KAYTUS

Linux OpenSSL FIPS Provider v3.1 Non-Proprietary Cryptographic Module

FIPS 140-3 Security Policy

FIPS 140-3 Security Level:	1
Document version:	1.3
Date:	June 29, 2024

Table of Contents

List of Tables

List of Figures

0 Introduction

0.1 Purpose

This is a non-proprietary Cryptographic Module Security Policy for the Linux OpenSSL FIPS Provider from KAYTUS SYSTEM PTE. LTD.. This Security Policy describes how the Linux OpenSSL FIPS Provider meets the security requirements of Federal Information Processing Standards (FIPS) Publication 140-3, which details the U.S. and Canadian Government requirements for cryptographic modules. More information about the FIPS 140-3 standard and validation program is available on the National Institute of Standards and Technology (NIST) and the Canadian Centre for Cyber Security (CCCS) Cryptographic Module Validation Program (CMVP) website at https://csrc.nist.gov/projects/cryptographic-module-validation-program.

This document also describes how to run the Linux OpenSSL FIPS Provider in a secure Approved mode of operation. This policy was prepared as part of the Level 1 FIPS 140-3 validation. The Linux OpenSSL FIPS Provider is referred to in this document as the provider, cryptographic module, or the module.

0.2 References

This document deals only with operations and capabilities of the module in the technical terms of a FIPS 140-3 cryptographic module security policy. More information is available on the module from the following sources:

- The search page on the CMVP website (https://csrc.nist.gov/Projects/cryptographic-module-validation-program/Validated-Modules/Search) can be used to locate and obtain vendor contact information for technical or sales-related questions about the module.

1 General 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.

Table 1 – Security Level per FIPS 140-3 Section

Section	FIPS 140-3 Section	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

2 Cryptographic Module Specification

2.1 Module Specification

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.

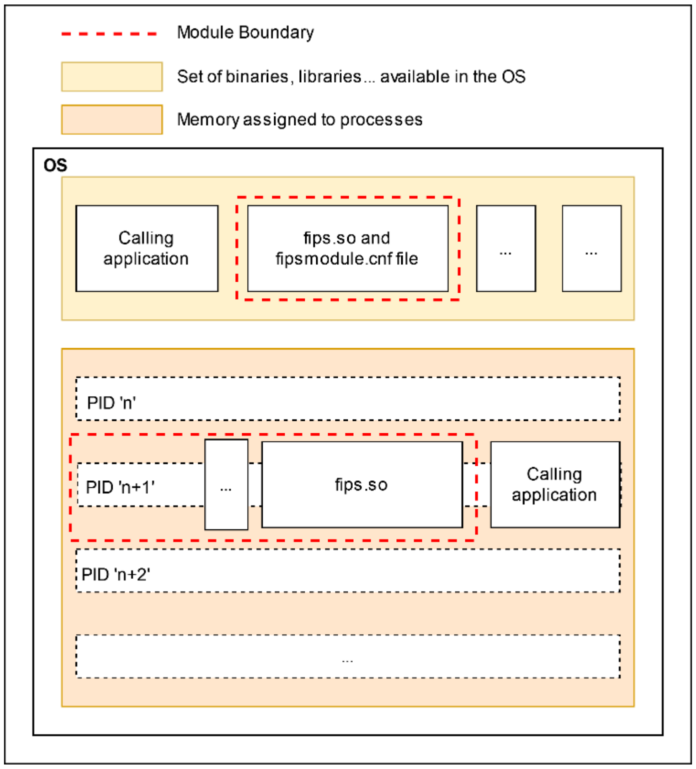
Based on that, the cryptographic boundary consists of the following shared library and the integrity check file used for the integrity test. The following table enumerates the files that comprise the module (surrounded with red lines in Figure 1 – Cryptographic Boundary).

Table 2 – Cryptographic Module Components

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

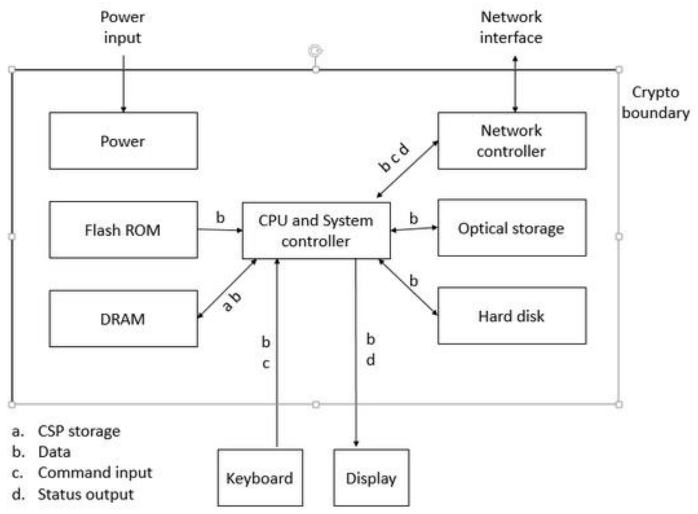
The image below illustrates the high-level software architecture of the module. The block diagram below shows the module and the delimitation of the cryptographic module boundary (dotted red line).

Figure 1 – Cryptographic Boundary


Module Boundary
Set of binaries, libraries... available in the OS
Memory assigned to processes
OS
Calling application
fips.so and fipsmodule.cnf file
...
...
PID 'n'
PID 'n+1'
...
fips.so
Calling application
PID 'n+2'
...

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



The module has been tested on the following platforms:

Table 3 – Software Tested Operational Environments

#	Operating system	Hardware Platform	Processor	PAA/Acceleration
1	Linux 5.4.85	KR2280V2 Rack Server	AST2600 (ARM Cortex A7)	N/A
2	Linux 5.15.50	KR2280V2 Rack Server	AST2600 (ARM Cortex A7)	N/A

Note: The operational environments include OpenSSL version 3.0.4.

There is no additional vendor affirmed operating environments.

2.2 Algorithms Implementation

The module supports the following Approved Algorithms listed in Table 4 below:

Table 4 – Approved Algorithms

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size / Key Strength	Use / Function
A4198	AES [SP 800-38D] [SP 800-38C]	GCM, CCM	Key Length: 128, 192, 256	Authenticated Symmetric Encryption and Decryption
A4198	AES [SP 800-38B] [SP 800-38D]	CMAC, GMAC	Key Length: 128, 192, 256	Generation and Verification of Message authentication
A4198	AES [SP 800-38A]	ECB, CBC, CBC-CS1, CBC-CS2, CBC-CS3, OFB, CFB1, CFB8, CFB128, CTR	Key Length: 128, 192, 256	Symmetric Encryption and Decryption
A4198	AES [SP 800-38F]	KW, KWP	Key Length: 128, 192, 256	Key Wrapping and Unwrapping
A4198	AES [SP 800-38E]	XTS	Key Length: 128, 256	Symmetric Encryption and Decryption
A4198	SHA-3, SHAKE [FIPS 202]	SHA3-224, SHA3-256, SHA3-384, SHA3-512 SHAKE-128, SHAKE-256	N/A	Message Digest
A4198	SHS [FIPS 180-4]	SHA-1 SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	N/A	Message Digest
A4198	HMAC [FIPS 198-1]	SHA-1	N/A	Message Authentication

		SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 SHA3-224, SHA3-256, SHA3-384, SHA3-512		
A4198	KMAC [SP 800-185]	KMAC-128, KMAC-256	KMAC-128, KMAC-256	Message Authentication
A4198	KAS-ECC-SSC [SP 800-56Arev3]	Ephemeral Unified Domain Parameter Generation Domain Parameter Validation Key Pair Generation Full Public Key Validation	P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571	Shared Secret Computation for Key Agreement
A4198	KAS-FFC-SSC [SP 800-56Arev3]	dhEphem Domain Parameter Generation Domain Parameter Validation Key Pair Generation Full Public Key Validation	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 safe prime groups per SP 800-56Ar3	Shared Secret Computation for Key Agreement
A4198	KAS-RSA-SSC [SP 800-56Brev2]	KAS1, KAS2 Key Generation Methods: rsakpg1-crt, rsakpg2-crt, rsakpg1-basic, rsakpg2-basic, rsakpg1-prime-factor, rsakpg2-prime-factor	Modulus: 2048, 3072, 4096, 6144, 8192	Shared Secret Computation for Key Agreement
A4198	CVL [SP 800-56Arev3] KAS-ECC-CDH		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	ECC CDH Primitive in Shared Secret Computation
A4198	Safe Primes	Key Generation for DH	Safe Prime Groups: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	Asymmetric Key Generation
		Key Verification for DH	Safe Prime Groups: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	Asymmetric Key Verification

A4198	DSA [FIPS 186-4]	Key Pair Generation	L = 2048, N = 224
			L = 2048, N = 256
			L = 3072, N = 256
				Asymmetric Key Generation

		Signature Generation	L = 2048, N = 224
			L = 2048, N = 256
			L = 3072, N = 256
			With all SHA-2 sizes
				Digital Signature Generation

		Signature Verification	L = 1024, N = 160
			L = 2048, N = 224
			L = 2048, N = 256
			L = 3072, N = 256
			With all SHA sizes
				Digital Signature Verification

		PQG Generation	L = 2048, N = 224
			L = 2048, N = 256
			L = 3072, N = 256
			With all SHA-2 sizes
				Primitive Generation

		PQG Verification	L = 1024, N = 160
			L = 2048, N = 224
			L = 2048, N = 256
			L = 3072, N = 256
			With all SHA sizes
				Primitive Verification

A4198	RSA [FIPS 186-4]	Key Pair Generation ANSI X9.31	Modulus: 2048, 3072, 4096	Asymmetric Key Generation
		Signature Generation (PKCS#1 v1.5)	Modulus: 2048, 3072, 4096 With SHA-224, SHA-256, SHA-384 and SHA-512	Digital Signature Generation
		Signature Generation (ANSI X9.31)	Modulus: 2048, 3072 and 4096 With SHA-256, SHA-384 and SHA-512	Digital Signature Generation
		Signature Generation (PKCS PSS)	Modulus: 2048, 3072 With SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256 Modulus: 4096 With SHA-224, SHA-256, SHA-384 and SHA-512	Digital Signature Generation
		Signature Verification (PKCS PSS and PKCS#1 v1.5)	Modulus: 1024, 2048, 3072, 4096 With SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256	Digital Signature Verification

		Signature Verification (ANSI X9.31)	Modulus: 1024, 2048, 3072, 4096 With SHA-1, SHA-256, SHA-384 and SHA-512	Digital Signature Verification
A4198	CVL [FIPS 186-4]	RSA Signature Generation Primitive	Modulus: 2048	Digital Signature Generation Primitive
A4198	ECDSA [FIPS 186-4]	Key Pair Generation	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 Testing Candidates	Asymmetric Key Generation
		Key Pair Verification	Curves: P-192, P-224, P-256, P-384, P-521 K-163, K-233, K-283, K-409, K-571 B-163, B-233. B-283, B-409, B-571	Asymmetric Key Verification
		Signature Generation	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 With all SHA-2 sizes	Digital Signature Generation
		Signature Verification	Curves: P-192, P-224, P-256, P-384, P-521 K-163, K-233, K-283, K-409, K-571 B-163, B-233. B-283, B-409, B-571 With SHA-1 and all SHA-2 sizes	Digital Signature Verification
A4198	DRBG [SP 800-90Arev1]	Hash DRBG	SHA-1 SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	Deterministic Random Number Generation
		HMAC DRBG	SHA-1 SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	
		CTR DRBG	AES-128, AES-192, AES-256	
N/A	KTS [SP 800-38F]	AES KW, KWP	AES KW, KWP	Key Transport

		AES CCM AES GCM (AES Cert #A4198; key establishment methodology provides between 128 and 256 bits of encryption strength)	AES CCM AES GCM	
		AES and HMAC (AES Cert #A4198 and HMAC Cert #A4198; key establishment methodology provides between 128 and 256 bits of encryption strength)	AES (any mode) and HMAC	
		AES and CMAC, GMAC (AES Cert #A4198 and AES Cert #A4198; key establishment methodology provides between 128 and 256 bits of encryption strength)	AES (any mode) and CMAC, GMAC	
A4198	KTS-RSA [SP 800-56Brev2]	Key encapsulation with RSA-OAEP, RSADP, RSAEP (Cert #A4198; key establishment methodology provides between 112 and 176 bits of encryption strength)	Modulus: 2048, 3072, 4096, 6144	Key Transport
A4198	KDA [SP 800-56Crev2]	One-Step KDF (Section 4), Two Step KDF (Section 5)	N/A	Key Derivation Function
A4198	HKDF [SP 800-56Crev2]	HMAC-based Extracted-and-Expand Key Derivation Function	N/A	Key Derivation Function
A4198	PBKDF2 [SP 800-132]	Option 1a	HMAC-SHA-1, SHA2-224, 256, 384, 512, 512/224, 512/256	Password-Based Key Derivation

			(C = 1 - 10.000, sLen = 16 – 512 bytes)	
A4198	KBKDF [SP 800-108rev1]	Counter, Feedback	CMAC AES128, CMAC AES192, CMAC AES256, HMAC-SHA-1, HMAC-SHA2-224, 256, 384, 512	Key-Based Key Derivation Function
A4198	CVL [SP 800-135rev1]	TLS 1.2 KDF RFC 7627 (SHA2-256, 384, 512), SSHv2 KDF (SHA-1, SHA2-224, 256, 384, 512), ANSI X9.63-2001 KDF (SHA2-224, 256, 384, 512), ANSI X9.42-2001 KDF (SHA-1, SHA2-224, 256, 384, 512, 512/224, 512/256, SHA3-224, 256, 384, 512).		Key Derivation Function *No parts of these protocols, other than the KDFs, have been tested by the CAVP or CMVP.*
A4198	CVL [RFC 8446] (Section 7.1)	TLS 1.3 KDF (SHA2-256, 384)		Key Derivation Function *No parts of these protocols, other than the KDFs, have been tested by the CAVP or CMVP.*

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 PBKDF algorithm is only used for internal pins and keys storage according FIPS SP 800-132.

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

Table 5 – Vendor Affirmed Approved Algorithms provided by Linux OpenSSL FIPS Provider

Algorithm and Standard	Caveat	Use / Function
CKG [SP 800 133rev2]	N/A	Key Generation Section 4 (Using the unmodified Output of a RBG) Section 6.1 (Direct Generation of Symmetric Keys) Section 6.2 (Derivation of Symmetric Keys)

The module does not implement either non-approved allowed algorithms (with or without security claimed) or non-approved and not allowed algorithms in the Approved mode of operation.

Table 6 – Security Functions Implementation

Name	Type	Description	SF Properties [O]	Algorithms / CAVP Cert
KTS1	KTS	Key Transport scheme using AES KW or AES KWP	Per IG D.G, approved key transport method. Provides between 128 and 256 bits of encryption strength.	AES-KW/A4198 AES-KWP/A4198
KTS2	KTS	Key Transport scheme using AES CCM or AES GCM	Per IG D.G, approved key transport method. Provides between 128 and 256 bits of encryption strength.	AES-CCM/A4198 AES-GCM/A4198

KTS3	KTS	Key Transport scheme using any mode of AES with HMAC	Per IG D.G, approved key transport method. Provides between 128 and 256 bits of encryption strength.	AES-CBC/A4198 AES-CBC-CS1/A4198 AES-CBC-CS2/A4198 AES-CBC-CS3/A4198 AES-CFB1/A4198 AES-CFB128/A4198 AES-CFB8/A4198 AES-CTR/A4198 AES-ECB/A4198 AES-OFB/A4198 HMAC-SHA-1/A4198 HMAC-SHA2-224/A4198 HMAC-SHA2-256/A4198 HMAC-SHA2-384/A4198 HMAC-SHA2-512/A4198 HMAC-SHA2-512/224/A4198 HMAC-SHA2-512/256/A4198 HMAC-SHA3-224/A4198 HMAC-SHA3-256/A4198 HMAC-SHA3-384/A4198 HMAC-SHA3-512/A4198
KTS4	KTS	Key Transport scheme using any mode of AES with AES CMAC or AES GMAC	Per IG D.G, approved key transport method. Provides between 128 and 256 bits of encryption strength.	AES-CBC/A4198 AES-CBC-CS1/A4198 AES-CBC-CS2/A4198 AES-CBC-CS3/A4198 AES-CFB1/A4198 AES-CFB128/A4198 AES-CFB8/A4198 AES-CMAC/A4198 AES-CTR/A4198 AES-ECB/A4198 AES-GMAC/A4198 AES-OFB/A4198
KTS5	KTS	Key Transport scheme using RSA	Per IG D.G, approved key transport method. Provides between 128 and 176 bits of encryption strength.	KTS-IFC/A4198

KAS-ECC	KAS	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	D.F scenario 2, path (1), no key confirmation, key derivation per IG 2.4.B	KAS-ECC-SSC Sp800-56Ar3/A4198 KDF TLS 1.2 RFC7627/A4198 KDF SSHv2/A4198 KDF ANSI X9.63/A4198 KDF ANSI X9.42/A4198 TLS v1.3 KDF/A4198
KAS-FFC	KAS	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	D.F scenario 2, path (1), no key confirmation, key derivation per IG 2.4.B	KAS-FFC-SSC Sp800-56Ar3/A4198 KDF TLS 1.2 RFC7627/A4198 KDF SSHv2/A4198 KDF ANSI X9.63/A4198 KDF ANSI X9.42/A4198 TLS v1.3 KDF/A4198
KAS-RSA	KAS	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	D.F scenario 1, path (1), no key confirmation, key derivation per IG 2.4.B	KAS-IFC-SSC Sp800-56Br2/A4198 KDF TLS 1.2 RFC7627/A4198 KDF SSHv2/A4198 KDF ANSI X9.63/A4198 KDF ANSI X9.42/A4198 TLS v1.3 KDF/A4198

2.3 Modes of Operation

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 (see Table 4 and Table 5 above).

The module does not support degraded operation.

3 Cryptographic Module 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, data output interface is inhibited when the module is performing pre-operational and conditional tests, zeroisation or when the module is in the error state.

The following table summarizes the FIPS 140-3 logical interfaces:

Table 7 – Ports and Interfaces

Physical Port	Logical Interface	Data that passes over port/interface
Keyboard, Serial port, Ethernet port, Network	Control Input	API function calls, API input parameters for control.
Ethernet ports, Keyboard	Data Input	API input parameters, kernel I/O files on file system.
Ethernet ports, Display	Data Output	API output parameters, kernel I/O files on file system.
Serial port, Ethernet port, Network, Display	Status Output	API return codes.

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

The sections below describe the module's authorized roles, services, and operator authentication method employed.

4.1 Authorized Roles and Authentication

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.

Table 8 – Roles

Name	Type	Operator Type	Authentication Methods
User	Role	Other	N/A
Crypto Officer	Role	CO	N/A

The module does not support authentication mechanisms. 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.

4.2 Module Services

All services implemented by the module are listed in tables below. The approved services are shown in Table 9. 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.
- X – Execute: The SSP is used within an Approved security function.
- Z – Zeroize: The SSP is zeroized.

Table 9 – Approved Services

Name	Roles	Description	Input	Output	SSP and Type of Access	Approved Security Function	Indicator
Initialization	CO	Perform initialization of the module	Module Default Entry Point	None	None	None	None
Show Status	User	Return the status of the module	API call parameters	Module Status	None	None	None
Show version information	User	Display the module name and version	API call parameters	Module Name and Version	N/A	N/A	None
Self-test on demand	User	Perform pre-operational and conditional self-tests	Power cycle and/or API call parameters	Status	None	Message Digest, Digital Signature Generation and Verification, Symmetric Encryption and Decryption, Random Number Generation, Key Derivation, Symmetric Digest, Keyed Hash, Key Agreement.	Return code from function call
Zeroization	User	Zeroize and de-allocate memory containing sensitive data	Power cycle or using provided free APIs	Status	All SSP's - Z	N/A	None

Symmetric Encryption/Decryption	User	Encrypt/Decrypt plaintext/ciphertext using supplied key	API call parameters: key, IV, plaintext/ciphertext	Status, ciphertext/plaintext	AES ECB key – W, X AES CBC Key – W, X AES CBC IV – W, X AES CBC-CS1 Key – W, X AES CBC-CS1 IV – W, X AES CBC-CS2 Key – W, X AES CBC-CS2 IV – W, X AES CBC-CS3 Key – W, X AES CBC-CS3 IV – W, X AES OFB Key – W, X AES OFB IV– W, X AES CFB 1 Key – W, X AES CFB 1 IV– W, X AES CFB 8 Key – W, X AES CFB 8 IV– W, X AES CFB 128 Key – W, X AES CFB 128 IV– W, X AES CCM key – W, X AES CCM IV- W, X AES GCM key – W, X AES GCM IV –W, X AES XTS key – W, X AES XTS IV – W, X AES CTR key – W, X	Symmetric Encryption and Decryption	Return code from function call

Name	Roles	Description	Input	Output	SSP and Type of Access	Approved Security Function	Indicator
					AES CTR IV – W, X		
Message Authentication Code	User	Generate or verify data integrity with CMAC and GMAC.	API call parameters: key, data, authenticated message digest to verify.	Status, authenticated message digest	AES CMAC key – R, X AES GMAC key – R, X	Message Authentication Code	Return code from function call
Asymmetric Key Generation	User	Generate RSA, DSA, ECDSA, ECDH, DH keys.	API call parameters: Curve, modulus, group, key length (N, L) Key pair to be verified	Status, Generated private and public key pair	RSA key pair – G, R, X DSA key pair – G, R, X ECDSA key pair – G, R, X DH key pair – G, R, X	Asymmetric Key Generation Asymmetric Key Verification Primitive Generation Primitive Verification	Return code from function call
Digital Signature	User	Generate or verify RSA, DSA, ECDSA digital signatures.	API call parameters: private key, message, signature (for verification)	Status, signature	RSA key pair – R, X DSA key pair – R, X ECDSA key pair – R, X	Digital Signature Generation, Digital Signature Verification	Return code from function call
Key Agreement	User	Perform key agreement primitives on behalf of the calling application (does not establish keys into the module)	API call parameters: private key, counter public key	Status, key components, key	ECDH Key Pair – R, X DH Key Pair – R, X RSA Key Pair – R, X Shared Secret – G	Shared Secret Computation for Key Agreement	Return code from function call
Message digest	User	Compute and return a message digest using SHS and SHA-3 algorithms	API call parameters: message	Status, hash	None	Message Digest	Return code from function call
Keyed Hash	User	Generate or verify data integrity with HMAC or KMAC.	API call parameters: HMAC/KMAC key, message, keyed hash value (for verification)	Status Keyed Hash value (Generation) True or false (Verification)	HMAC key – R, X KMAC key – R, X	Message Authentication Code	Return code from function call

Name	Roles	Description	Input	Output	SSP and Type of Access	Approved Security Function	Indicator
Key Derivation	User	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.	API call parameters: Shared Secret, salt, additional info depending on the algorithm used.	Status and derived keying material	Shared Secret – R, X Keying material – G HKDF salt – R, X PBKDF2 salt – R, X PBKDF2 password – R, X	Key Derivation Function Password-Based Key Derivation Function Key-Based Key Derivation Function	Return code from function call
Key Transport	User	Encrypt or Decrypt a key value on behalf of the calling application (does not establish keys into de module)	API call parameters: Key-encryption key, Key to be encapsulated or wrapped	Status and Encapsulated key	RSA key pair – R, X AES key – R, X HMAC key – R, X	Key Wrapping Key Encapsulating	Return code from function call
Random Number Generation	User	Used for random number and symmetric key generation using HMAC, HASH or CTR DRBG	API call parameters: number of bytes to be generated	Status and random bitstring	DRBG Entropy Input String – R, X DRBG Seed – W, X DRBG “C” value (HASH DRBG) – W, X DRBG “Key” value (HMAC and CTR) – W, X	Random Number Generation	Return code from function call

5 Software/Firmware security

The cryptographic module is composed of a software shared library as well as file where the 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.

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

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 and 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 Parameter Management

The following section includes all the information related to the Sensitive Security Parameters (SSPs) and its management. Table 10 below identifies 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.

Table 10 – Sensitive Security Parameters Definition

Key	Strength	Security Function Cert Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use & Related Keys
AES ECB, CBC, CBC-CS1, CBC-CS2, CBC-CS3, CFB1, CFB8, CFB128, OFB, CTR Keys	128, 192, 256-bit key	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Symmetric Encryption and Decryption
AES CBC, CBC-CS1, CBC-CS2, CBC-CS3, CFB1, CFB8, CFB128, OFB, CTR IVs	128 bits	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Initialization Vector
AES XTS Key	128, 256-bit key	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Symmetric Encryption and Decryption
AES XTS IV	128 bits	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Initialization Vector for AES_XTS
AES KW/KWP Key	128, 192, 256-bit key	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Key Wrapping
AES CCM Key	128, 192, 256-bit key	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Symmetric Encryption and Decryption

Key	Strength	Security Function Cert Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use & Related Keys
AES CCM IV	56, 64, 72, 80, 88, 96, 104 bits	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Initialization Vector for AES_CCM
AES GCM Key	128, 192, 256-bit key	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Authenticated encryption and decryption
AES GCM IV	96-bits value	A4198	External or internally generated	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Initialization vector for AES_GCM
AES CMAC Key	128, 192, 256-bit key	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Message Authentication
AES GMAC Key	128, 192, 256-bit key	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Message Authentication
HMAC Key	112-bits or greater	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Message Authentication
KMAC Key	128-bits or greater	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Message Authentication
DSA Public Key	1024, 2048, 3072-bit key	A4198	External or internally generated	Plaintext / Plaintext	N/A	Not persistently stored	Power cycle or EVP_PKEY_free()	Signature verification
DSA Private Key	224 and 256-bit key	A4198	External or internally generated	Plaintext / Plaintext	N/A	Not persistently stored	Power cycle or EVP_PKEY_free()	Signature generation

Key	Strength	Security Function Cert Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use & Related Keys
ECDSA Public Key	From 192-576 bits	A4198	External or internally generated	Plaintext / Plaintext	N/A	Not persistently stored	Power cycle or EVP_PKEY_free()	Signature verification
ECDSA Private Key	From 224-576 bits	A4198	External or internally generated	Plaintext / Plaintext	N/A	Not persistently stored	Power cycle or EVP_PKEY_free()	Signature generation
RSA Private Key	2048, 3072, 4096, 6144-bit key	A4198	External or internally generated	Plaintext / Plaintext	N/A	Not persistently stored	Power cycle or EVP_PKEY_free()	Signature Generation OAEP Decryption
RSA Public Key	1024, 2048, 3072, 4096, 6144-bit key	A4198	External or internally generated	Plaintext / Plaintext	N/A	Not persistently stored	Power cycle or EVP_PKEY_free()	Signature Verification OAEP Encryption
Diffie-Hellman Public Key	From 2048-bit to 8192-bit key	A4198	External or internally generated	Plaintext / Plaintext	N/A	Not persistently stored	Power cycle or EVP_PKEY_free()	Derive Shared Secret
Diffie-Hellman Private Key	From 2048-bit to 8192-bit key	A4198	External or internally generated	Plaintext / Plaintext	N/A	Not persistently stored	Power cycle or EVP_PKEY_free()	Derive Shared Secret
ECDH Public Key	From 224-576 bits	A4198	External or internally generated	Plaintext / Plaintext	N/A	Not persistently stored	Power cycle or EVP_PKEY_free()	Derive Shared Secret
ECDH Private Key	From 224-576 bits	A4198	External or internally generated	Plaintext / Plaintext	N/A	Not persistently stored	Power cycle or EVP_PKEY_free()	Derive Shared Secret
Shared Secret	Shared secret bitstring	A4198	Internally derived	NA / Plaintext	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Shared Secret for deriving keying material

Key	Strength	Security Function Cert Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use & Related Keys
HKDF salt	Salt bitstring	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Key Derivation Function
Keying material	Keying material bitstring	A4198	Internally derived	NA / Plaintext	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	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)
PBKDF2 password	From 8 to 128 characters (bytes) with increment of 8 characters.	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or EVP_KDF_free()	PBKDF2 Key derivation function
PBKDF2 salt	From 128 to 4096-bit salt bitstring with increment of 8-bits	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or EVP_KDF_free()	PBKDF2 Key derivation function

Entropy input string	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	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Entropy material for HASH DRBG/HMAC DRBG/CTR DRBG

Key	Strength	Security Function Cert Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use & Related Keys
	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							

Key	Strength	Security Function Cert Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use & Related Keys
DRBG seed	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	A4198	External	Plaintext / Never exits the module	N/A	Not persistently stored	Power cycle or OPENSSL_cleanse()	Seeding material for HASH DRBG/HMAC DRBG/CTR DRBG
DRBG 'C' value	Internal state value	A4198	Internally generated	NA / Never exits the module	N/A	Not persistently stored	Power cycle or EVP_RAND_CTX_free()	Used for HASH DRBG

Key	Strength	Security Function Cert Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use & Related Keys
DRBG 'V' value	Internal state value	A4198	Internally generated	NA / Never exits the module	N/A	Not persistently stored	Power cycle or EVP_RAND_CTX_free()	Used for CTR DRBG/HASH DRBG/HMAC DRBG
DRBG 'Key' value	128, 192, 256-bit key for CTR-DRBG 160, 224, 256, 384, 512 for HMAC-DRBG	A4198	Internally generated	NA / Never exits the module	N/A	Not persistently stored	Power cycle or EVP_RAND_CTX_free()	Used for CTR DRBG/HMAC DRBG

9.1 Random Number Generation

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

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.

9.2 SSP 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.

9.3 SSP 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:

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.
3. 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.

9.4 SSP Derivation

The module provides the following SSP Derivation methods:

- Key-Based Key Derivation (KDKDF 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.

9.5 SSP Transport

The module provides approved key transport methods according to [IG] D.G. 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 transport scheme (KTS-RSA 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.

9.6 SSP Entry/Output

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.

Information for the input and output for each SSP, if applicable, is provided under columns "Input" and "Output" of Table 10.

9.7 SSP Storage

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.

Table 11 - Storage Areas

Name	Description	Type
RAM	System Memory	Dynamic

Information for storage of each SSP, is provided under column "Storage" of Table 10.

9.8 SSP Zeroisation

SSP zeroisation functions are implemented in the *fips.so* library, and they depend on the SSP to be zeroized. Information for zeroising each SSP is provided under column “Zeroization” of Table 10.

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 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 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.

10 Self-Tests

FIPS 140-3 requires that the module performs a set of self-test in order to provide the operator assurance that faults have not been introduced that would prevent the module's correct operation.

The module performs the pre-operational self-test and conditional cryptographic algorithms self-tests during the initialization of the module. In addition, some functions require continuous testing of the cryptographic functionality, such as the asymmetric key generation.

The following sections list the self-tests performed by the module, their expected error status and error resolutions.

10.1 Pre-operational Self-Tests

The module executes the following pre-operational self-tests when it is loaded into memory, without operator intervention. These pre-operational self-tests ensure that the module is not corrupted:

- Integrity test: Using a HMAC-SHA2-256 for fips.so library, as it is described in section 5.

The following cryptographic algorithms are self-tested as part of the pre-operational self-tests:

Table 12 - Pre-operational Self-Tests

Algorithm	OE	Test Properties	Type	Details
HMAC-SHA-256	Both	128-bit hardcoded key	KAT	Integrity self-test perform at the initialization of the module in order to verify its integrity.

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.

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 this tests fails, the module enters in Critical Error state.

10.2.1 Conditional Cryptographic Algorithms 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.

These tests are detailed in the table below:

Table 13 - Conditional Cryptographic Algorithm Self-Tests

Algorithm	OE	Test Properties	Type	Details	Conditions
HMAC-SHA256	Both		KAT	Used for module integrity test	Power-Up and On-demand
SHS	Both	SHA-1, SHA2-512 and SHA3-256	KAT	Generation	Power-Up and On-demand
AES	Both	ECB mode with 128-bit key	KAT	Decrypt	Power-Up and On-demand
AES GCM	Both	256 bits key size	KAT	Encrypt and Decrypt	Power-Up and On-demand
RSA	Both	2048 bits key size with SHA2-256 and mode PKCS#1v1.5.	KAT	Sign and Verify	Power-Up and On-demand
DSA	Both	2048, 224 bits key size with SHA2-256	KAT	Sign and Verify	Power-Up and On-demand
ECDSA	Both	P-224 and K-233 curves with SHA2-256	KAT	Sign and Verify	Power-Up and On-demand
DRBG	Both	CTR_DRBG: AES 128-bit key with derivation function. HASH_DRBG: SHA2-256. HMAC_DRBG: SHA-1	KAT	Random Bit Generation	Power-Up and On-demand
KBKDF	Both	Counter Mode (HMAC-SHA2-256).	KAT	Key Derivation	Power-Up and On-demand

Algorithm	OE	Test Properties	Type	Details	Conditions
PBKDF2	Both		KAT	Derivation of the Master Key (MK) (per Section 5.3 of SP 800-132).	Power-Up and On-demand
SP 800-135rev1 KDFs	Both	TLS 1.2, SSHv2, ANSI X9.63-2001 and ANSI X9.42-2001 KDFs.	KAT	Key Derivation	Power-Up and On-demand
TLS v1.3 KDF	Both	TLS v1.3 KDF (per Section 7.1 of RFC 8446).	KAT	Key Derivation	Power-Up and On-demand
KAS-FFC-SSC	Both	ffdhe2048 safe prime group with dhEphem scheme	KAT	Shared Secret Computation (per Section 6 of SP 800-56Arev3).	Power-Up and On-demand
KAS-ECC-SSC	Both	P-256 curve with Ephemeral Unified scheme	KAT	Shared Secret Computation (per Section 6 of SP 800-56Arev3).	Power-Up and On-demand
KAS-RSA-SSC	Both	2048 bits key size	KAT	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	Both		KAT	One-step KDF (per Section 4 of SP 800-56Crev2) and Two-Step KDF (per Section 5 of SP 800-56Crev2).	Power-Up and On-demand
KTS-RSA	Both	2048 bits key size	KAT	Encrypt and Decrypt for Basic and Decrypt for CRT (per IG D.G and SP 800-56Brev2).	Power-Up and On-demand

The *SELF_TEST_kats()* API function is in charge of performing all the KAT self-tests for each algorithm prior to their first use, 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.

10.2.2 Conditional Pairwise Consistency Test

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.

Table 14 - Conditional Pair-wise Consistency Tests

Algorithm	OE	Test Properties	Type	Details	Conditions
DSA	Both	Generated Key size.	PCT	Pairwise Consistency Test on Generation of a DSA Key Pair.	Key Generation.
ECDSA	Both	Generated curve.	PCT	Pairwise Consistency Test on Generation of an ECDSA Key Pair.	Key Generation.
RSA	Both	Generated Key size.	PCT	Pairwise Consistency Test on Generation of an RSA Key Pair.	Key Generation.

10.2.3 Conditional Critical Function Tests

The module obtains the following assurances per SP 800-56Arev3 and SP 800-56Brev2:

Table 15 - Assurances

Standard	Assurances
SP 800-56Arev3	Per Section 5.6.2 of SP 800-56Arev3, required per [IG] D.F
SP 800-56Brev2	Per Section 6.4 of SP 800-56Brev2, required per [IG] D.F

The module performs the following DRBG health checks when initializing the DRBG for the first time during the power-up of the module:

- DRBG Instantiate Test
- DRBG Generate Test
- DRBG Reseed Test

If any of these DRBG health checks fails, the module will enter in a critical error state.

10.3 On-Demand 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.

10.4 Error Handling

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

Table 16 – Error Codes

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

11 Life-cycle assurance

11.1 Delivery and Operation

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 Crypto Officer and User Guidance

Crypto Officer

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.

User 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.

11.2.1 Installation

The operator shall follow the Crypto Officer instructions in section 11.2.

11.2.2 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.

11.2.3 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.

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

11.2.4 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 4. 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).

11.2.5 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 4, 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.

11.2.6 PBKDF

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

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, 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.

13 Appendix I

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

14 References and Acronyms

14.1 References

Table 17 - References

Abbreviation	Full Specification Name
FIPS 140-3	FIPS 140-3 Security Requirements for Cryptographic modules
NIST SP 800-140	FIPS 140-3 Derived Test Requirements (DTR): CMVP Validation Authority Updates to ISO/IEC 24759
ISO 19790	ISO/IEC 19790:2012/Cor.1:2015(E), Information technology — Security techniques — Security requirements for cryptographic modules
ISO 24759	ISO/IEC 24759:2017(E), Information technology — Security techniques — Test requirements for cryptographic modules
IG	Implementation Guidance for FIPS 140-3 and the Cryptographic Module Validation Program
FIPS PUB 197	FIPS 197 Advanced Encryption Standard
FIPS PUB 180-4	FIPS 180-4 Secure Hash Standard
FIPS PUB 198-1	FIPS 198-1 The Keyed-Hash Message Authentication Code (HMAC)
FIPS PUB 186-4	FIPS 186-4 Digital Signature Standard (DSS)
FIPS PUB 202	FIPS 202 SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions
NIST SP 800-38A	NIST SP 800-38A, Recommendation for Block Cipher Modes of Operation Methods and Techniques
NIST SP 800-38B	NIST SP 800-38B, Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication
NIST SP 800-38C	NIST SP 800-38C, Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality
NIST SP 800-38D	NIST SP 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC
NIST SP 800-38E	NIST SP 800-38E, Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices
NIST SP 800-38F	NIST SP 800-38F, Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping
NIST SP 800-56Arev3	NIST SP 800-56Arev3, Recommendation for Pair-Wise Key Establishment Schemes using Discrete Logarithm Cryptography
NIST SP 800-56Brev2	NIST SP 800-56Brev2, Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography
NIST SP 800-56Crev2	NIST SP 800-56Crev2, Recommendation for Key-Derivation Methods in Key-Establishment Schemes

Abbreviation	Full Specification Name
NIST SP 800-57 Part 1 Rev5	NIST SP 800-57 Part 1, Rev5, Recommendation for Key Management: Part 1 – General
NIST SP 800-90Arev1	NIST SP 800-90Arev1, Recommendation for Random Number Generation Using Deterministic Random Bit Generators
NIST SP 800-108rev1	NIST SP 800-108rev1, Recommendation for Key Derivation Using Pseudorandom Functions
NIST SP 800-131Arev2	NIST SP 800-131Arev2, Transitioning the Use of Cryptographic Algorithms and Key Lengths
NIST SP 800-132	NIST SP 800-132, Recommendation for Password-Based Key Derivation: Part 1: Storage Applications
NIST SP 800-133rev2	NIST SP 800-133rev2, Recommendation for Cryptographic Key Generation
NIST SP 800-135rev1	NIST SP 800-135rev1, Recommendation for Existing Application-Specific Key Derivation Functions
NIST SP 800-185	NIST SP 800-185, SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash
RFC 7627	Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension

14.2 Acronyms

Table 18 - Acronyms definitions

Acronym	Definition
AES	Advanced Encryption Standard
API	Application Program Interface
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CKG	Cryptographic Key Generation
CMVP	Cryptographic Module Validation Program
CTR	Counter Mode
CVL	Component Validation List
DH	Diffie-Hellman
DSA	Digital Signature Algorithm
DRGB	Deterministic Random Bit Generator
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
FFC	Finite Field Cryptographic
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode

Acronym	Definition
GPC	General Purpose Computer
HMAC	Hash Message Authentication Code
KAS	Key Agreement Scheme
KAT	Known Answer Test
KDF	Key Derivation Function
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
PCT	Pair-wise Consistency Test
RSA	Rivest, Shamir, Adleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SSC	Shared Secret Computation
TLS	Transport Layer Security