JUNIPER KERNEL CRYPTO CRYPTOGRAPHIC MODULE VERSION 1.0

FIPS 140-2 NON-PROPRIETARY SECURITY POLICY

VERSION 1.3

Last update: April 22, 2022

Juniper Networks, Inc.

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

1 Cryptographic Module Specification

This document is the non-proprietary FIPS 140-2 Security Policy for version 1.0 of the Juniper Kernel Crypto 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 140-2 (Federal Information Processing Standards Publication 140-2) for a Security Level 1 software module.

The following sections describe the cryptographic module and how it conforms to the FIPS 140-2 specification in each of the required areas.

1.1 Module Overview

The Juniper Kernel Crypto Cryptographic Module (hereafter referred to as "the module") is a software module running as part of the operating system kernel that provides general purpose cryptographic services. It is bound to the Juniper OpenSSL Cryptographic Module validated under FIPS certificate #4131 to check the integrity of its static kernel binary file.

The module provides cryptographic services to kernel applications through a C language Application Program Interface (API) and to applications running in the user space through an AF_ALG socket type interface. The module utilizes processor instructions to optimize and increase the performance of cryptographic algorithms.

For the purpose of the FIPS 140-2 validation, the module is a software-only, multi-chip standalone cryptographic module validated at overall security level 1. The table below shows the security level claimed for each of the eleven sections that comprise the FIPS 140-2 standard.

Table 1 - Security Levels

FIPS 140-2 Section		Security Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services and Authentication	1
4	Finite State Model	1
5	Physical Security	N/A
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self-Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A
Overall Level		1

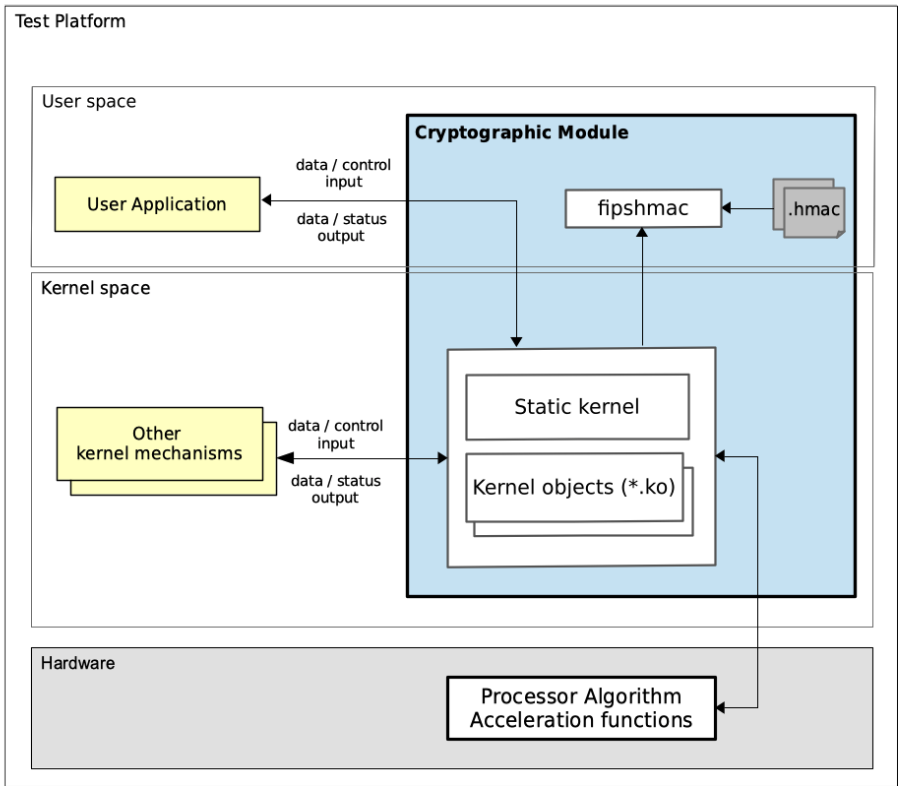
The table below enumerates the components that comprise the module with their location in the target platform.

Component	Description
/usr/bin/fips_chk_hmac	Integrity test utility
/usr/bin/.fips_chk_hmac.hmac	Integrity check HMAC file for the integrity test utility.
/soft/current/bzImage-re-64b.bin	Static kernel binary
/soft/current/.bzImage-re-64b.bin.hmac	Integrity check HMAC file for static kernel binary
/lib/modules/4.8.28-WR2.2.1_standard-g3ee1c25afa39/crypto/*.ko	Cryptographic kernel object files
/lib/modules/4.8.28-WR2.2.1_standard-g3ee1c25afa39/arch/x86/crypto/*.ko	Cryptographic kernel object files

Table 2 - Module Components

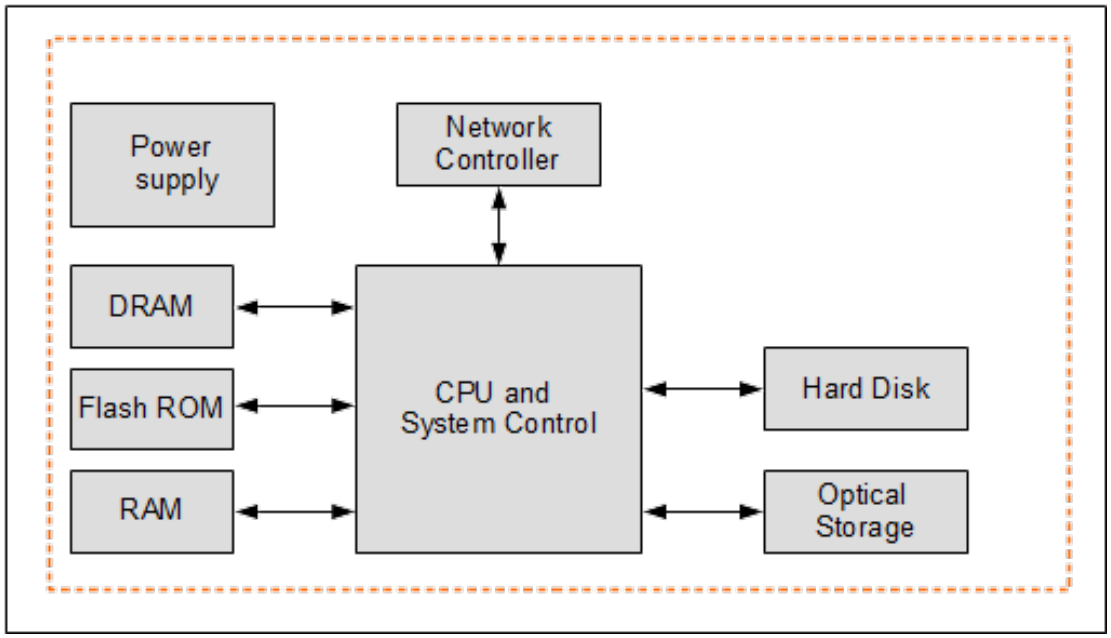
The software block diagram below shows the module, its interfaces with the operational environment and the delimitation of its logical boundary, comprised of all the components within the **BLUE** box.

Figure 1 - Software Block Diagram



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

Figure 2 - Hardware Block Diagram



The module has been tested on the test platforms shown below.

Table 3 - Tested Platforms

Test Platform	Processor	Test Configuration
Juniper Networks® Packet Transport Router Model PTX10003-80C	Intel® Xeon® E5-2628L v4	Junos® OS Evolved version 19.4R2 with and without AES-NI (PAA)

Note: Per FIPS 140-2 IG G.5, the Cryptographic Module Validation Program (CMVP) makes no statement as to the correct operation of the module or the security strengths of the generated keys when this module is ported and executed in an operational environment not listed on the validation certificate.

1.2 Modes of Operation

The module supports two modes of operation:

- **FIPS mode** (the Approved mode of operation): only approved or allowed security functions with sufficient security strength can be used.
- **non-FIPS mode** (the non-Approved mode of operation): only non-approved security functions can be used.

The module enters FIPS mode after power-up tests succeed. Once the module is operational, the mode of operation is implicitly assumed depending on the security function invoked and the security strength of the cryptographic keys.

Critical security parameters used or stored in FIPS mode are not used in non-FIPS mode, and vice versa.

2 Cryptographic Module Ports and Interfaces

As a software-only module, the module does not have physical ports. For the purpose of the FIPS 140-2 validation, the physical ports are interpreted to be the physical ports of the hardware platform on which it runs.

The logical interfaces are the API through which kernel components request services, and the AF_ALG type socket that allows the applications running in the user space to request cryptographic services from the module. The following table summarizes the four logical interfaces.

Table 4 - Ports and Interfaces

FIPS Interface	Physical Port	Logical Interface
Data Input	Keyboard	API input parameters from kernel system calls, AF_ALG type socket.
Data Output	Display	API output parameters from kernel system calls, AF_ALG type socket.
Control Input	Keyboard	API function calls, API input parameters for control from kernel system calls, AF_ALG type socket, kernel command line.
Status Output	Display	API return codes, AF_ALG type socket, kernel logs.
Power Input	PC Power Supply Port	N/A

3 Roles, Services and Authentication

3.1 Roles

The module supports the following roles:

- **User role**: performs cryptographic services (in both FIPS mode and non-FIPS mode), key zeroization, get status, and on-demand self-test.
- **Crypto Officer role**: performs module installation and initialization.

The User and Crypto Officer roles are implicitly assumed by the entity accessing the module services.

3.2 Services

The module provides services to users that assume one of the available roles. All services are shown in Table 5 and Table 6, and described in detail in the user documentation (i.e., man pages).

The table below shows the services available in FIPS mode. For each service, the associated cryptographic algorithms, the roles to perform the service, and the cryptographic keys or Critical Security Parameters (CSPs) and their access rights are listed. The following convention is used to specify access rights to a CSP:

- **Create**: the calling application can create a new CSP.
- **Read**: the calling application can read the CSP.
- **Update**: the calling application can write a new value to the CSP.
- **Zeroize**: the calling application can zeroize the CSP.
- **n/a**: the calling application does not access any CSP or key during its operation.

If the services involve the use of the cryptographic algorithms, the corresponding Cryptographic Algorithm Validation System (CAVS) certificate numbers of the cryptographic algorithms can be found in Table 7 of this security policy. Notice that the algorithms mentioned in the Network Protocol Services correspond to the same implementation of the algorithms described in the Cryptographic Library Services.

Table 5 - Services in FIPS mode of operation

Service	Algorithms	Role	Access	Keys/CSP
Cryptographic Module Services				
Symmetric Encryption and Decryption	AES	User	Read	AES key
	Triple-DES	User	Read	Triple-DES key
Random number generation	DRBG	User	Read, Update	Entropy input string, Internal state
Message digest	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	User	n/a	n/a
Message authentication code (MAC)	HMAC	User	Read	HMAC key
	CMAC with AES	User	Read	AES key

Service	Algorithms	Role	Access	Keys/CSP
	CMAC with Triple-DES	User	Read	Triple-DES key
Encrypt-then-MAC (authenc) operation for IPsec	AES (CBC mode), Triple-DES (CBC mode), HMAC	User	Read	AES key, Triple-DES key, HMAC key
Key Transport Scheme (KTS)	AES-GCM AES-CCM AES-CBC + HMAC Triple-DES-CBC + HMAC	User	Read	AES key, Triple-DES key, HMAC key
Key encapsulation	RSA	User	Read	RSA public and private keys
Other Services				
Error detection code[1]	crc32c , crct10dif1	User	n/a	None
Data compression[1]	deflate, lz4, lz4hc, lzo, zlib, 842	User	n/a	None
Memory copy operation[1]	ecb(cipher_null)	User	n/a	None
Show status	n/a	User	n/a	None
Zeroization	n/a	User	Zeroize	All CSPs
Self-Tests	AES, Triple-DES, SHS, HMAC, RSA, DRBG	User	n/a	None
Module installation	n/a	Crypto Officer	n/a	None
Module initialization	n/a	Crypto Officer	n/a	None

The table below lists the services only available in non-FIPS mode of operation.

Table 6 - Services in non-FIPS mode of operation

Service	Algorithms / Key sizes	Role	Access	Keys
Symmetric encryption and decryption	2-key Triple-DES listed in Table 9	User	Read	2-key Triple-DES key
	AES-XTS with 192-bit keys	User	Read	AES key
	AES-CBC-CTS	User	Read	AES key
	AES-KW	User	Read	AES key

[1] Algorithms in these services do not provide any cryptographic attribute.

	AES-OFB	User	Read	AES key
	Generic GCM encryption with external IV RFC4106 GCM encryption with external IV	User	Read	AES key
Message digest	GHASH outside the GCM context	User	n/a	None
	SHA3			
Message authentication code (MAC)	HMAC with keys less than 112 bits	User	Read	HMAC key
	HMAC-SHA3	User	Read	HMAC key
	CMAC with 2-key Triple-DES	User	Read	2-key Triple-DES key
Signature generation and verification	RSA signature/verification primitive operations listed in Table 9	User	Read	RSA private key
Key encapsulation	RSA with keys smaller than 2048 bits	User	Read	RSA private key
Shared Secret Computation	Diffie-Hellman	User	Read	Diffie-Hellman private key
	EC Diffie-Hellman	User	Read	EC Diffie-Hellman private key

3.3 Algorithms

The Juniper Kernel Crypto Cryptographic Module is compiled to use the support from the processor and assembly code for AES, SHA and GHASH operations to enhance the performance of the module.

The following table shows the CAVS certificates and their associated information of the cryptographic implementation in FIPS mode.

3.3.1 Approved Algorithms

Table 7 - Cryptographic Algorithms

Algorithm	Mode / Method	Key Lengths, Curves or Moduli (in bits)	Use	Standard	CAVP Certs
AES	ECB	128, 192, 256	Data Encryption and Decryption	[FIPS197], [SP800-38A]	C1883 C1890 C1891 A2409 A2410 A2411
	CBC, CTR	128, 192, 256	Data Encryption and Decryption	[FIPS197], [SP800-38A]	C1883 A2409
	CMAC	128, 192, 256	MAC Generation and Verification	[SP800-38B]	C1883 A2409
	CCM	128, 192, 256	Data Encryption and Decryption	[SP800-38C]	C1883 A2409
	XTS	128, 256	Data Encryption and Decryption for Data Storage	[SP800-38E]	C1883 A2409
	GCM	128, 192, 256	Data Encryption and Decryption	[SP800-38D]	C1883 C1891 A2409 A2410
			Data Decryption	[SP800-38D]	C1890 A2411
	GMAC	128, 192, 256	MAC Generation and Verification	[SP800-38D]	C1883 C1891 A2409 A2411
			MAC Verification	[SP800-38D]	C1890
DRBG	Hash_DRBG: SHA-1, SHA-256, SHA-384, SHA-512 with/without PR	n/a	Deterministic Random Bit Generation	[SP800-90A]	C1883
	HMAC_DRBG: SHA-1, SHA-256, SHA-384, SHA-512				C1883

Algorithm	Mode / Method	Key Lengths, Curves or Moduli (in bits)	Use	Standard	CAVP Certs
	with/without PR				
	CTR_DRBG: AES-128, AES-192, AES-256 with DF, with/without PR				C1883
HMAC	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	112 or greater	Message Authentication Code	[FIPS198-1]	C1883 A2409
	SHA-512 (for integrity check)	N/A			A650 (from bound OpenSSL module)
RSA	PKCS#1v1.5 with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	1024, 2048, 3072	Digital Signature Verification	[FIPS186-4]	C1883
SHS	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	n/a	Message Digest	[FIPS180-4]	C1883 A2409
Triple-DES	ECB, CBC, CTR	192 (two-key Triple-DES)	Data Decryption	[SP800-67], [SP800-38A]	C1883 A2409
		192 (three-key Triple-DES)	Data Encryption and Decryption		
	CMAC	192	MAC Generation and Verification	[SP800-67], [SP800-38B]	C1883 A2409
KTS	AES CCM	128, 192, 256	Key Wrapping and Unwrapping	[SP800-38F]	C1883 C1890 C1891 A2409 A2410 A2411
	AES GCM	128, 192, 256			
	AES CBC and HMAC	128, 192. 256			C1883 A2409

Algorithm	Mode / Method	Key Lengths, Curves or Moduli (in bits)	Use	Standard	CAVP Certs
	Triple-DES CBC and HMAC	192			C1883 A2409

3.3.2 *Non-Approved but Allowed Algorithms*

The following table describes the non-Approved but allowed algorithms in FIPS mode:

Table 8 – FIPS-Allowed Cryptographic Algorithms

Algorithm	Caveat	Use
NDRNG	n/a	The module obtains the entropy data from NDRNG to seed the DRBG.
RSA encrypt/decrypt primitives with keys equal or larger than 2048 bits up to 16384 or more.	Provides between 112 and 256 bits of encryption strength.	Key Establishment; allowed per [FIPS140-2_IG] D.9

3.3.3 *Non-Approved Algorithms*

The table below shows the non-Approved cryptographic algorithms implemented in the module that are only available in non-FIPS mode.

Table 9 - Non-Approved Cryptographic Algorithms

Algorithm	Implementation Name	Use
AES in XTS mode with 192-bit keys	“xts”	Data Encryption and Decryption
AES-CBC-CTS	“cts”	Data Encryption and Decryption
AES-KW	“kw”	Key wrapping and unwrapping
AES-OFB	“ofb”	Data Encryption and Decryption
2-key Triple-DES	“des3_ede”, ”cmac(des3_ede)”	Data Encryption
Generic GCM encryption with external IV	“gcm(aes)” with external IV	Data Encryption
RFC4106 GCM encryption with external IV	“rfc4106(gcm(aes))” with external IV	Data Encryption
GHASH	“ghash”	Hashing outside the GCM mode
HMAC with less than 112 bits key	“hmac”	MAC Generation and Verification
HMAC-SHA3	“hmac(sha3-224)” “hmac(sha3-256)” “hmac(sha3-384)”	MAC Generation and Verification

Algorithm	Implementation Name	Use
	"hmac(sha3-512)"	
RSA primitive operations	"rsa"	Digital Signature Generation and Verification
RSA Key Encapsulation	"rsa"	Key Encapsulation with keys smaller than 2,048 bits
Diffie-Hellman	"dh"	Shared secret computation
EC Diffie-Hellman	"ecdh"	Shared secret computation
EC key generation	"ecdh"	Key generation
SHA-3	"sha3-224" "sha3-256" "sha3-384" "sha3-512"	Message digest and MAC generation/verification

3.4 Operator Authentication

The module does not implement user authentication. The role of the user is implicitly assumed based on the service requested.

4 Physical Security

The module is comprised of software only for security level 1; therefore this section is not applicable.

5 Operational Environment

5.1 Applicability

The module operates in a modifiable operational environment per FIPS 140-2 level 1 specifications. The module runs on a commercially available general-purpose operating system executing on the hardware specified in Table 3.

5.2 Policy

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.

6 Cryptographic Key Management

The following table summarizes the Critical Security Parameters (CSPs) that are used by the cryptographic services implemented in the module:

Table 10 - Lifecycle of Critical Security Parameters (CSPs)

Name	Generation	Entry and Output	Zeroization
AES keys	Not Applicable.	The key is passed into the module via API input parameters in plaintext.	crypto_free_cipher() crypto_free_ablkcipher() crypto_free_blkcipher() crypto_free_skcipher() crypto_free_aead()
Triple-DES keys			
HMAC keys			crypto_free_shash() crypto_free_ahash()
Entropy input string	Obtained from the NDRNG.	None	crypto_free_rng()
DRBG internal state (V, C for Hash; V, C, Key for HMAC and CTR)	During DRBG initialization.	None	crypto_free_rng()
RSA Key Transport public and private keys	Not Applicable.	Keys are passed into the module via API input parameters in plaintext.	crypto_free_kpp()

The following sections describe how CSPs, in particular cryptographic keys, are managed during its life cycle.

6.1 Random Number Generation

The module employs a Deterministic Random Bit Generator (DRBG) based on [SP800-90A] for the creation of random numbers. In addition, the module provides a Random Number Generation service to calling applications.

The DRBG supports the Hash_DRBG, HMAC_DRBG and CTR_DRBG mechanisms. The DRBG is initialized during module initialization; the module loads by default the DRBG using the HMAC_DRBG mechanism with SHA-256 without prediction resistance.

The module uses a Non-Deterministic Random Number Generator (NDRNG) as the entropy source for seeding the DRBG. The NDRNG is provided by the operational environment (i.e., Linux RNG), which is within the module's logical boundary. The NDRNG provides at least 128 bits of entropy to the DRBG during initialization (seed) and reseeding (reseed).

The module performs conditional self-tests on the output of NDRNG to ensure that consecutive random numbers do not repeat, and performs DRBG health tests as defined in section 11.3 of [SP800-90A].

CAVEAT: The module generates random strings whose strengths are modified by available entropy.

6.2 Key Generation

The module does not provide any dedicated key generation service for symmetric keys. However, the Random Number Generation service can be called by the user to obtain random numbers which can be used as key material for symmetric algorithms or HMAC.

6.3 Key Agreement / Key Transport / Key Derivation

The module supports the RSA key transport key establishment methodology:

- RSA key transport: key establishment methodology provides between 112 and 256 bits of encryption strength. This is allowed by [FIPS140-2_IG] IG D.9.

The module supports the following Key Transport Schemes (KTS) methodologies using AES-GCM, AES-CCM, AES-CBC + HMAC and Triple-DES-CBC + HMAC:

- AES-GCM: key establishment methodology provides between 128 and 256 bits of encryption strength
- AES-CCM: key establishment methodology provides between 128 and 256 bits of encryption strength
- AES-CBC + HMAC: key establishment methodology provides between 128 and 256 bits of encryption strength
- Triple-DES-CBC + HMAC: key establishment methodology provides 112 bits of encryption strength.

6.4 Key Entry / Output

The module does not support manual key entry or intermediate key generation key output. The keys are provided to the module via API input parameters in plaintext form and output via API output parameters in plaintext form. This is allowed by [FIPS140-2_IG] IG 7.7, according to the “CM Software to/from App Software via GPC INT Path” entry on the Key Establishment Table.

6.5 Key / CSP Storage

Symmetric and asymmetric 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.

The module does not perform persistent storage of keys. The keys and CSPs are stored as plaintext in the RAM. The only exception is the HMAC key used for the Integrity Test, which is stored in the module and relies on the operating system for protection.

6.6 Key / CSP Zeroization

The memory occupied by keys is allocated by regular memory allocation operating system calls. The application is responsible for calling the appropriate zeroization functions provided in the module's API listed in Table 10. The zeroization functions overwrite the memory occupied by keys with “zeros” and deallocate the memory with the regular memory deallocation operating system call.

7 Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

The test platforms listed in Table 3 have been tested and found to conform to the EMI/EMC requirements specified by 47 Code of Federal Regulations, FCC PART 15, Subpart B, Unintentional Radiators, Digital Devices, Class A (i.e., Business use). These devices are designed to provide reasonable protection against harmful interference when the devices are operated in a commercial environment. They shall be installed and used in accordance with the instruction manual.

8 Self-Tests

FIPS 140-2 requires that the module perform power-up tests to ensure the integrity of the module and the correctness of the cryptographic functionality at start up. In addition, the module performs conditional test for NDRNG. If any self-test fails, the kernel panics and the module enters the error state. In the error state, no data output or cryptographic operations are allowed.

See section 9.2.4 for descriptions of possible self-test errors and recovery procedures.

8.1 Power-Up Tests

The module performs power-up tests when the module is loaded into memory, without operator intervention. Power-up tests ensure that the module is not corrupted and that the cryptographic algorithms work as expected.

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

8.1.1 Integrity Tests

The module verifies its integrity through the following mechanisms:

- All kernel object (*.ko) files are signed with a 2048-bit RSA private key and SHA-1. Before these kernel objects are loaded into memory, the module performs RSA signature verification by using the RSA public key from the X.509 certificates that are compiled into the module's binary files. If the signature cannot be verified, the kernel panics to indicate that the test fails and the module enters the error state.
- The integrity of the static kernel binary is ensured with the HMAC-SHA-512 value stored in the corresponding fips_chk_hmac.hmac file that was computed at build time by OpenSSL. At run time, the module invokes the fips_chk_hmac (where the cryptography is provided by the bound OpenSSL module) utility to calculate the HMAC value of the static kernel binary file, and then compares it with the pre-stored one. If the two HMAC values do not match, the kernel panics to indicate that the test fails and the module enters the error state.
- The integrity of the fips_chk_hmac utility is ensured with the HMAC-SHA-512 value stored in the corresponding .hmac file that was computed at build time. At run time, the utility itself calculates the HMAC value of the utility, and then compares it with the pre-stored one. If the two HMAC values do not match, the kernel panics to indicate that the test fails and the module enters the error state.

Both the RSA signature verification and HMAC-SHA-512 algorithms are approved algorithms implemented in the module.

8.1.2 Cryptographic Algorithm Tests

The module performs self-tests on all FIPS-Approved cryptographic algorithms supported in the Approved mode of operation, using the Known Answer Tests (KAT) shown in the following table:

Table 11 - Self-tests

Algorithm	Power-Up Tests
AES	• KAT of AES in ECB mode with 128, 192 and 256 bit keys, encryption/decryption • KAT of AES in CBC mode with 128, 192 and 256 bit keys, encryption/decryption • KAT of AES in CTR mode with 128, 192 and 256 bit keys, encryption/decryption • KAT of AES in GCM mode with 128, 192 and 256 bit keys, encryption/decryption • KAT of AES in CCM mode with 128, 192 and 256 bit keys, encryption/decryption
Triple DES	• KAT of 3-key Triple-DES in ECB mode, encryption/decryption • KAT of 3-key Triple-DES in CBC mode, encryption/decryption • KAT of 3-key Triple-DES in CTR mode, encryption/decryption
CMAC	• KAT of AES in CMAC mode with 128 and 256 bit keys • KAT of 3-key Triple-DES in CMAC mode
SHS	• KAT of SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512
HMAC	• KAT of HMAC-SHA-1 • KAT of HMAC-SHA-224 • KAT of HMAC-SHA-256 • KAT of HMAC-SHA-384 • KAT of HMAC-SHA-512
DRBG	• KAT with No PR, AES-128, AES-192 and AES-256 • KAT with No PR, HMAC-SHA1, HMAC-SHA-256 • KAT with No PR, SHA-256 • KAT with PR, AES-128 • KAT with PR, HMAC-SHA-256 • KAT with PR, SHA-256
RSA	• KAT of RSA signature verification is covered by the integrity tests which is allowed by [FIPS140-2_IG] IG 9.3

For the KAT, the module calculates the result and compares it with the known value. If the answer does not match the known answer, the KAT is failed and the module enters the Error state.

The KATs cover the different cryptographic implementations available in the operating environment.

8.2 On-Demand Self-Tests

On-Demand self-tests can be invoked by powering-off and reloading the module which cause the module to run the power-up tests again. During the execution of the on-demand self-tests, services are not available and no data output or input is possible.

8.3 Conditional Tests

The module performs the Continuous Random Number Generator Test (CRNGT), shown in the following table:

Table 12 - Conditional Tests

Algorithm	Conditional Test
NDRNG	• CRNGT

9 Guidance

9.1 Crypto Officer Guidance

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 140-2 validated module.

9.1.1 Operating Environment Configurations

To configure the operating environment to support FIPS, the following shall be performed with the root privilege:

(1) Enter CLI configuration mode.

(2) Configure FIPS level to 1:
```
set system fips level 1
```

(3) Commit changes:
```
commit
```

(4) Exit configuration mode to enter operational mode:
```
exit
```

(5) Reboot the system with the new settings (answer `yes` to prompt):
```
request system reboot
```

Now, the operating environment is configured to support FIPS operation. The Crypto Officer should check the existence of the file, `/proc/sys/crypto/fips_enabled`, and that it contains "1". If the file does not exist or does not contain "1", the operating environment is not configured to support FIPS and the module will not operate as a FIPS validated module properly.

9.2 User Guidance

In order to run in FIPS mode, the module must be operated using the FIPS Approved services, with their corresponding FIPS Approved and FIPS allowed cryptographic algorithms provided in this Security Policy (see section 3.2). In addition, key sizes must comply with [SP800-131A].

9.2.1 AES-GCM IV

In case the module's power is lost and then restored, the key used for the AES GCM encryption or decryption shall be redistributed.

The module generates the 96-bit IV internally randomly by the module's DRBG, which is compliant with provision 2) of IG A.5.

When a GCM IV is used for decryption, the responsibility for the IV generation lies with the party that performs the AES-GCM encryption therefore there is no restriction on the IV generation.

9.2.2 AES-XTS

The AES algorithm in XTS mode can be only 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 XTS-AES shall not exceed 2^{20} AES blocks that is 16MB of data. To meet the requirement in [FIPS140-2_IG] A.9, the module implements a check to ensure that the two AES keys used in XTS-AES algorithm are not identical.

Note: AES-XTS shall be used with 128 and 256-bit keys only. AES-XTS with 192-bit keys is not an Approved service.

9.2.3 Triple-DES

[SP800-67] imposes a restriction on the number of 64-bit block encryptions performed under the same three-key Triple-DES key.

When the three-key Triple-DES is generated as part of a recognized IETF protocol, the module is limited to 2^{20} 64-bit data block encryptions. This scenario occurs in the following protocols:

- Transport Layer Security (TLS) versions 1.1 and 1.2, conformant with [RFC5246]
- Secure Shell (SSH) protocol, conformant with [RFC4253]
- Internet Key Exchange (IKE) versions 1 and 2, conformant with [RFC7296]

In any other scenario, the module cannot perform more than 2^{16} 64-bit data block encryptions.

The user is responsible for ensuring the module's compliance with this requirement.

9.2.4 Handling FIPS Related Errors

When the module fails any self-test, it will panic the kernel and the operating system will not load. Errors occurred during self-tests transition the module into the error state. The only way to recover from this error state is to reboot the system. If the failure persists, the module must be reinstalled by the Crypto Officer following the instructions as specified in section 9.1.

The kernel dumps self-test success and failure messages into the kernel message ring buffer. The user can use dmesg to read the contents of the kernel ring buffer. The format of the ring buffer (dmesg) output for self-test status is:

> alg: self-tests for %s (%s) passed

Typical messages are similar to "alg: self-tests for xts(aes) (xts(aes-x86_64)) passed" for each algorithm/sub-algorithm type.

10 Mitigation of Other Attacks

The module does not implement mitigation of other attacks.

11 Appendix B - Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
API	Application Program Interface
APT	Advanced Package Tool
CAVP	Cryptographic Algorithm Validation Program
CAVS	Cryptographic Algorithm Validation System
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CLMUL	Carry-less Multiplication
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CPACF	CP Assist for Cryptographic Function
CRNGT	Continuous Random Number Generator Test
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DF	Derivation Function
DSA	Digital Signature Algorithm
DTLS	Datagram Transport Layer Security
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
EMI/EMC	Electromagnetic Interference/Electromagnetic Compatibility
FCC	Federal Communications Commission
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
GPC	General Purpose Computer
HMAC	Hash Message Authentication Code
IG	Implementation Guidance
KAS	Key Agreement Schema
KAT	Known Answer Test

KDF	Key Derivation Function
KW	Key Wrap
LPAR	Logical Partitions
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
NDRNG	Non-Deterministic Random Number Generator
OFB	Output Feedback
PAA	Processor Algorithm Acceleration
PAI	Processor Algorithm Implementation
PCT	Pair-wise Consistency Test
PR	Prediction Resistance
PRNG	Pseudo-Random Number Generator
PSS	Probabilistic Signature Scheme
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SSSE3	Supplemental Streaming SIMD Extensions 3
TLS	Transport Layer Security
XTS	XEX-based Tweaked-codebook mode with ciphertext Stealing

12 Appendix C - References

FIPS140-2 **FIPS PUB 140-2 - Security Requirements For Cryptographic Modules**
May 2001
http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

FIPS140-2_IG **Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program**
August 16, 2019
http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf

FIPS180-4 **Secure Hash Standard (SHS)**
March 2012
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

FIPS186-4 **Digital Signature Standard (DSS)**
July 2013
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

FIPS197 **Advanced Encryption Standard**
November 2001
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

FIPS198-1 **The Keyed Hash Message Authentication Code (HMAC)**
July 2008
http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

PKCS#1 **Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1**
February 2003
http://www.ietf.org/rfc/rfc3447.txt

SP800-38A **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

SP800-38B **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

SP800-38C **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

SP800-38D	**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
SP800-38E	**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
SP800-56A	**NIST Special Publication 800-56A Revision 2 - Recommendation for Pair Wise Key Establishment Schemes Using Discrete Logarithm Cryptography** May 2013 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800 56Ar2.pdf
SP800-56B	**NIST Special Publication 800-56B Revision 1 - Recommendation for Pair-Wise Key-Establishment Schemes Using Integer Factorization Cryptography** September 2014 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Br1.pdf
SP800-57	**NIST Special Publication 800-57 Part 1 Revision 4 - Recommendation for Key Management Part 1: General** January 2016 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf
SP800-67	**NIST Special Publication 800-67 Revision 1 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher** January 2012 http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf
SP800-90A	**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
SP800-131A	**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