JUNOS® OS EVOLVED OPENSSL CRYPTOGRAPHIC MODULE VERSION 3.0.8

FIPS 140-3 NON-PROPRIETARY SECURITY POLICY

Document Version 1.2

Last update: 2024-08-23

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

Prepared for:

Juniper Networks, Inc.

1133 Innovation Way

Sunnyvale, CA 94089

www.juniper.net

Table of Contents

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for version 3.0.8 of the Junos® OS Evolved OpenSSL Cryptographic Module. It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-3 (Federal Information Processing Standards Publication 140-3) for a Security Level 1 software module. It has a one-to-one mapping to the [SP800-140B] starting with section B.2.1 named "General" that maps to section 1 in this document and ending with section B.2.12 named "Mitigation of other attacks" that maps to section 12 in this document.

This Non-Proprietary Security Policy may be reproduced and distributed, but only whole and intact and including this notice. Other documentation is proprietary to their authors.

1.2 How this Security Policy was Prepared

The vendor has provided the non-proprietary Security Policy of the cryptographic module, which was further consolidated into this document by atsec information security together with other vendor-supplied documentation. In preparing the Security Policy document, the laboratory formatted the vendor-supplied documentation for consolidation without altering the technical statements therein contained. The further refining of the Security Policy document was conducted iteratively throughout the conformance testing, wherein the Security Policy was submitted to the vendor, who would then edit, modify, and add technical contents. The vendor would also supply additional documentation, which the laboratory formatted into the existing Security Policy, and resubmitted to the vendor for their final editing.

1.3 Security Levels

The following sections describe the cryptographic module and how it conforms to the FIPS 140-3 specification in each of the required areas.

Table 1 - Security Levels

ISO/IEC 24759 Section 6. [Number Below]	FIPS 140-3 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

2 Cryptographic Module Specification

2.1 Module Embodiment

The Junos® OS Evolved OpenSSL Cryptographic Module (hereafter referred to as "the module") is defined as a software module in a multi-chip standalone embodiment. It provides a C language application program interface (API) for use by other applications that require cryptographic functionality. The module consists of one software component, the "FIPS provider" (i.e. fips.so), which implements the FIPS requirements and the cryptographic functionality provided to the operator.

2.2 Tested Operational Environments

The module has been tested on the following platforms with the corresponding module variants and configuration options:

Table 2 - Tested Operational Environments

#	Operating System	Hardware Platform	Processor	PAA/Acceleration	Module Version Tested
1	Junos® OS Evolved 22.4	Juniper Networks® Packet Transport Router Model PTX10001-36MR	Intel® Xeon® D-2163IT	AES-NI, SHA Extensions (PAA)	3.0.8

2.3 Approved Algorithms

Table 3 lists all the approved security functions of the module, including specific key strengths employed for approved services.

Table 3 - Approved Algorithms

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use
#A4246, #A4247, #A4248, #A4249	SHS [FIPS180-4]	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	N/A	Message Digest
#A4236	SHA-3 [FIPS202]	SHA3-224, SHA3-256, SHA3-384, SHA3-512	N/A	Message Digest
#A4236	SHA-3 [FIPS202]	SHAKE-128, SHAKE-256	N/A	XOF
#A4236	KMAC [SP800-185]	KMAC-128, KMAC-256	128, 256 bits with 128, 256 bits of security strength	MAC Generation and Verification
#A4229, #A4230, #A4231	AES [FIPS197], [SP800-38A]	CBC, CTR, CFB1, CFB8, CFB128, OFB, CBC-CTS-CS1, CBC-CTS-CS2, CBC-CTS-CS3	128, 192, 256 bits with 128, 192, 256 bits of security strength	Symmetric Encryption and Decryption
#A4229, #A4230, #A4231	AES [FIPS197], [SP800-38B]	CMAC	128, 192, 256 bits with 128, 192, 256 bits of security strength	MAC Generation and Verification

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use
	AES [FIPS197], [SP800-38C]	CCM	128, 192, 256 bits with 128, 192, 256 bits of security strength	Symmetric Encryption and Decryption
#A4229, #A4230, #A4231, #A4232, #A4233, #A4234, #A4235	AES [FIPS197], [SP800-38A]	ECB	128, 192, 256 bits with 128, 192, 256 bits of security strength	Symmetric Encryption and Decryption
#A4237, #A4238, #A4239, #A4240, #A4241, #A4242, #A4243, #A4244, #A4245	AES [FIPS197], [SP800-38D]	GCM (external IV)	128, 192, 256 bits with 128, 192, 256 bits of security strength	Symmetric Decryption
	AES [FIPS197], [SP800-38D]	GCM (internal IV)	128, 192, 256 bits with 128, 192, 256 bits of security strength	Symmetric Encryption
	AES [FIPS197], [SP800-38D]	GMAC	128, 192, 256 bits with 128, 192, 256 bits of security strength	MAC Generation MAC Verification
#A4229, #A4230, #A4231	AES [FIPS197], [SP800-38E]	XTS	128, 256 bits with 128, 256 bits of security strength	Symmetric Encryption and Decryption for Data Storage
	AES [FIPS197], [SP800-38F]	KW , KWP (KTS)	128, 192, 256 bits with 128, 192, 256 bits of security strength	Key Wrapping and Unwrapping
#A4246, #A4247, #A4248, #A4249	HMAC [FIPS198-1]	SHA-1, SHA2-224, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	112-524288 bits with 112 to 256 bits of security strength	MAC Generation MAC Verification
#A4236		SHA3-224, SHA3-256, SHA3-384, SHA3-512		
#A4246, #A4247, #A4248, #A4249		SHA2-256		MAC Generation MAC Verification Integrity Check
#A4250	KBKDF [SP 800-108r1]	Counter and feedback mode, using CMAC and HMAC SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	112-4096 bits with 112 to 256 bits of security strength	KBKDF Key Derivation
#A4224	KDA OneStep[1]	(HMAC) SHA-1, SHA2-224, SHA2-256, SHA2-384,	224 to 8192 bits with 112 to 256 bits of security strength	KDA OneStep Key Derivation

[1]This algorithm is referred to as “Single Step KDF” or “SSKDF” by OpenSSL.

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use
	[SP 800-56Cr2]	SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 KMAC-128, KMAC-256		
#A4228	KDA HKDF [SP 800-56Cr2]	HMAC with SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384	224 to 8192 bits with 112 to 256 bits of security strength	HKDF Key Derivation
CVL #A4236, #A4246, #A4247, #A4248, #A4249	ANS 9.42 KDF [SP 800-135r1]	ANS 9.42 KDF AES KW with SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	224 to 8192 bits with 112 to 256 bits of security strength	ANS X9.42 KDF Key Derivation
CVL #A4246, #A4247, #A4248, #A4249	ANS 9.63 KDF [SP 800-135r1]	ANS 9.63 KDF SHA2-224, SHA2-256, SHA2-384, SHA2-512	224 to 8192 bits with 112 to 256 bits of security strength	ANS 9.63 KDF Key Derivation
CVL #A4232, #A4233, #A4234, #A4235	SSH KDF [SP 800-135r1]	SSH KDF SHA-1, SHA2-256, SHA2-384, SHA2-512	224 to 8192 bits with 112 to 256 bits of security strength	SSH KDF Key Derivation
CVL #A4246, #A4247, #A4248, #A4249	TLS 1.2 KDF [SP 800-135r1]	TLS v1.2 KDF with SHA2-256, SHA2-384, SHA2-512	224 to 8192 bits with 112 to 256 bits of security strength	TLS 1.2 KDF Key Derivation
CVL #A4228	TLS 1.3 KDF [RFC 8446]	TLS v1.3 KDF with SHA2-256, SHA2-384	224 to 8192 bits with 112 to 256 bits of security strength	TLS 1.3 KDF Key Derivation
#A4236, #A4246, #A4247, #A4248, #A4249	PBKDF2 [SP 800-132]	Option 1a with SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	N/A	Password-based Key Derivation
#A4251	KAS-FFC-SSC [SP 800-56Ar3]	dhEphem (initiator/responder)	MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 with 112 to 200 bits of security strength	DH shared secret Computation

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use
#A4227, #A4246, #A4247, #A4248, #A4249	KAS-ECC-SSC [SP 800-56Ar3]	Ephemeral Unified Model (initiator/responder)	B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 with 112, 128, 192, 256 bits of security strength	ECDH shared secret Computation
#A4246, #A4247, #A4248, #A4249 Modulus sizes other than 1024, 2048, 3072 and 4096 bits are not CAVP tested but approved per IG C.F	RSA [FIPS186-4]	B.3.6 Probable Primes	2048-15360 bits with 112 to 256 bits of security strength	Key Pair Generation
		PKCS#1v1.5: SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	2048-16384 bits with 112 to 256 bits of security strength	Digital Signature Generation
		PSS: SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256		
		PKCS#1v1.5: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	1024-16384 bits with 80 to 256 bits of security strength	Digital Signature Verification
		PSS: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256		
#A4225, #A4246, #A4247, #A4248, #A4249	ECDSA [FIPS186-4]	FIPS 186-4 Appendix B.4.2 Testing Candidates	B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 with 112, 128, 192, 256 bits of security strength	Key Pair Generation
		N/A		Key Pair Verification
#A4225, #A4226, #A4236, #A4246, #A4247, #A4248, #A4249		SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512		Signature Generation
		SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512		Signature Verification
#A4251	Safe primes [SP 800-56Ar3]	[SP 800-56Ar3] Section 5.6.1.1.4 Testing Candidates	MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 with	Key Pair Generation
	Safe primes [SP 800-56Ar3]	[SP 800-56Ar3] Sections 5.6.2.1.2 and 5.6.2.1.4		Key pair Verification

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use
			112 to 200 bits of security strength	
Vendor Affirmed	CKG [SP 800-133r2 Section 4]	Safe primes	MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 with 112 to 200 bits of security strength	Key Pair Generation
		RSA	2048-15360 bits with 112 to 256 bits of security strength	
		ECDSA	B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 with 112, 128, 192, 256 bits of security strength	
	RSA [FIPS 186-4]	PKCS#1v1.5: SHA3-224, SHA3-256, SHA3-384, SHA3-512	2048-16384 bits with 112 to 256 bits of security strength	Signature Generation
		PSS: SHA3-224, SHA3-256, SHA3-384, SHA3-512		
		PKCS#1v1.5: SHA3-224, SHA3-256, SHA3-384, SHA3-512	1024-16384 bits with 80 to 256 bits of security strength	Signature Verification
		PSS: SHA3-224, SHA3-256, SHA3-384, SHA3-512		
Kernel Bound Module				
#A3599 #A3600 #A3601 #A3603 #A3604 #A3605	Hash_DRBG [SP 800-90Ar1]	SHA-1, SHA2-256, SHA2-512 with/without PR	N/A	Random Number Generation
#A3599 #A3600 #A3601 #A3603 #A3604 #A3605	HMAC_DRBG [SP 800-90Ar1]	SHA-1, SHA2-256, SHA2-512 with/without PR	N/A	
#A3599 #A3600 #A3601 #A3602	CTR_DRBG [SP 800-90Ar1]	AES-128, AES-192, AES-256 with DF, with/without PR	128, 192, 256 bits with 128, 192 and 256 bits of security strength	

2.4 Non-Approved Algorithms Allowed in the Approved Mode of Operation

The module does not implement non-approved algorithms that are allowed in the approved mode of operation.

2.5 Non-Approved Algorithms Allowed in the Approved Mode of Operation with No Security Claimed

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

2.6 Non-Approved Algorithms Not Allowed in the Approved Mode of Operation

Table 4 - Non-Approved Algorithms Not Allowed in the Approved Mode of Operation

Algorithm / Functions	Use / Function
AES GCM (external IV)	Symmetric Encryption

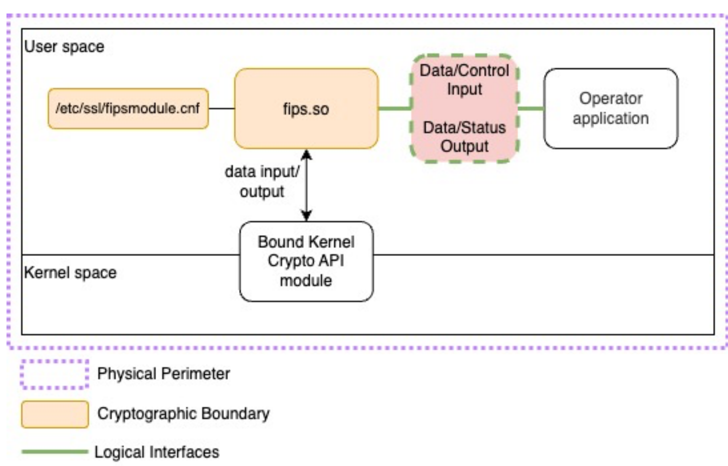
2.7 Module Design and Components

Figure 1 shows a block diagram that represents the design of the module when the module is operational and providing services to other user space applications. In this diagram, the physical perimeter of the operational environment (a general-purpose computer on which the module is installed) is indicated by a purple dashed line. The cryptographic boundary is represented by a shared library implementing the FIPS provider (fips.so) along with its HMAC value which resides within a configuration file called /etc/ssl/fipsmodule.cnf.

Green lines indicate the flow of data between the cryptographic module and its operator application, through the logical interfaces defined in Section 3.

Components in white are only included in the diagram for informational purposes. They are not included in the cryptographic boundary (and therefore not part of the module's validation). For

Figure 1 – Software Block Diagram



example, the kernel is responsible for managing system calls issued by the module itself, as well as other applications using the module for cryptographic services.

The module also uses the Junos OS Evolved Kernel Cryptographic Module Version 2.0 as a bound module (also referred to as "the bound Kernel Crypto API module") for performing random number generation, relying on the Kernel DRBG. The Junos OS Evolved Kernel Cryptographic Module Version 2.0 is a FIPS 140-3 validated module with certificate #4776.

2.8 Rules of Operation

In the operational state, the module accepts service requests from calling applications through its logical interfaces. At any point in the operational state, a calling application can end its process, thus causing the module to end its operation.

The module supports two modes of operation:

- The approved mode of operation, in which the approved or vendor affirmed services are available as specified in Table 7.
- The non-approved mode of operation, in which the non-approved services are available.

3 Cryptographic Module Interfaces

All data output via data output interface is inhibited when the module is performing pre-operational test or zeroization or when the module enters error state.

Table 5 summarizes the logical interfaces:

Table 5 - Ports and Interfaces

Physical Ports	Logical Interface	Data that passes over port/interface
As a software-only module, the module does not have physical ports. The operator can only interact with the module through the API provided by the module. Thus, the physical ports are interpreted to be the physical ports of the hardware platform on which the module runs	**Data Input**	API input parameters
	Data Output	API output parameters
	Control Input	API function calls
	Status Output	API return codes, error messages
	Power Input	N/A

The module does not implement a control output interface.

4 Roles, Services and Authentication

4.1 Roles

The module supports the Crypto Officer role only. This sole role is implicitly and always assumed by the operator of the module. No support is provided for a maintenance role.

Table 6 lists the roles supported by the module with corresponding services with input and output parameters.

Table 6 - Role, Service Commands, Input and Output

Role	Service	Input	Output
Crypto Officer	Message Digest	Message	Digest value
	XOF	Message, output length	Digest value
	Symmetric Encryption	Plaintext, AES key	Ciphertext
	Symmetric Decryption	Ciphertext, AES key	Plaintext
	MAC Generation	Message, AES key, KMAC, or HMAC key	MAC tag
	MAC Verification	Message, AES key, KMAC, or HMAC key, MAC tag	Pass/fail
	KBKDF Key Derivation	Key-derivation key	KBKDF derived key
	KDA OneStep Key Derivation	DH or ECDH shared secret	KDA OneStep derived key
	HKDF Key Derivation	DH or ECDH shared secret	HKDF derived key
	ANS X9.42 KDF Key Derivation	DH or ECDH shared secret	ANS X9.42 KDF derived key
	ANS X9.63 KDF Key Derivation	DH or ECDH shared secret	ANS X9.63 KDF derived key
	SSH KDF Key Derivation	DH or ECDH shared secret	SSH KDF derived key
	TLS 1.2 KDF Key Derivation	DH or ECDH shared secret	TLS 1.2 KDF derived key
	TLS 1.3 KDF Key Derivation	DH or ECDH shared secret	TLS 1.3 KDF derived key
	Password-based Key Derivation	Password, salt, iteration count	PBKDF2 derived key
	Key Unwrapping	Key wrapping key, wrapped key	Unwrapped key
	Key Wrapping	Key wrapping key, key to be wrapped	Wrapped key
	Random Number Generation	Output length	Random bytes
	DH Shared Secret Computation	Owner private key, peer public key	DH shared secret
	ECDH Shared Secret Computation	Owner private key, peer public key	ECDH shared secret

Role	Service	Input	Output
	Signature Generation	Message, private key	Signature
	Signature Verification	Message, public key, signature	Pass/fail
	Key pair Generation	Key size	Key pair
	Key pair Verification	Key pair	Pass/fail
	Show Version	N/A	Name and version information
	Show Status	N/A	Module status
	Self-test	N/A	Pass/fail results of self-tests
	Zeroization	Any SSP	N/A

4.2 Authentication

The module does not support authentication for roles.

4.3 Services

The module provides services to operators that assume the available role. All services are described in detail in the API documentation (manual pages). The next tables define the services that utilize approved and non-approved security functions in this module. For the respective tables, the convention below applies when specifying the access permissions (types) that the service has for each SSP.

- **G = Generate**: The module generates or derives the SSP.
- **R = Read**: The SSP is read from the module (e.g., the SSP is output).
- **W = Write**: The SSP is updated, imported, or written to the module.
- **E = Execute**: The module uses the SSP in performing a cryptographic operation.
- **Z = Zeroize**: The module zeroizes the SSP.
- **n/a**: the calling application does not access any SSP or key during its operation.

To interact with the module, a calling application must use the EVP API layer provided by OpenSSL. This layer will delegate the request to the FIPS provider, which will in turn perform the requested service.

Table 7 lists the approved services in this module, the algorithms involved, the Sensitive Security Parameters (SSPs) involved and how they are accessed, and the roles that can request the service. In this table, CO specifies the Crypto Officer role.

Table 7 – Approved Services

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator
			Cryptographic Library Services			
Message Digest	Compute a message digest	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	N/A	CO	N/A	EVP_Digest* functions will return 0
XOF	Compute the output of an XOF	SHAKE-128, SHAKE-256	N/A	CO	N/A	EVP_Digest* functions will return 0
Symmetric Encryption	Encrypt a plaintext	AES ECB, CBC, CBC-CTS-CS1, CBC-CTS-CS2, CBC-CTS-CS3, CFB1, CFB8, CFB128, CTR, OFB, CCM , XTS	AES key	CO	W, E	EVP_Encrypt* functions will return 0
Symmetric Encryption (AES-GCM)	Encrypt a plaintext	AES GCM	AES key	CO	W, E	ERR_peek_last_error function returns something different from 0x1C80012C
Symmetric Decryption	Decrypt a ciphertext	AES ECB, CBC, CBC-CTS-CS1, CBC-CTS-CS2, CBC-CTS-CS3, CFB1, CFB8, CFB128, CTR, OFB, CCM, GCM, XTS	AES key	CO	W, E	EVP_Decrypt* functions will return 0
Key Wrapping	Perform AES-based key wrapping	AES-KW, AES-KWP	AES key wrapping key	CO	W,E	EVP_Encrypt* functions will return 0
Key Unwrapping	Perform AES-based key unwrapping	AES-KW, AES-KWP	AES key wrapping key	CO	W,E	EVP_Decrypt* functions will return 0
MAC Generation	Compute a MAC tag	AES CMAC AES GMAC KMAC	AES Key, KMAC key, HMAC key	CO	W, E	EVP_MAC* functions will return 0

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator
MAC Verification	Verify a MAC tag	HMAC SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512				EVP_MAC* functions will return 0
KBKDF Key Derivation	Derive a key from a key-derivation key	KBKDF	Key-derivation key	CO	W, E	EVP_KDF* functions will return 0
			KBKDF derived key		G, R	
KDA OneStep Key Derivation	Derive a key from a shared secret	KDA OneStep	DH shared secret		W, E	EVP_KDF* functions will return 0
			ECDH shared secret		W, E	
			KDA OneStep derived key		G, R	
HKDF Key Derivation	Derive a key from a shared secret	HKDF	DH shared secret		W, E	EVP_KDF* functions will return 0
			ECDH shared secret		W, E	
			HKDF derived key		G, R	
ANS X9.42 KDF Key Derivation	Derive a key from a shared secret	ANS X 9.42 KDF	DH shared secret		W, E	EVP_KDF* functions will return 0
			ECDH shared secret		W, E	
			ANS X9.42 KDF derived key		G, R	
ANS X9.63 KDF Key Derivation	Derive a key from a shared secret	ANS X 9.63 KDF	DH shared secret		W, E	EVP_KDF* functions will return 0
			ECDH shared secret		W, E	
			ANS X9.63 KDF derived key		G, R	
SSH KDF Key Derivation	Derive a key from a shared secret	SSH KDF	DH shared secret		W, E	EVP_KDF* functions will return 0
			ECDH shared secret		W, E	
			SSH KDF derived key		G, R	
TLS 1.2 KDF Key Derivation	Derive a key from a shared secret	TLS 1.2 KDF	DH shared secret		W, E	EVP_KDF* functions will return 0
			ECDH shared secret		W, E	
			TLS 1.2 KDF derived key		G, R	
TLS 1.3 KDF Key Derivation		TLS 1.3 KDF	DH shared secret		W, E	EVP_KDF* functions will return 0
			ECDH shared secret		W, E	
			TLS 1.3 KDF derived key		G, R	
Password-based Key Derivation	Derive a key from a password	PBKDF2	Password	CO	W, E	EVP_KDF* functions will return 0
			PBKDF2 derived key		G, R	
	Compute a shared secret	KAS-FFC-SSC	DH private key (owner), DH public key (peer)	CO	W, E	

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator
DH Shared Secret Computation			DH shared secret		G, R	EVP_PKEY* functions will return 0
ECDH Shared Secret Computation		KAS-ECC-SSC	EC private key (owner), EC public key (peer)		W, E	
			ECDH shared secret		G, R	
Signature Generation	Generate a signature	RSA signature generation (PKCS#1 v1.5 and PSS) ECDSA signature generation	RSA private key EC private key	CO	W, E	EVP_DigestSign* functions will return 0
Signature Verification	Verify a signature	RSA signature verification (PKCS#1 v1.5 and PSS) ECDSA signature verification	RSA public key EC public key	CO	W, E	EVP_Digestverify* functions will return 0
Key pair Generation	Generate a key pair	CKG Safe primes key pair generation RSA key pair generation ECDSA key pair generation	DH private key, DH public key	CO	G, R	EVP_PKEY* will return 0
			RSA private key, RSA public key			
			EC private key, EC public key			
Key pair Verification	Verify a key pair	Safe primes key pair verification ECDSA key pair verification	DH private key, DH public key	CO	W	EVP_PKEY* will return 0
			EC private key, EC public key			
Random Number Generation	Generate random bytes	CTR_DRBG from Kernel bound module	Entropy input	CO	W, E	RAND_bytes* will return 0
			DRBG seed		E, G	
			DRBG internal state (V, Key)		W, E, G	
		Hash_DRBG from Kernel bound module	Entropy input		W, E	
			DRBG seed		E, G	
			DRBG internal state (V, C)		W, E, G	
		HMAC_DRBG from Kernel bound module	Entropy input		W, E	
			DRBG seed		E, G	
			DRBG internal state (V, C)		W, E, G	
Show version	Return the name and	N/A	None	CO	N/A	N/A

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator
	version information					
Show status	Return the module status	N/A	None	CO	N/A	N/A
Self-test	Perform the CASTs and integrity test	SHA-1, SHA2-512, SHA3-256, SHA3-512, AES ECB, AES GCM, HMAC, KBKDF, ANS X9.42 KDF, ANS X9.63 KDF, SSH KDF, TLS 1.2 KDF, TLS 1.3 KDF, PBKDF2, KAS-FFC-SSC, KAS-ECC-SSC, Diffie-Hellman, RSA (PKCS#1 v1.5) ECDSA See Table 10 for specifics	None	CO	N/A	N/A
Zeroization	Zeroize CSPs	N/A	All SSPs	CO	Z	N/A

Table 8 - Non-Approved Services

Service	Description	Algorithm Accessed	Roles	Indicator
Symmetric Encryption	Encrypt a plaintext	AES GCM (external IV)	CO	ERR_peek_last_error() function returns 0x1C80012C

5 Software/Firmware security

5.1 Integrity Techniques

The integrity of the module is verified by comparing a HMAC SHA2-256 value calculated at run time with the HMAC SHA2-256 value stored in the /etc/ssl/fipsmodule.cnf file that was computed during installation of the module.

5.2 On-Demand Integrity Test

Integrity tests are performed as part of the pre-operational self-tests, which are executed when the module is initialized. The integrity test may be invoked on-demand by unloading and subsequently re-initializing the module, or by calling the OSSL_PROVIDER_self_test function. This will perform (among others) the software integrity test.

6 Operational Environment

6.1 Applicability

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 executing on the hardware specified in Table 2.

6.2 Policy and Requirements

The module shall be installed as stated in Section 11. If properly installed, the operating system provides process isolation and memory protection mechanisms that ensure appropriate separation for memory access among the processes on the system. Each process has control over its own data and uncontrolled access to the data of other processes is prevented.

The module does not support concurrent operators .

The module does not have the capability of loading software or firmware from an external source.

Instrumentation tools like the ptrace system call, gdb and strace utilities, userspace live patching, as well as other tracing mechanisms offered by the Linux environment such as ftrace or systemtap, shall not be used in the operational environment. The use of any of these tools implies that the cryptographic module is running in a non-tested operational environment.

7 Physical Security

The module is comprised of software only, and therefore this section is not applicable.

8 Non-invasive Security

This module does not implement any non-invasive security mechanism and therefore this section is not applicable.

9 Sensitive Security Parameter Management

Table 9 summarizes the Sensitive Security Parameters (SSPs) that are used by the cryptographic services implemented in the module in the approved services (Table 7). Notice that the table does not include SSPs related to the Random Number Generation service, which is implemented in the bound Kernel Crypto API module.

Table 9 - SSPs

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & Related Keys
AES key	128, 192, 256 bits	AES AES CMAC AES GMAC #A4229, #A4230, #A4231, #A4232, #A4233, #A4234, #A4235, #A4237, #A4238, #A4239, #A4240, #A4241, #A4242, #A4243, #A4244, #A4245	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	EVP_CIPHER_CTX_free EVP_MAC_CTX_free	**Use:** Symmetric Encryption, Symmetric Decryption, MAC Generation, MAC Verification, Key Wrapping, Key Unwrapping **Related SSPs:** N/A
HMAC key	112-256 bits	HMAC #A4236, #A4246, #A4247, #A4248, #A4249	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	EVP_MAC_CTX_free	**Use:** MAC Generation, MAC Verification **Related SSPs:** N/A
KMAC key	128-256 bits	KMAC #A4236	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	EVP_MAC_CTX_free	**Use:** MAC Generation, MAC Verification **Related SSPs:** N/A

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & Related Keys
Key-derivation key	112-256 bits	KBKDF #A4250	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	EVP_KDF_CTX_free	**Use:** KBKDF Key Derivation **Related SSPs:** KBKDF derived key
DH shared secret	112-256 bits	KAS-FFC-SSC KDA OneStep HKDF ANS 9.42 KDF ANS 9.63 KDF SSH KDF TLS 1.2 KDF TLS 1.3 KDF #A4224, #A4227, #A4228, #A4232, #A4233, #A4234, #A4235, #A4236, #A4246, #A4247, #A4248, #A4249, #A4251	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	Computed during DH shared secret computation per [SP800-56Ar3]	RAM	EVP_KDF_CTX_free	**Use:** DH Shared Secret Computation, KDA OneStep Key Derivation, HKDF Key Derivation, ANS X9.42 KDF Key Derivation, ANS X9.63 KDF Key Derivation, SSH KDF Key Derivation, TLS 1.2 KDF Key Derivation, TLS 1.3 KDF Key Derivation **Related SSPs:** DH private key, DH public key, KDA OneStep derived key, HKDF derived key, ANS X9.42 KDF derived key, ANS X9.63 KDF derived key, SSH KDF derived key, TLS 1.2 KDF derived key, TLS 1.3 KDF derived key

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & Related Keys
ECDH shared secret	112-256 bits	KAS-ECC-SSC KDA OneStep HKDF ANS 9.42 KDF ANS 9.63 KDF SSH KDF TLS 1.2 KDF TLS 1.3 KDF #A4224, #A4227, #A4228, #A4232, #A4233, #A4234, #A4235, #A4236, #A4246, #A4247, #A4248, #A4249, #A4251	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	Computed during ECDH shared secret computation per [SP800-56Ar3]	RAM	EVP_KDF_CTX_free	**Use:** ECDH Shared Secret Computation, KDA OneStep Key Derivation, HKDF Key Derivation, ANS X9.42 KDF Key Derivation, ANS X9.63 KDF Key Derivation, SSH KDF Key Derivation, TLS 1.2 KDF Key Derivation, TLS 1.3 KDF Key Derivation **Related SSPs:** ECDH private key, ECDH public key, KDA OneStep derived key, HKDF derived key, ANS X9.42 KDF derived key, ANS X9.63 KDF derived key, SSH KDF derived key, TLS 1.2 KDF derived key, TLS 1.3 KDF derived key
Password	N/A	PBKDF2 #A4236, #A4246, #A4247, #A4248, #A4249	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	MD/EE	RAM	EVP_KDF_CTX_free	**Use:** Password-based Key Derivation **Related SSPs:** PBKDF2 derived key

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & Related Keys
KBKDF derived key	112-256 bits	KBKDF #A4250	[SP 800-133r2], Section 6.2	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EVP_KDF_CTX_free	**Use:** KBKDF Key Derivation **Related SSPs:** Key-derivation key
KDA OneStep derived key	112-256 bits	KDA OneStep #A4224	[SP 800-133r2], Section 6.2	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EVP_KDF_CTX_free	**Use:** KDA OneStep Key Derivation **Related SSPs:** DH shared secret, ECDH shared secret
HKDF derived key	112-256 bits	KBKDF #A4228	[SP 800-133r2], Section 6.2	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EVP_KDF_CTX_free	**Use:** HKDF Key Derivation **Related SSPs:** DH shared secret, ECDH shared secret
ANS X9.42 KDF derived key	112-256 bits	ANS X9.42 KDF #A4236 #A4246 #A4247 #A4248 #A4249	[SP 800-133r2], Section 6.2	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EVP_KDF_CTX_free	**Use:** ANS X9.42 KDF Key Derivation **Related SSPs:** DH shared secret, ECDH shared secret
ANS X9.63 KDF derived key	112-256 bits	ANS X9.63 KDF #A4246 #A4247 #A4248 #A4249	[SP 800-133r2], Section 6.2	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EVP_KDF_CTX_free	**Use:** ANS X9.63 KDF Key Derivation **Related SSPs:** DH shared secret, ECDH shared secret

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & Related Keys
SSH KDF derived key	112-256 bits	SSH KDF #A4246 #A4247 #A4248 #A4249	[SP 800-133r2], Section 6.2	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EVP_KDF_CTX_free	**Use:** SSH KDF Key Derivation **Related SSPs:** DH shared secret, ECDH shared secret
TLS 1.2 KDF derived key	112-256 bits	TLS 1.2 KDF #A4246 #A4247 #A4248 #A4249	[SP 800-133r2], Section 6.2	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EVP_KDF_CTX_free	**Use:** TLS 1.2 Key Derivation **Related SSPs:** DH shared secret, ECDH shared secret
TLS 1.3 KDF derived key	112-256 bits	TLS 1.3 KDF #A4228	[SP 800-133r2], Section 6.2	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EVP_KDF_CTX_free	**Use:** TLS 1.3 KDF Key Derivation **Related SSPs:** DH shared secret, ECDH shared secret
PBKDF2 derived key	112-256 bits	PBKDF2 #A4236 #A4246 #A4247 #A4248 #A4249	[SP 800-132], Section 6.2	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EVP_KDF_CTX_free	**Use:** Password-based Key Derivation **Related SSPs:** Password
DH private key	112-200 bits	KAS-FFC-SSC #A4251	[SP 800-56Ar3] Section 5.6.1.1.4 Testing Candidates	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format.	N/A	RAM	EVP_PKEY_free	**Use:** DH Shared Secret Computation, Key pair verification **Related SSPs:** DH shared secret
DH public key	112-200 bits			**Export:** CM to TOEPP Path.				

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & Related Keys
				Passed from the module via API parameters in plaintext (P) format.				
EC private key	112, 128, 192, 256 bits	KAS-ECC-SSC ECDSA #A4227, #A4246, #A4247, #A4248, #A4249	FIPS 186-4 Appendix B.4.2 Testing Candidates	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EVP_PKEY_free	**Use:** ECDH Shared Secret Computation Signature Generation Key pair Verification **Related SSPs:** ECDH shared secret
EC public key	112, 128, 192, 256 bits							
RSA private key	112-256 bits	RSA #A4246, #A4247, #A4248, #A4249	FIPS 186-4 Appendix B.3.6 Probable Primes with Conditions Based on Auxiliary Probable Primes	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EVP_PKEY_free	**Use:** Signature Generation **Related Keys:** N/A
RSA public key	80-256 bits							**Use:** Signature Verification **Related SSPs:** N/A

9.1 Random Bit Generators

The module does not implement any random number generator. Instead, it uses the Random Number Generation (RNG) service provided by the bound Kernel Crypto API module, which implements a Deterministic Random Bit Generator (DRBG) based on [SP 800-90Ar1].

The module generates SSPs (e.g., keys) whose strengths are modified by available entropy.

9.2 SSP Generation

The module implements Cryptographic Key Generation (CKG, vendor affirmed), compliant with [SP 800-133r2]. When random values are required, they are obtained from the [SP 800-90Ar1] approved DRBG, compliant with Section 4, example 1 of [SP 800-133r2]. The following methods are implemented:

- Safe primes key pair generation: compliant with [SP 800-133r2], Section 5.2, which maps to [SP 800-56Ar3]. The method described in Section 5.6.1.1.4 of [SP 800-56Ar3] ("Testing Candidates") is used.
- RSA key pair generation: compliant with [SP 800-133r2], Section 5.1, which maps to FIPS 186-4. The method described in Appendix B.3.6 of FIPS 186-4 ("Probable Primes with Conditions Based on Auxiliary Probable Primes") is used. Keys other than 2048, 3072 and 4096 bits are not CAVP tested but approved per IG C.F.
- ECC (ECDH and ECDSA) key pair generation: compliant with [SP 800-133r2], Section 5.1, which maps to FIPS 186-4. The method described in Appendix B.4.2 of FIPS 186-4 ("Rejection Sampling") is used.

Additionally, the module implements the following key derivation methods:

- KBKDF: compliant with [SP 800-108r1]. This implementation can be used to generate secret keys from a pre-existing key-derivation-key.
- KDA OneStep, HKDF: compliant with [SP 800-56Cr2]. These implementations shall only be used to generate secret keys in the context of an [SP 800-56Ar3] key agreement scheme.
- ANS X9.42 KDF, ANS X9.63 KDF: compliant with [SP 800-135r1]. These implementations shall only be used to generate secret keys in the context of an ANSI X9.42-2001 resp. ANSI X9.63-2001 key agreement scheme.
- SSH KDF, TLS 1.2 KDF, TLS 1.3 KDF: compliant with [SP 800-135r1]. These implementations shall only be used to generate secret keys in the context of the SSH, TLS 1.2, or TLS 1.3 protocols, respectively.
- PBKDF2: compliant with option 1a of [SP 800-132]. This implementation shall only be used to derive keys for use in storage applications.

Intermediate key generation values are not output from the module and are explicitly zeroized after processing the service.

9.3 SSP Establishment

The module provides Diffie-Hellman (DH) and Elliptic Curve Diffie-Hellman (ECDH) shared secret computation compliant with [SP 800-56Ar3], in accordance with scenario 2 (1) of FIPS 140-3 IG D.F.

For Diffie-Hellman, the module supports the use of the safe primes defined in RFC 3526 (IKE) and RFC 7919 (TLS). Note that the module only implements key pair generation, key pair verification, and shared secret computation. No other part of the IKE or TLS protocols is implemented (with the exception of the TLS 1.2 and 1.3 KDFs):

- IKE (RFC 3526):
    - MODP-2048 (ID = 14)
    - MODP-3072 (ID = 15)

  - MODP-4096 (ID = 16)
  - MODP-6144 (ID = 17)
  - MODP-8192 (ID = 18)
- TLS (RFC 7919)
  - ffdhe2048 (ID = 256)
  - ffdhe3072 (ID = 257)
  - ffdhe4096 (ID = 258)
  - ffdhe6144 (ID = 259)
  - ffdhe8192 (ID = 260)

For Elliptic Curve Diffie-Hellman, the module supports the NIST-defined B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, and P-521 curves.

According to FIPS 140-3 IG D.B, the key sizes of DH and ECDH shared secret computation provide 112-200 and 112-256 bits of security strength respectively in approve mode of operation.

The module also supports the AES KW and AES KWP key wrapping mechanisms. These algorithms can be used to wrap SSPs with a security strength of 128, 192, or 256 bits, depending on the wrapping key size.

9.4 SSP entry/output

The module only supports SSP entry and output to and from the calling application running on the same operational environment. This corresponds to manual distribution, electronic entry/output ("CM Software to/from App via TOEPP Path") per FIPS 140-3 IG 9.5.A Table 1. There is no entry or output of cryptographically protected SSPs.

SSPs can be entered into the module via API input parameters, when required by a service. SSPs can also be output from the module via API output parameters, immediately after generation of the SSP (see Section 9.2).

9.5 SSP storage

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

9.6 SSP zeroization

The memory occupied by SSPs is allocated by regular memory allocation operating system calls. The operator application is responsible for calling the appropriate destruction functions provided in the module's API. The destruction functions (listed in Table 9) overwrite the memory occupied by SSPs with zeroes and de-allocate the memory with the regular memory de-allocation operating system call. All data output is inhibited during zeroization.

10 Self-Tests

The module performs the pre-operational self-test and CASTs automatically when the module is loaded into memory. Pre-operational self-test ensure that the module is not corrupted, and the CASTs ensure that the cryptographic algorithms work as expected. While the module is executing the pre-operational test and the CASTs, the module services are not available, and input and output are inhibited. The module is not available for use by the calling application until the pre-operational self-test and the CASTs are completed successfully. After the pre-operational test and the CASTs succeed, the module becomes operational. If any of the pre-operational test or any of the CASTs fail an error message is returned, and the module transitions to the error state.

All the self-tests are listed in Table 10, with the respective condition under which those tests are performed. Note that the pre-operational integrity test is only executed after all cryptographic algorithm self-tests (CASTs) executed successfully.

Table 10 – Self-Tests

Algorithm	Parameters	Condition	Type	Test
HMAC	SHA2-256	Initialization (after CASTs)	Pre-operational Integrity Test	MAC tag verification on fips.so file
SHA-1	N/A	Initialization	Cryptographic Algorithm Self-Test	KAT digest generation
SHA2-512	N/A	Initialization	Cryptographic Algorithm Self-Test	KAT digest generation
SHA3-256	N/A	Initialization	Cryptographic Algorithm Self-Test	KAT digest generation
SHA3-512	N/A	Initialization	Cryptographic Algorithm Self-Test	KAT digest generation
AES GCM	256-bit key	Initialization	Cryptographic Algorithm Self-Test	KAT encryption and decryption
AES ECB	128-bit key	Initialization	Cryptographic Algorithm Self-Test	KAT decryption
HMAC	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	Initialization	Cryptographic Algorithm Self-Test	KAT MAC generation
KBKDF	HMAC SHA2-256 in counter mode	Initialization	Cryptographic Algorithm Self-Test	KAT key derivation
KDA OneStep	SHA2-224	Initialization	Cryptographic Algorithm Self-Test	KAT key derivation
HKDF	SHA2-256	Initialization	Cryptographic Algorithm Self-Test	KAT key derivation
ANS X9.42 KDF	AES-128 KW with SHA-1	Initialization	Cryptographic Algorithm Self-Test	KAT key derivation

Algorithm	Parameters	Condition	Type	Test
ANS X9.63 KDF	SHA2-256	Initialization	Cryptographic Algorithm Self-Test	KAT key derivation
SSH KDF	SHA-1	Initialization	Cryptographic Algorithm Self-Test	KAT key derivation
TLS 1.2 KDF	SHA2-256	Initialization	Cryptographic Algorithm Self-Test	KAT key derivation
TLS 1.3 KDF	SHA2-256	Initialization	Cryptographic Algorithm Self-Test	KAT key derivation
PBKDF2	SHA2-256 with 4096 iterations and 288-bit salt	Initialization	Cryptographic Algorithm Self-Test	KAT password-based key derivation
KAS-FFC-SSC	ffdhe2048	Initialization	Cryptographic Algorithm Self-Test	KAT shared secret computation
	MODP-2048	Initialization	Cryptographic Algorithm Self-Test	KAT shared secret computation
KAS-ECC-SSC	P-256	Initialization	Cryptographic Algorithm Self-Test	KAT shared secret computation
RSA	PKCS#1 v1.5 with SHA2-256 and 2048-bit key	Initialization	Cryptographic Algorithm Self-Test	KAT signature generation and verification
ECDSA	SHA2-256 and P-224, SHA2-256 and B-233	Initialization	Cryptographic Algorithm Self-Test	KAT signature generation and verification
DH	N/A	DH key pair generation	Pair-wise Consistency Test	Section 5.6.2.1.4[2] pair-wise consistency
RSA	PKCS#1 v1.5 with SHA2-256	RSA key pair generation	Pair-wise Consistency Test	Sign/Verify pair-wise consistency
ECDSA	SHA2-256	EC key pair generation	Pair-wise Consistency Test	Sign/Verify pair-wise consistency
		Kernel Bound Module		
DRBG	AES-128, AES-192 and AES-256 with and without PR	Initialization	Cryptographic Algorithm Self-Test	KAT DRBG generation and reseed

[2] As mentioned in SP 800-56Ar3 section 5.6.2.1.4 - Owner Assurance of Pair-wise Consistency, the module recalculate the public key based on the private key and the domain parameters and then check if it matches the existing public key.

Algorithm	Parameters	Condition	Type	Test
	HMAC-SHA1, HMAC-SHA2-256 and SHA2-512 with and without PR			Health test per section 11.3 of 90A
	SHA1, SHA2-256 and SHA2-512 with and without PR			

10.1 Pre-Operational Tests

The module performs pre-operational tests automatically when the module is powered on. The pre-operational self-tests ensure that the module is not corrupted. The module transitions to the operational state only after the pre-operational self-tests are passed successfully.

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

10.1.1 Pre-Operational Software Integrity Test

The integrity of the shared library component of the module is verified by comparing an HMAC SHA2-256 value calculated at run time with the HMAC SHA2-256 value stored in the /etc/ssl/fipsmodule.cnf file that was computed at installation time.

If the software integrity test fails, the module transitions to the error state (Section 10.3). As mentioned previously, the HMAC and SHA2-256 algorithms go through their respective CASTs before the software integrity test is performed.

10.2 Conditional Self-Tests

10.2.1 Cryptographic Algorithm Self-Tests

The module performs self-tests on all approved cryptographic algorithms as part of the approved services supported in the approved mode of operation, using the tests shown in Table 10. Data output through the data output interface is inhibited during the self-tests. If any of these tests fails, the module transitions to the error state (Section 10.3).

10.2.2 Conditional Pair-Wise Consistency Test

Upon generation of a DH, RSA or EC key pair, the module will perform a pair-wise consistency test (PCT) as shown in Table 10, which provides some assurance that the generated key pair is well formed. For DH key pairs, this test consists of the PCT described in Section 5.6.2.1.4 of [SP 800-56Ar3]. For RSA and EC key pairs, this test consists of a signature generation and a signature verification operation. If the test fails, the module transitions to the error state (Section 10.3).

10.3 Error States

If the module fails any of the self-tests, the module enters the error state. In the error state, the module immediately stops functioning and ends the application process. Consequently, the data output interface is inhibited, and the module accepts no more inputs or requests (as the module is no longer running).

Table 11 lists the error states and the status indicator values that explain the error that has occurred.

Table 11 - Error States

Error State	Cause of Error	Status Indicator
Error	Software integrity test failure	Module will not load
	CAST failure	Module will not load
	PCT failure	Module stops functioning

11 Life-cycle Assurance

11.1 Delivery and Operation

11.1.1 Module Installation

The binaries of the module are contained in the base Junos Evolved installation image. The Crypto Officer shall follow this Security Policy to configure the operational environment and install the module to be operated as a FIPS 140-3 validated module.

11.1.2 End of Life Procedures

As the module does not persistently store SSPs, secure sanitization of the module consists of unloading the module. This will zeroize all SSPs in volatile memory.

11.2 Crypto Office Guidance

In order to run in Approved mode, the module must be operated using the approved services, with their corresponding approved cryptographic algorithms provided in this Security Policy (see section 4.3). In addition, key sizes must comply with [SP 800-131Ar2].

11.2.1 Verification of the Module Installation

The operator is responsible to verify the correct installation of module which is already pre-installed on the image file (junos-evo-install-ptx-fixed-x86-64-22.4R2.11-S1-EVO.iso).

The following steps are required:

- run the following command:

  openssl fipsinstall -module /usr/lib64/ossl-modules/fips.so -in /etc/ssl/openssl-fips.cnf -provider_name fips -verify

  Which should output the following:

  VERIFY PASSED

- run the following command to check the name and the version of the OpenSSL:

  openssl list -provider

  Which should output the following:

  ```
  Providers:
  base
   name: OpenSSL Base Provider
    version: 3.0.8
    status: active
  fips
    name: Junos OS Evolved OpenSSL Cryptographic Module
    version: 3.0.8
    status: active
  ```

11.2.2 AES GCM IV

When no IV is explicitly provided, the AES GCM IV generation is performed in compliance with Scenario 2 of IG C.H (Random IV). The AES-GCM IV is generated randomly internal to the module using the approved DRBG provided by the bound kernel. The DRBG seeds itself from the entropy source of the kernel. The GCM IV is 96 bits in length.

11.2.3 AES XTS

The AES algorithm in XTS mode can be only used for the cryptographic protection of data on storage devices, as specified in [SP 800-38E]. The length of a single data unit encrypted with the XTS-AES shall not exceed 2^{20} AES blocks that is 16MB of data. To meet the requirement in [FIPS140-3_IG] IG C.I, the module implements a check to ensure that the two AES keys used in XTS-AES algorithm are not identical.

11.2.4 Key Derivation using SP 800-132 PBKDF2

The module provides password-based key derivation (PBKDF2), compliant with [SP 800-132]. The module supports option 1a from section 5.4 of [SP 800-132], in which the Master Key (MK) or a segment of it is used directly as the Data Protection Key (DPK). In accordance to [SP 800-132] and FIPS IG D.N, the following requirements shall be met:

- Derived keys shall only be used in storage applications. The MK shall not be used for other purposes. The module accepts a minimum length of 12 bits for the MK or DPK.
- Password and passphrases, used as an input for the PBKDF2, shall not be used as cryptographic keys.
- The minimum length of the password or passphrase accepted by the module is 8 characters. This will result in a password strength equal to 10^8. Combined with the minimum iteration count as described below, this provides an acceptable trade-off between user experience and security against brute-force attacks.
- A portion of the salt, with a length of at least 128 bits (this is verified by the module to determine the service is approved), shall be generated randomly using the [SP 800-90Ar1] DRBG provided by the module.
- The iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users. The module only allows minimum iteration count to be 1000.

11.2.5 Compliance to SP 800-56Ar3 assurances

The module offers DH and ECDH shared secret computation services compliant to the [SP 800-56Ar3] and meeting IG D.F scenario 2 path (1). In order to meet the required assurances listed in section 5.6 of [SP 800-56Ar3], the module shall be used together with an application that implements the "TLS protocol" and the following steps shall be performed.

1. The entity using the module, must use the module's "Key pair Generation" service for generating DH/ECDH ephemeral keys. This meets the assurances required by key pair owner defined in the section 5.6.2.1 of [SP 800-56Ar3].
2. As part of the module's shared secret computation(SSC) service, the module internally performs the public key validation on the peer's public key passed in as input to the SSC function. This meets the public key validity assurance required by the sections 5.6.2.2.1/5.6.2.2.2 of [SP 800-56Ar3].

3. The module does not support static keys therefore the "assurance of peer's possession of private key" is not applicable.

12 Mitigation of Other Attacks

Certain cryptographic subroutines and algorithms are vulnerable to timing analysis. The module mitigates this vulnerability by using constant-time implementations. This includes, but is not limited to:

- Big number operations: computing GCDs, modular inversion, multiplication, division, and modular exponentiation (using Montgomery multiplication).
- Elliptic curve point arithmetic: addition and multiplication (using the Montgomery ladder).
- Vector-based AES implementations.

In addition, RSA, ECDSA, ECDH, and DH employ blinding techniques to further impede timing and power analysis. No configuration is needed to enable these countermeasures.

13 Appendix B - Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
API	Application Program Interface
APT	Advanced Package Tool
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CKG	Cryptographic Key Generation
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter Mode
CTS	Ciphertext Stealing
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EVP	Envelope
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
GMAC	Galois Counter Mode Message Authentication Code
HKDF	HMAC-based Key Derivation Function
HMAC	Hash Message Authentication Code
IKE	Internet Key Exchange
IG	Implementation Guidance
KAS	Key Agreement Schema
KAT	Known Answer Test
KBKDF	Key-based Key Derivation Function
KDF	Key Derivation Function

KW	Key Wrap
KWP	Key Wrap with Padding
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
OAEP	Optimal Asymmetric Encryption Padding
OFB	Output Feedback
PAA	Processor Algorithm Acceleration
PCT	Pair-wise Consistency Test
PBKDF2	Password-based Key Derivation Function v2
PKCS	Public-Key Cryptography Standards
PSS	Probabilistic Signature Scheme
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SSC	Shared Secret Computation
SSH	Secure Shell
SSP	Sensitive Security Parameter
TLS	Transport Layer Security
XOF	Extendable Output Function
XTS	XEX-based Tweaked-codebook mode with ciphertext Stealing

14 Appendix C - References

ANSI X9.42-2001	**Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography** 2001 https://webstore.ansi.org/standards/ascx9/ansix9422001
ANSI X9.63-2001	**Public Key Cryptography for the Financial Services Industry, Key Agreement and Key Transport Using Elliptic Curve Cryptography** 2001 https://webstore.ansi.org/standards/ascx9/ansix9632001
FIPS 140-3_IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program** August 12, 2020 https://csrc.nist.gov/CSRC/media/Projects/cryptographic-module-validation-program/documents/fips 140-3/FIPS 140-3 IG.pdf
FIPS 180-4	**Secure Hash Standard (SHS)** March 2012 http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS 186-4	**Digital Signature Standard (DSS)** July 2013 http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
FIPS 197	**Advanced Encryption Standard** November 2001 http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS 198-1	**The Keyed Hash Message Authentication Code (HMAC)** July 2008 http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
FIPS 202	**SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions** August 2015 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
PKCS#1	**Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1** February 2003 http://www.ietf.org/rfc/rfc3447.txt
RFC 3526	**More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)** May 2003 https://www.ietf.org/rfc/rfc3526.txt

RFC 7919	**Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)** August 2016 https://tools.ietf.org/html/rfc7919.txt
RFC 8446	**The Transport Layer Security (TLS) Protocol Version 1.3** August 2018 https://www.ietf.org/rfc/rfc8446.txt
SP 800-38A	**NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
SP 800-38A Addendum	**NIST Special Publication 800-38A Addendum - Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode** October 2010 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a-add.pdf
SP 800-38B	**NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf
SP 800-38C	**NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality** May 2004 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf
SP 800-38D	**NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf
SP 800-38E	**NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices** January 2010 http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf
SP 800-38F	**NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf

SP 800-52	**NIST Special Publication 800-52 Revision 2 - Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations** August 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r2.pdf
SP 800-56Ar3	**NIST Special Publication 800-56A Revision 3 - Recommendation for Pair Wise Key Establishment Schemes Using Discrete Logarithm Cryptography** April 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf
SP 800-56Cr2	**NIST Special Publication 800-56C Revision 2 - Recommendation for Key-Derivation Methods in Key-Establishment Schemes** August 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf
SP 800-57	**NIST Special Publication 800-57 Part 1 Revision 5 - Recommendation for Key Management Part 1: General** May 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf
SP 800-90Ar1	**NIST Special Publication 800-90A - Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP 800-108r1	**NIST Special Publication 800-108r1 - Recommendation for Key Derivation Using Pseudorandom Functions** August 2022 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-108r1.pdf
SP 800-131Ar2	**NIST Special Publication 800-131A – Revision 2 - Transitioning the Use of Cryptographic Algorithms and Key Lengths** March 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf
SP 800-132	**NIST Special Publication 800-132 - Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** December 2010 https://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf
SP 800-133r2	**NIST Special Publication 800-133 – Revision 2 - Recommendation for Cryptographic Key Generation** June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf
SP 800-140B	**CMVP Security Policy Requirements** March 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140B.pdf

SP 800-185	**NIST Special Publication 800-185 - SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash** **December 2016** **https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf**