	CAST	SELF_TEST_kats() = 1	Key Derivation	Power-up and On-demand
PBKDF (A4198)	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 ANS 9.42 (A4198)	Key Derivation	KAT	CAST	SELF_TEST_kats() = 1	Key Derivation	Power-up and On-demand
KDF ANS 9.63 (A4198)	Key Derivation	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 SSH (A4198)	SSHv2 KDF	KAT	CAST	SELF_TEST_kats() = 1	Key Derivation	Power-up and On-demand
TLS v1.2 KDF RFC7627 (A4198)	TLS v1.2 KDF	KAT	CAST	SELF_TEST_kats() = 1	Key Derivation	Power-up and On-demand
TLS v1.3 KDF (A4198)	TLS v1.3 KDF (per Section 7.1 of RFC 8446).	KAT	CAST	SELF_TEST_kats() = 1	Key Derivation	Power-up and On-demand
KAS-FFC-SSC Sp800-56Ar3 (A4198)	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 (A4198)	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
KAS-IFC-SSC (A4198)	2048 bits key size	KAT	CAST	SELF_TEST_kats() = 1	RSA Primitive Computation (per Scenario 1 of IG D.F and Section 8.2.2 in SP 800-56Brev2) as part of KTS-RSA Encryption KAT.	Power-up and On-demand
KDA OneStep SP800-56Cr2 (A4198)	One-step KDF	KAT	CAST	SELF_TEST_kats() = 1	One-step KDF (per Section 4 of SP 800-56Crev2).	Power-up and On-demand

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KDA TwoStep SP800-56Cr2 (A4198)	Two-Step KDF	KAT	CAST	SELF_TEST_kats() = 1	Two-Step KDF (per Section 5 of SP 800-56Crev2).	Power-up and On-demand
Key Encapsulation KAT for Basic (A4198)	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
Key Decapsulation KAT for Basic (A4198)	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
Key Decapsulation KAT for CRT (A4198)	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
DSA KeyGen (FIPS186-4) (A4198)	Generated Key Size	PCT	PCT	dsa_keygen_pairwise_test() = 1	Pairwise Consistency Test on Generation of a DSA Key Pair.	Key Generation
ECDSA KeyGen (FIPS186-4) (A4198)	Generated curve	PCT	PCT	ecdsa_keygen_pairwise_test() = 1	Pairwise Consistency Test on Generation of an ECDSA Key Pair.	Key Generation
RSA KeyGen (FIPS186-4) (A4198)	Generated Key size	PCT	PCT	rsa_keygen_pairwise_test() = 1	Pairwise Consistency Test on Generation of an RSA Key Pair.	Key Generation
KAS-ECC SP800-56Ar3 Assurances (A4198)	SP 800-56Arev3 Assurances	KAT	Critical Function	ossl_ec_key_public_check() = 1	Assurances per Section 5.6.2 of SP 800-56Arev3,	Power-up and On-demand

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
					required per [IG] D.F	
KAS-FFC SP800-56Ar3 Assurances (A4198)	SP 800-56Arev3 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
KAS-IFC SP800-56Br2 Assurances (A4198)	SP 800-56Brev2 Assurances	KAT	Critical Function	SELF_TEST_kats() = 1	Assurances per Section 6.4 of SP 800-56Brev2, required per [IG] D.F	Power-up and On-demand
DRBG Health Checks	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 18: Conditional Self-Tests

In addition to the pre-operational self-tests, the module also performs the Conditional Cryptographic Algorithm Self-Tests (CASTs) for all cryptographic functions of each approved cryptographic algorithm implemented by the module during the module initialization. Therefore, all CASTs are performed prior to the first operational use of the cryptographic algorithm.

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 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.

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 Critical Error state.

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.

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 (A4198)	KAT	SW/FW Integrity	On Demand	Automatic

Table 19: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A4198)	KAT	CAST	On Demand	Programmatically
SHA-1 (A4198)	KAT	CAST	On Demand	Programmatically
SHA2-512 (A4198)	KAT	CAST	On Demand	Programmatically
SHA3-256 (A4198)	KAT	CAST	On Demand	Programmatically
AES-ECB Decrypt (Inverse Function) (A4198)	KAT	CAST	On Demand	Programmatically
AES-GCM Authenticated Encrypt (Forward Cipher) (A4198)	KAT	CAST	On Demand	Programmatically
AES-GCM Authenticated Decrypt (Forward Cipher) (A4198)	KAT	CAST	On Demand	Programmatically
RSA SigGen (FIPS186-4) (A4198)	KAT	CAST	On Demand	Programmatically
RSA SigVer (FIPS186-4) (A4198)	KAT	CAST	On Demand	Programmatically
DSA SigGen (FIPS186-4) (A4198)	KAT	CAST	On Demand	Programmatically
DSA SigVer (FIPS186-4) (A4198)	KAT	CAST	On Demand	Programmatically
ECDSA SigGen (FIPS186-4) (A4198)	KAT	CAST	On Demand	Programmatically

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
ECDSA SigVer (FIPS186-4) (A4198)	KAT	CAST	On Demand	Programmatically
Counter DRBG (A4198)	KAT	CAST	On Demand	Programmatically
Hash DRBG (A4198)	KAT	CAST	On Demand	Programmatically
HMAC DRBG (A4198)	KAT	CAST	On Demand	Programmatically
KDF SP800-108 (A4198)	KAT	CAST	On Demand	Programmatically
PBKDF (A4198)	KAT	CAST	On Demand	Programmatically
KDF ANS 9.42 (A4198)	KAT	CAST	On Demand	Programmatically
KDF ANS 9.63 (A4198)	KAT	CAST	On Demand	Programmatically
KDF SSH (A4198)	KAT	CAST	On Demand	Programmatically
TLS v1.2 KDF RFC7627 (A4198)	KAT	CAST	On Demand	Programmatically
TLS v1.3 KDF (A4198)	KAT	CAST	On Demand	Programmatically
KAS-FFC-SSC Sp800-56Ar3 (A4198)	KAT	CAST	On Demand	Programmatically
KAS-ECC-SSC Sp800-56Ar3 (A4198)	KAT	CAST	On Demand	Programmatically
KAS-IFC-SSC (A4198)	KAT	CAST	On Demand	Programmatically
KDA OneStep SP800-56Cr2 (A4198)	KAT	CAST	On Demand	Programmatically
KDA TwoStep SP800-56Cr2 (A4198)	KAT	CAST	On Demand	Programmatically
Key Encapsulation KAT for Basic (A4198)	KAT	CAST	On Demand	Programmatically
Key Decapsulation KAT for Basic (A4198)	KAT	CAST	On Demand	Programmatically
Key Decapsulation KAT for CRT (A4198)	KAT	CAST	On Demand	Programmatically

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
DSA KeyGen (FIPS186-4) (A4198)	PCT	PCT	On Demand	Programmatically
ECDSA KeyGen (FIPS186-4) (A4198)	PCT	PCT	On Demand	Programmatically
RSA KeyGen (FIPS186-4) (A4198)	PCT	PCT	On Demand	Programmatically
KAS-ECC SP800-56Ar3 Assurances (A4198)	KAT	Critical Function	On Demand	Programmatically
KAS-FFC SP800-56Ar3 Assurances (A4198)	KAT	Critical Function	On Demand	Programmatically
KAS-IFC SP800-56Br2 Assurances (A4198)	KAT	Critical Function	On Demand	Programmatically
DRBG Health Checks	Health Checks	Critical Function	On Demand	Programmatically

Table 20: 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 test, critical security functions tests.	Failure of pre-operational self-tests, cryptographic algorithm self-tests, pairwise consistency test, critical security functions tests.	Restart the module	verify_integrity() = 0 SELF_TEST_kats() = 0 rsa_keygen_pairwise_test() = 0 ecdsa_keygen_pairwise_test() = 0 dsa_keygen_pairwise_test() = 0 ossl_ec_key_public_check() = 0 DH_check_pub_key() = 0 self_test_drbg ()= 0
Soft Error	AES XTS check failure.	AES XTS check failure.	The module clears the error automatically	aes_xts_check_keys_differ() = 0

Table 21: Error States

If any of the self-tests described in sections 10.1, 10.2 and 10.3 fails, 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 is allowed, until the error is cleared.

The previous table 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 delivered already compiled and in the binary form for the following operational environments:

- Hardware Platform: KR2280V2 Rack Server, Operating System: Linux 5.4.85 with OpenSSL version => 3.0.4
- Hardware Platform: KR2280V2 Rack Server, Operating System: Linux 5.15.50 with OpenSSL version => 3.0.4

For installation, the operator shall follow the Crypto Officer guidance below described.

11.2 Administrator Guidance

The Crypto Officer shall run the following command in order to generate the fipsmodule.cnf file:

$ openssl fipsinstall –out fipsmodule.cnf -module fips.so -self_test_onload

After this step, the module is ready to use.

In order to check that the installation has been successfully performed, the Crypto Officer can execute the following command:

$ openssl list –providers

And the Linux provider should be indicated.

11.3 Non-Administrator Guidance

The module only supports one mode of operation: Approved. The module will be in Approved mode when all pre-operational self-tests have been completed successfully, and only Approved security functions are invoked. There are no additional installation, configuration, or usage instructions for operators intending to use the Linux OpenSSL FIPS Module.

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

12 Mitigation of Other Attacks

12.1 Attack List

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, DSA 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/DSA/ECDSA key.