Red Hat, Inc.

Red Hat Enterprise Linux 8 NSS Cryptographic Module

FIPS 140-3 Non-Proprietary Security Policy

Prepared by:

atsec information security corporation
4516 Seton Center Pkwy, Suite 250
Austin, TX 78759
www.atsec.com

Document version: 1.2
Last update: 2026-07-30

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for version 3.101.0-36157fa50e4c0485 of the Red Hat Enterprise Linux 8 NSS Cryptographic Module. It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-3 (Federal Information Processing Standards Publication 140-3) for an overall Security Level 1 module.

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 Security Levels

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

Table 1: Security Levels

1.3 Additional Information

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.

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:
The Red Hat Enterprise Linux 8 NSS 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) designed to support cross-platform development of security-enabled client and server applications. Applications built with NSS can support SSLv3, TLS, IKEv2, PKCS#5, PKCS#7, PKCS#11, PKCS#12, S/MIME, X.509 v3 certificates, and other security standards supporting FIPS 140-3 validated cryptographic algorithms. It combines a vertical stack of Linux components intended to limit the external interface each separate component may provide.

Module Type: Software

Module Embodiment: Multi-Chip Standalone

Cryptographic Boundary:
The cryptographic boundary consists only of the libsoftokn3.so and libfreeblpriv3.so libraries along with their associated integrity check values as listed in Section 2.2. If any other NSS API outside of these two libraries is invoked, the user is not interacting with the module specified in this Security Policy.

Tested Operational Environment's Physical Perimeter (TOEPP):
The TOEPP of the module is defined as the general-purpose computer on which the module is installed.

The entropy source and the PAA provided by the processor are located within the module's physical perimeter and outside of the module's cryptographic boundary.

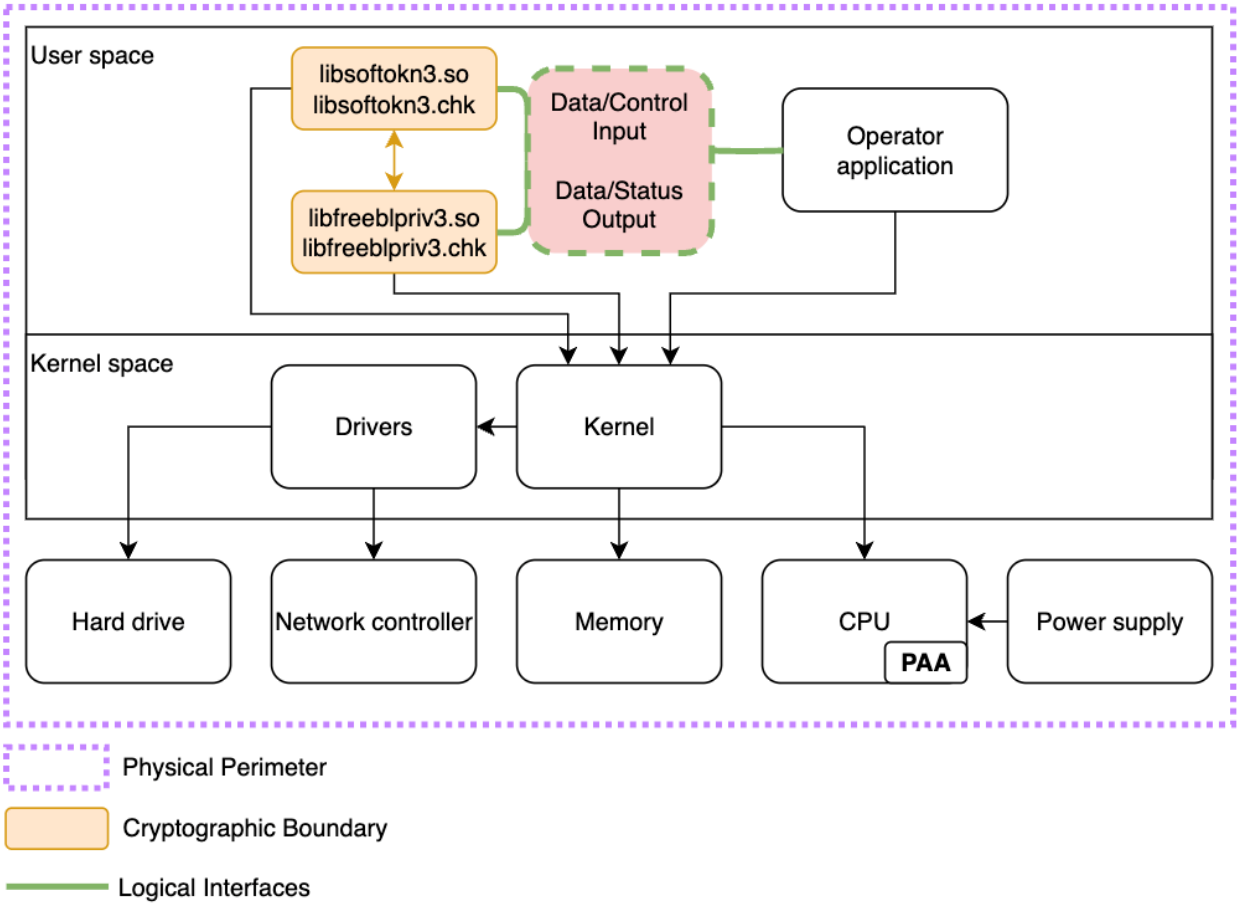

Figure 1: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Hardware:

N/A for this module.

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libsoftokn3.so, libfreeblpriv3.so	3.101.0-36157fa50e4c0485	N/A	HMAC-SHA-256

Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)

Tested Module Identification – Hybrid Disjoint Hardware:

N/A for this module.

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Red Hat Enterprise Linux 8	Dell PowerEdge R440	Intel(R) Xeon(R) Silver 4216	Yes	N/A	3.101.0-36157fa50e4c0485
Red Hat Enterprise Linux 8	Dell PowerEdge R440	Intel(R) Xeon(R) Silver 4216	No	N/A	3.101.0-36157fa50e4c0485

Table 3: Tested Operational Environments - Software, Firmware, Hybrid

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

N/A for this module.

2.3 Excluded Components

There are no components within the cryptographic boundary excluded from the FIPS 140-3 requirements.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved	Automatically entered whenever an approved service is requested.	Approved	Equivalent to the indicator of the requested service ("CKR_OK" for approved DRBG service and NSC_NSSGetFIPSStatus returns CKS_NSS_FIPS_OK (1) for all other approved services).
Non-Approved	Automatically entered whenever a non-approved service is requested.	Non-Approved	Equivalent to the indicator of the requested service (NSC_NSSGetFIPSStatus does not return CKS_NSS_FIPS_OK (1)).

Table 4: Modes List and Description

After passing all pre-operational self-tests and cryptographic algorithm self-tests executed on start-up, the module automatically transitions to the approved mode. No operator intervention is required to reach this point.

Mode Change Instructions and Status:

The module automatically switches between the approved and non-approved modes depending on the services requested by the operator. The status indicator of the mode of operation is equivalent to the indicator of the service that was requested.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A6443, A6445	-	SP 800-38A
AES-CBC-CS1	A6443, A6445	-	SP 800-38A
AES-CMAC	A6443, A6445	-	SP 800-38B
AES-CTR	A6443, A6445	-	SP 800-38A
AES-ECB	A6443, A6445	-	SP 800-38A
AES-GCM	A6443, A6445, A6446	-	SP 800-38D
AES-KW	A6443, A6445	-	SP 800-38F
AES-KWP	A6443, A6445	-	SP 800-38F
ECDSA KeyGen (FIPS186-5)	A6443	-	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A6443	-	FIPS 186-5
ECDSA SigVer (FIPS186-5)	A6443	-	FIPS 186-5
Hash DRBG	A6443	-	SP 800-90A Rev. 1
HMAC-SHA2-224	A6443, A6447	-	FIPS 198-1
HMAC-SHA2-256	A6443, A6447	-	FIPS 198-1
HMAC-SHA2-384	A6443, A6447	-	FIPS 198-1
HMAC-SHA2-512	A6443, A6447	-	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A6443	-	SP 800-56A Rev. 3
KAS-FFC-SSC Sp800-56Ar3	A6443	-	SP 800-56A Rev. 3
KDA HKDF SP800-56Cr2	A6442	-	SP 800-56C Rev. 2
KDF IKEv2 (CVL)	A6444	-	SP 800-135 Rev. 1
KDF SP800-108	A6443	-	SP 800-108 Rev. 1
KTS-IFC	A6443	-	SP 800-56B Rev. 2
PBKDF	A6443	-	SP 800-132
RSA KeyGen (FIPS186-5)	A6443	-	FIPS 186-5
RSA SigGen (FIPS186-5)	A6443	-	FIPS 186-5
RSA SigVer (FIPS186-2)	A6443	-	FIPS 186-4
RSA SigVer (FIPS186-4)	A6443	-	FIPS 186-4
RSA SigVer (FIPS186-5)	A6443	-	FIPS 186-5
Safe Primes Key Generation	A6443	-	SP 800-56A Rev. 3
SHA2-224	A6443, A6447	-	FIPS 180-4
SHA2-256	A6443, A6447	-	FIPS 180-4

Algorithm	CAVP Cert	Properties	Reference
SHA2-384	A6443, A6447	-	FIPS 180-4
SHA2-512	A6443, A6447	-	FIPS 180-4
TLS v1.2 KDF RFC7627 (CVL)	A6443	-	SP 800-135 Rev. 1

Table 5: Approved Algorithms

The table above lists all approved cryptographic algorithms of the module, including specific key lengths employed for approved services in Section 4.3, and implemented modes or methods of operation of the algorithms.

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
Symmetric Cryptographic Key Generation (CKG)	Key type:Symmetric	N/A	SP 800-133r2, section 4, example 1, and section 6.1
Asymmetric Cryptographic Key Generation (CKG)	Key type:Asymmetric	N/A	SP 800-133r2, section 4, example 1

Table 6: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
MD2, MD5, SHA-1	Message digest
RC2, RC4, DES, Triple-DES, CDMF, Camellia, SEED, ChaCha20(-Poly1305)	Encryption, Decryption
AES GCM (external IV)	Encryption
CBC-MAC, AES XCBC-MAC, AES XCBC-MAC-96	Message authentication
HMAC (MD2, MD5, SHA-1; < 112-bit keys)	Message authentication
HMAC/SSLv3 MAC (constant-time implementation)	Message authentication
MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, DES, Triple-DES, AES, Camellia, SEED, ANS X9.63 KDF, SSL 3 PRF, IKEv1 PRF, TLS 1.0/1.1 KDF, TLS KDF without extended master secret	Key derivation
KBKDF, HKDF, TLS 1.2 KDF, IKEv2 PRF (< 112-bit keys)	Key derivation

Name	Use and Function
KBKDF (MD2, MD5)	Key derivation
IKEv2 PRF (MD2, MD5)	Key derivation
PKCS#5 PBE, PKCS#12 PBE	Password-based key derivation
PBKDF2 (short password; short salt; insufficient iterations; < 112-bit keys)	Password-based key derivation
J-PAKE	Shared secret computation
KAS-FFC-SSC (FIPS 186-type groups)	Shared secret computation
X25519	Shared secret computation
DSA	Signature generation, Signature verification, Parameter generation, Parameter verification, Key pair generation
RSA (primitive; PKCS#1 v1.5 or PSS with MD2, MD5, SHA-1)	Signature generation, Signature verification
RSA (< 2048-bit keys)	Signature generation
RSA (< 1024-bit keys)	Signature verification
ECDSA (component)	Signature generation, Signature verification
RSA (data encryption / decryption)	Asymmetric encryption, Asymmetric decryption with primitive or SHA-1, SHA-224
RSA (< 2048 bits; > 4096 bits)	Key pair generation
Ed25519, X25519	Key pair generation
Diffie Hellman (FIPS 186-type groups)	Key pair generation
Symmetric key generation (< 112 bits)	Secret key generation

Table 7: Non-Approved, Not Allowed Algorithms

The table above lists all the non-approved cryptographic algorithms of the module employed by the non-approved services in Section 4.4.

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Encryption with AES	BC-UnAuth	Encryption using AES		AES-CBC: (A6443, A6445) AES-CBC-CS1: (A6443, A6445)

Name	Type	Description	Properties	Algorithms
				AES-CTR: (A6443, A6445) AES-ECB: (A6443, A6445)
Decryption with AES	BC-UnAuth	Decryption using AES		AES-CBC: (A6443, A6445) AES-CBC-CS1: (A6443, A6445) AES-CTR: (A6443, A6445) AES-ECB: (A6443, A6445)
Authenticated Encryption with AES	BC-Auth	Authenticated encryption using AES		AES-KW: (A6443, A6445) AES-KWP: (A6443, A6445) AES-GCM: (A6443, A6445, A6446)
Authenticated Decryption with AES	BC-Auth	Authenticated decryption using AES		AES-KW: (A6443, A6445) AES-KWP: (A6443, A6445) AES-GCM: (A6443, A6445, A6446)
Key Derivation with PBKDF2	PBKDF	Key derivation using PBKDF2		PBKDF: (A6443)
Key Derivation with KBKDF	KBKDF	Key derivation using KBKDF		KDF SP800-108: (A6443)
Key Derivation with HKDF	KAS-56CKDF	Key derivation using HKDF		KDA HKDF SP800-56Cr2: (A6442)
Key Derivation with TLS 1.2 KDF	KAS-135KDF	Key derivation using TLS 1.2 KDF		TLS v1.2 KDF RFC7627: (A6443)
Key Derivation with IKEv2 KDF	KAS-135KDF	Key derivation using IKEv2 KDF		KDF IKEv2: (A6444)
Key Wrapping with AES (KTS)	KTS-Wrap	Key wrapping using AES	Standard:SP 800-38D and SP 800-38F IG D.G:approved key wrapping method Key confirmation:no	AES-KW: (A6443, A6445) AES-KWP: (A6443, A6445)

Name	Type	Description	Properties	Algorithms
			Caveat:Key establishment methodology provides between 128 and 256 bits of security strength	
Key Unwrapping with AES (KTS)	KTS-Unwrap	Key unwrapping using AES	Standard:SP 800-38D and SP 800-38F IG D.G:approved key unwrapping method Key confirmation:no Caveat:Key establishment methodology provides between 128 and 256 bits of security strength	AES-KW: (A6443, A6445) AES-KWP: (A6443, A6445) AES-GCM: (A6443, A6445, A6446)
Message Authentication with HMAC	MAC	Message authentication using HMAC		HMAC-SHA2-224: (A6443, A6447) HMAC-SHA2-256: (A6443, A6447) HMAC-SHA2-384: (A6443, A6447) HMAC-SHA2-512: (A6443, A6447)
Message Authentication with CMAC	MAC	Message authentication using CMAC		AES-CMAC: (A6443, A6445)
Random Number Generation with Hash_DRBG	DRBG	Random number generation using Hash_DRBG		Hash DRBG: (A6443)
Shared Secret Computation with KAS-ECC-SSC	KAS-SSC	Shared secret computation using KAS-ECC-SSC		KAS-ECC-SSC Sp800-56Ar3: (A6443)
Shared Secret Computation with KAS-FFC-SSC	KAS-SSC	Shared secret computation using KAS-FFC-SSC		KAS-FFC-SSC Sp800-56Ar3: (A6443)
Signature Generation with RSA	DigSig-SigGen	Signature generation using RSA		RSA SigGen (FIPS186-5): (A6443)

Name	Type	Description	Properties	Algorithms
Signature Generation with ECDSA	DigSig-SigGen	Signature generation using ECDSA		ECDSA SigGen (FIPS186-5): (A6443)
Signature Verification with RSA (legacy use)	DigSig-SigVer	Signature verification using RSA using legacy moduli		RSA SigVer (FIPS186-2): (A6443) RSA SigVer (FIPS186-4): (A6443)
Signature Verification with RSA	DigSig-SigVer	Signature verification using RSA		RSA SigVer (FIPS186-5): (A6443)
Signature Verification with ECDSA	DigSig-SigVer	Signature verification using ECDSA		ECDSA SigVer (FIPS186-5): (A6443)
Symmetric Key Generation with Hash_DRBG	CKG	Direct symmetric key generation using Hash_DRBG		Hash DRBG: (A6443) Symmetric Cryptographic Key Generation (CKG): () Key type: Symmetric
Key Pair Generation with RSA	AsymKeyPair-KeyGen CKG	Key pair generation using RSA		RSA KeyGen (FIPS186-5): (A6443) Asymmetric Cryptographic Key Generation (CKG): () Key type: Asymmetric
Key Pair Generation with ECDSA	AsymKeyPair-KeyGen CKG	Key pair generation using ECDSA		ECDSA KeyGen (FIPS186-5): (A6443) Asymmetric Cryptographic Key Generation (CKG): () Key type: Asymmetric

Name	Type	Description	Properties	Algorithms
Key Pair Generation with Safe Primes	AsymKeyPair-KeyGen CKG	Key pair generation using Safe Primes		Safe Primes Key Generation: (A6443) Asymmetric Cryptographic Key Generation (CKG): () Key type: Asymmetric
Message Digest with SHA	SHA	Message digest using SHA		SHA2-224: (A6443, A6447) SHA2-256: (A6443, A6447) SHA2-384: (A6443, A6447) SHA2-512: (A6443, A6447)
Authenticated Encryption with RSA	KTS-Encap	Authenticated asymmetric encryption using RSA		KTS-IFC: (A6443)
Authenticated Decryption with RSA	KTS-Decap	Authenticated asymmetric decryption using RSA		KTS-IFC: (A6443)

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 AES-GCM IV

The Crypto Officer shall consider the following requirements and restrictions when using the module.

For TLS 1.2, the module offers the AES-GCM implementation and uses the context of Scenario 1 of FIPS 140-3 IG C.H. NSS is compliant with SP 800-52r2 Section 3.3.1 and the mechanism for IV generation is compliant with RFC 5288.

The module does not implement the TLS protocol. The module's implementation of AES-GCM is used together with an application that runs outside the module's cryptographic boundary. The design of the TLS protocol implicitly ensures that the counter (the nonce_explicit part of the IV) does not exhaust the maximum number of possible values for a given session key.

In the event the module's power is lost and restored, the consuming application must ensure that a new key for use with the AES-GCM key encryption or decryption under this scenario shall be established.

Alternatively, the Crypto Officer can use the module's API to perform AES-GCM encryption using internal IV generation that complies with Scenario 2 of the IG C.H. These IVs are always at least 96 bits and generated using the approved DRBG internal to the module's boundary.

Additionally, the module offers an internal deterministic IV generation mode compliant with Scenario 3 of FIPS 140-3 IG C.H. The size of the fixed (name) field used by this IV generation mode is at least 32 bits. The module then internally generates a 32 bit or longer deterministic non-repetitive counter. The module explicitly ensures that this counter is monotonically increasing at each invocation of the AES-GCM for the same encryption key, and that this counter does not exhaust all its possible values. The generated GCM IV is at least 96 bits in length.

Finally, for TLS 1.3, the AES-GCM implementation uses the context of Scenario 5 of FIPS 140-3 IG C.H. The protocol that provides this compliance is TLS 1.3, defined in RFC 8446 of August 2018, using the cipher-suites that explicitly select AES-GCM as the encryption/decryption cipher (Appendix B.4 of RFC 8446). The module supports acceptable AES-GCM cipher suites from Section 3.3.1 of SP800-52r2. TLS 1.3 employs separate 64-bit sequence numbers, one for protocol records that are received, and one for protocol records that are sent to a peer. These sequence numbers are set at zero at the beginning of a TLS 1.3 connection and each time when the AES-GCM key is changed. After reading or writing a record, the respective sequence number is incremented by one. The protocol specification determines that the sequence number should not wrap, and if this condition is observed, then the protocol implementation must either trigger a re-key of the session (i.e., a new key for AES-GCM), or terminate the connection.

In case the module's power is lost and then restored, a new key for use with the AES-GCM encryption/decryption shall be established.

2.7.2 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 140-3 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 enforces the length of the MK or DPK to be of 112 bits or more for the service to be approved.
- Passwords or 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. The probability of guessing the value is estimated to be at most $1/62^8 = 4 \times 10^{-15}$, when the password is a combination of lowercase, uppercase, and numeric characters. If the password solely consists of digits, the probability of guessing the value is estimated to be 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 shall be generated randomly using the SP 800-90Ar1 DRBG provided by the module. The module restricts minimum length to 128 bits.
- 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.

2.7.3 SP 800-56Ar3 Assurances

The module provides shared secret computation (KAS-FFC-SSC and KAS-ECC-SSC) compliant with SP800-56Ar3, in accordance with scenario 2 (1) of FIPS 140-3 IG D.F.

To comply with the assurances found in Section 5.6.2 of SP 800-56Ar3, the operator must use the module together with an application that implements the TLS protocol. Additionally, the module's approved Key Pair Generation service (see Section 4.3) must be used to generate ephemeral Diffie-Hellman or EC Diffie-Hellman key pairs, or the key pairs must be obtained from another FIPS-validated module. As part of this service, the module will internally perform the full public key validation of the generated public key.

The module's shared secret computation service will internally perform the full public key validation of the peer public key, complying with Section 5.6.2.2.2 of SP 800-56Ar3.

2.7.4 RSA Approved Modulus Size

As allowed by FIPS 140-3 IG C.F, the module implements approved RSA signature verification with 1024, 1280, 1536 and 1792-bit moduli. The 1024-bit modulus has been CAVP tested for RSA signature verification in compliance with FIPS 186-4, while the 1536-bit modulus has been CAVP tested for RSA signature verification in compliance with FIPS 186-2. The 1280 and 1792-bit modulus are approved for FIPS 186-2 signature verification, but are untested as no CAVP testing is available for these moduli.

For all other approved moduli (namely 2048-, 3072-, and 4096-bit keys) supported by the module, RSA key pair generation, signature generation, signature verification, key encapsulation and key un-encapsulation are approved and CAVP tested in compliance with FIPS 186-5.

2.7.5 RSA Asymmetric Encryption and Decryption

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

To comply with SP 800-56B Rev. 2 assurances found in its Section 6 (specifically SP 800-56B Rev. 2 Section 6.4 Required Assurances) the entity using the module must obtain required assurances listed in section 6.4 of SP 800-56B Rev. 2 by performing the following steps:

1. The entity requesting the RSA decryption service from the module, shall only use an RSA private key that was generated by an active FIPS validated module that implements FIPS 186-5 compliant RSA key generation service and performs the key pair validity and the pairwise consistency as stated in section 6.4.1.1 of the SP 800-56B Rev. 2. Additionally, the entity shall renew these assurances over time by using any method described in section 6.4.1.5 of the SP 800-56B Rev. 2.
2. For use of an RSA encryption service in the context of key transport per IG D.G the entity using the module shall:
    a. verify the validity of the peer's public key using the public key validation service of the module;
    b. confirm the peer's possession of private key by using any method specified in section 6.4.2.3 of the SP 800-56B Rev. 2.

    Only after the above assurances are successfully met, shall the entity use the peer's public key to perform the RSA encryption service of the module.

2.7.6 Legacy Use

Digital signature using RSA with 1024, 1280, 1536, 1792-bit moduli is allowed for legacy use only.

These legacy algorithms can only be used on data that was generated prior to the Legacy Date specified in FIPS 140-3 IG C.M.

2.7.7 Key Agreement

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

2.8 RBG and Entropy

Cert Number	Vendor Name
E175	Red Hat, Inc.

Table 9: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
RHEL 8 Userspace CPU Time Jitter RNG Entropy Source	Non-Physical	RHEL 8 on Intel Cascade Lake Xeon Silver 4216	256 bits	227 bits	HMAC-SHA2-512

Table 10: Entropy Sources

The module employs a Deterministic Random Bit Generator (DRBG) implementation based on SP 800-90Ar1. This DRBG is used internally by the module (e.g. to generate symmetric keys, seeds for asymmetric key pairs, and random numbers for security functions). It can also be accessed using the specified API functions.

The DRBG implemented is a SHA-256 Hash_DRBG, seeded by the entropy source described in the table above. It does not employ prediction resistance.

The DRBG is instantiated with an 880-bits long entropy input (corresponding to 781 bits of entropy). Additionally, the DRBG is reseeded with a 440-bit long entropy input (corresponding to 390 bits of entropy). Outputs of multiple GetEntropy() calls are concatenated to receive the entropy input length greater than 256 bits. The output is truncated to get the entropy input string which is not a multiple of 256.

The module complies with the Public Use Document for ESV certificate E175 by reading entropy data from the get_random() function with the GRND_RANDOM flag set, which corresponds to the GetEntropy() conceptual interface. The operational environment on the ESV certificate is identical to the operating system described in this document, Which implements the entropy source (outside the module's cryptographic boundary). Thus, the module is compliant with scenario 1(b) of IG 9.3.A. There are no maintenance requirements for the entropy source.

2.9 Key Generation

The module implements Cryptographic Key Generation (CKG, vendor affirmed), compliant with SP 800-133r2 as specified in Section 2.6.

When random values are required, they are obtained from the SP 800-90Ar1 approved DRBG, compliant with Section 4 of SP 800-133r2.

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

2.10 Key Establishment

The module implements shared secret computation and key transport methods as listed in the Security Function Implementation table in Section 2.6.

2.11 Industry Protocols

For KAS-FFC-SSC, the module supports the use of the safe primes defined in RFC 3526 (IKE) and RFC 7919 (TLS) as listed in Section 2.10. Note that the module only implements key pair generation and verification, and shared secret computation. No other part of the IKE or TLS protocols is implemented (with the exception of the TLS 1.2 KDF (RFC 7627) and IKEv2 KDF).

TLS 1.2 KDF (RFC 7627) and IKEv2 implementations shall only be used to generate secret keys in the context of the TLS 1.2 and IKE protocols respectively.

AES-GCM with internal IV generation is offered in the approved mode compliant with TLS 1.2 (RFC 5288) and TLS 1.3 (RFC 8446). This functionality shall only be used in conjunction with the TLS protocol.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

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

Table 11: Ports and Interfaces

The logical interfaces are the APIs through which the applications request services. These logical interfaces are logically separated from each other by the API design. The module does not implement a control output interface.

4 Roles, Services, and Authentication

4.1 Authentication Methods

The module does not support authentication for roles.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 12: Roles

No support is provided for multiple concurrent operators.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Encryption	Encrypt a plaintext	CKS_NSS_FIPS_OK (1)	AES Key, IV, plaintext	Ciphertext	Encryption with AES	Crypto Officer - AES Key: W,E
Decryption	Decrypt a ciphertext	CKS_NSS_FIPS_OK (1)	AES Key, IV, ciphertext	Plaintext	Decryption with AES	Crypto Officer - AES Key: W,E
Authenticated Symmetric Encryption	Encrypt a plaintext	CKS_NSS_FIPS_OK (1)	AES Key, IV, plaintext	Ciphertext, MAC tag	Authenticated Encryption with AES	Crypto Officer - AES Key: W,E
Authenticated Symmetric Decryption	Decrypt a ciphertext	CKS_NSS_FIPS_OK (1)	AES key, IV, MAC tag, ciphertext	Plaintext or fail	Authenticated Decryption with AES	Crypto Officer - AES Key: W,E
Authenticated Asymmetric Encryption	Encrypt a plaintext using RSA	CKS_NSS_FIPS_OK (1)	RSA Public key, plaintext	Ciphertext	Authenticated Encryption with RSA	Crypto Officer - RSA Public Key: W,E
Authenticated Asymmetric Decryption	Decrypt a ciphertext with RSA	CKS_NSS_FIPS_OK (1)	RSA Private Key, ciphertext	Plaintext or fail	Authenticated Decryption with RSA	Crypto Officer - RSA Private Key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Key Derivation from a KDK	Derive a key from a key-derivation key	CKS_NSS_FIPS_OK (1)	Key-Derivation Key, output length	KBKDF Derived Key	Key Derivation with KBKDF	Crypto Officer - Key-Derivation Key: W,E - KBKDF Derived Key: G,R
Key Derivation from a shared secret	Derive a key from a shared secret	CKS_NSS_FIPS_OK (1)	Shared Secret, output length	HKDF Derived Key/TLS Derived Key/IKE Derived Key	Key Derivation with HKDF Key Derivation with TLS 1.2 KDF Key Derivation with IKEv2 KDF	Crypto Officer - Shared Secret: W,E - HKDF Derived Key: G,R - TLS Derived Key: G,R - IKE Derived Key: G,R
Password-Based Key Derivation	Derive a key from a password	CKS_NSS_FIPS_OK (1)	Password, salt, iteration count, output length	PBKDF2 Derived Key	Key Derivation with PBKDF2	Crypto Officer - Password: W,E - PBKDF2 Derived Key: G,R
Key Wrapping	Wrap a CSP	CKS_NSS_FIPS_OK (1)	AES Key-Encryption Key, any CSP (other than Password)	Wrapped CSP	Key Wrapping with AES (KTS)	Crypto Officer - AES Key-Encryption Key: W,E - Wrapped CSP: E,R
Key Unwrapping	Unwrap a CSP	CKS_NSS_FIPS_OK (1)	AES Key-Decryption Key, Wrapped CSP	Any CSP (other than Password)	Key Unwrapping with AES (KTS)	Crypto Officer - AES Key-Decryption Key: W,E - Wrapped CSP: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Message Authentication with HMAC	Compute a MAC tag	CKS_NSS_FIPS_OK (1)	HMAC Key	MAC tag	Message Authentication with HMAC	Crypto Officer - HMAC Key: W,E
Message Authentication with CMAC	Compute a MAC tag	CKS_NSS_FIPS_OK (1)	AES Key	MAC tag	Message Authentication with CMAC	Crypto Officer - AES Key: W,E
Message Digest	Compute a message digest	CKS_NSS_FIPS_OK (1)	Message	Digest value	Message Digest with SHA	Crypto Officer
Random Number Generation	Generate random bytes	CKR_OK	Output length	Random bytes	Random Number Generation with Hash_DRBG	Crypto Officer - Entropy Input: W,E,Z - DRBG Seed: G,E,Z - DRBG Internal State (V, C): G,W,E
KAS-FFC-SSC Shared Secret Computation	Compute a shared secret	CKS_NSS_FIPS_OK (1)	FFC Private Key (owner), FFC Public Key (peer)	Shared Secret	Shared Secret Computation with KAS-FFC-SSC	Crypto Officer - FFC Private Key: W,E - FFC Public Key: W,E - Shared Secret: G,R
EC Shared Secret Computation	Compute a shared secret	CKS_NSS_FIPS_OK (1)	EC Private Key (owner), EC Public Key (peer)	Shared Secret	Shared Secret Computation with KAS-ECC-SSC	Crypto Officer - EC Private Key: W,E - EC Public Key: W,E - Shared Secret: G,R

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Signature Generation with RSA	Generate a signature	CKS_NSS_FIPS_OK (1)	RSA Private Key, message	Signature	Signature Generation with RSA	Crypto Officer - RSA Private Key: W,E
Signature Generation with ECDSA	Generate a signature	CKS_NSS_FIPS_OK (1)	EC Private Key, message	Signature	Signature Generation with ECDSA	Crypto Officer - EC Private Key: W,E
Signature Verification with RSA	Verify a signature	CKS_NSS_FIPS_OK (1)	RSA Public Key, message, signature	Pass/fail	Signature Verification with RSA Signature Verification with RSA (legacy use)	Crypto Officer - RSA Public Key: W,E
Signature Verification with ECDSA	Verify a signature	CKS_NSS_FIPS_OK (1)	EC Public Key, message, signature	Pass/fail	Signature Verification with ECDSA	Crypto Officer - EC Public Key: W,E
Key Pair Generation with Safe Primes	Generate a key pair	CKS_NSS_FIPS_OK (1)	Group	Module-generated FFC Public Key, Module-generated FFC Private Key	Key Pair Generation with Safe Primes	Crypto Officer - Module-generated FFC Private Key: G,R - Module-generated FFC Public Key: G,R - Intermediate key generation value: G,E,Z
Key Pair Generation with RSA	Generate a key pair	CKS_NSS_FIPS_OK (1)	Modulus size	Module-generated RSA Public Key, Module-	Key Pair Generation with RSA	Crypto Officer - Module-generated RSA Private Key: G,R

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
				generated RSA Private Key		- Module-generated RSA Public Key: G,R - Intermediate key generation value: G,E,Z
Key Pair Generation with ECDSA	Generate a key pair	CKS_NSS_FIPS_OK (1)	Curve	Module-generated EC Public Key, Module-generated EC Private Key	Key Pair Generation with ECDSA	Crypto Officer - Module-generated EC Private Key: G,R - Module-generated EC Public Key: G,R - Intermediate key generation value: G,E,Z
Symmetric Key Generation	Generate a secret key	CKS_NSS_FIPS_OK (1)	Key size	Symmetric Key	Symmetric Key Generation with Hash_DRBG	Crypto Officer - Symmetric Key: G,R
Show Version	Return the module name and version information	None	N/A	Module name and version information	None	Crypto Officer
Show Status	Return the module status	None	N/A	Module status	None	Crypto Officer
Self-Test	Perform the CASTs and	None	N/A	Pass/fail	Message Digest with SHA	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	integrity tests				Message Authentication with HMAC Encryption with AES Decryption with AES Authenticated Encryption with AES Authenticated Decryption with AES Message Authentication with CMAC Key Derivation with KBKDF Key Derivation with HKDF Key Derivation with TLS 1.2 KDF Key Derivation with IKEv2 KDF Key Derivation with PBKDF2 Shared Secret Computation with KAS-FFC-SSC Shared Secret Computation with KAS-ECC-SSC Signature Generation with RSA Signature	

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					Verification with RSA Signature Generation with ECDSA Signature Verification with ECDSA	
Zeroization	Zeroize all SSPs	N/A	Any SSP	None	None	Crypto Officer - AES Key: Z - AES Key-Encryption Key: Z - AES Key-Decryption Key: Z - HMAC Key: Z - Key-Derivation Key: Z - Shared Secret: Z - Password: Z - KBKDF Derived Key: Z - PBKDF2 Derived Key: Z - HKDF Derived Key: Z - TLS Derived Key: Z - IKE Derived Key: Z - Entropy Input: Z - DRBG

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Seed: Z - DRBG Internal State (V, C): Z - FFC Private Key: Z - FFC Public Key: Z - EC Private Key: Z - EC Public Key: Z - RSA Private Key: Z - RSA Public Key: Z - Intermediate key generation value: Z

Table 13: Approved Services

The module provides services to operators that assume the available role. All services are described in detail in the API documentation (manual pages). For the above table, the convention below applies when specifying the access permissions (types) that the service has for each SSP.

- **Generate (G):** The module generates or derives the SSP.
- **Read (R):** The SSP is read from the module (e.g. the SSP is output).
- **Write (W):** The SSP is updated, imported, or written to the module.
- **Execute (E):** The module uses the SSP in performing a cryptographic operation.
- **Zeroize (Z):** The module zeroizes the SSP.

To interact with the module, a calling application must use the FIPS token APIs provided by libsoftokn3.so. The FIPS token API layer can be used to retrieve the approved service indicator for the module. This indicator consists of three independent service indicators:

1. The session indicator, which must be used for all cryptographic services except the key (pair) generation and key derivation services. It can be accessed by invoking the NSC_NSSGetFIPSStatus function with the CKT_NSS_SESSION_LAST_CHECK parameter. If the output parameter is set to CKS_NSS_FIPS_OK (1), the service was approved.

2. The object indicator, which must be used for the key (pair) generation and key derivation services. It can be accessed by invoking the NSC_NSSGetFIPSStatus function with the CKT_NSS_OBJECT_CHECK parameter and the output derived key. If the output parameter is set to CKS_NSS_FIPS_OK (1), the service was approved.
3. The DRBG service indicator, which must be used for the DRBG service. It can be accessed by invoking the C_SeedRandom or C_GenerateRandom functions. If any of these functions returns CKR_OK, the service was approved.

The above-described behavior maps to example scenario 3 of FIPS 140-3 IG 2.4.C.

4.4 Non-Approved Services

Name	Description	Algorithms	Role
Message Digest	Compute a message digest	MD2, MD5, SHA-1	CO
Encryption	Encrypt a plaintext	RC2, RC4, DES, Triple-DES, CDMF, Camellia, SEED, ChaCha20(-Poly1305) AES GCM (external IV)	CO
Decryption	Decrypt a ciphertext	RC2, RC4, DES, Triple-DES, CDMF, Camellia, SEED, ChaCha20(-Poly1305)	CO
Message Authentication	Compute a MAC tag	CBC-MAC, AES XCBC-MAC, AES XCBC-MAC-96 HMAC (MD2, MD5, SHA-1; < 112-bit keys) HMAC/SSLv3 MAC (constant-time implementation)	CO
Key Derivation	Derive a key from a key-derivation key or a shared secret	MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, DES, Triple-DES, AES, Camellia, SEED, ANS X9.63 KDF, SSL 3 PRF, IKEv1 PRF, TLS 1.0/1.1 KDF, TLS KDF without extended master secret KBKDF, HKDF, TLS 1.2 KDF, IKEv2 PRF (< 112-bit keys) KBKDF (MD2, MD5) IKEv2 PRF (MD2, MD5)	CO
Password-Based Key Derivation	Derive a key from a password	PKCS#5 PBE, PKCS#12 PBE PBKDF2 (short password; short salt; insufficient iterations; < 112-bit keys)	CO
Shared Secret Computation	Compute a shared secret	J-PAKE KAS-FFC-SSC (FIPS 186-type groups) X25519	CO
Signature Generation	Generate a signature	DSA RSA (primitive; PKCS#1 v1.5 or PSS with MD2, MD5, SHA-1) RSA (< 2048-bit keys) ECDSA (component)	CO

Name	Description	Algorithms	Role
Signature Verification	Verify a signature	DSA RSA (< 1024-bit keys) ECDSA (component)	CO
Asymmetric Encryption	Encrypt a plaintext	RSA (data encryption / decryption)	CO
Asymmetric Decryption	Decrypt a plaintext	RSA (data encryption / decryption)	CO
Parameter Generation	Generate domain parameters	DSA	CO
Parameter Verification	Verify domain parameters	DSA	CO
Key Pair Generation	Generate a key pair	DSA Diffie Hellman (FIPS 186-type groups) RSA (< 2048 bits; > 4096 bits) Ed25519, X25519	CO
Secret Key Generation	Generate a secret key	Symmetric key generation (< 112 bits)	CO

Table 14: Non-Approved Services

The table above lists the non-approved services in this module, the algorithms involved, the roles that can request the service, and the respective service indicator. In this table, CO specifies the Crypto Officer role.

4.5 External Software/Firmware Loaded

The module does not load external software or firmware.

5 Software/Firmware Security

5.1 Integrity Techniques

Each software component of the module has an associated HMAC-SHA2-256 integrity check value. The module makes use of an HMAC key hard-coded within the module to perform the integrity test. The integrity of the module is verified by comparing the HMAC-SHA2-256 values calculated at run time with the integrity values embedded in the check (.chk) files that were computed at build time. If the integrity test fails, the module enters the Power-On Error state.

5.2 Initiate on Demand

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

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:
Any SSPs contained within the module are protected by the process isolation and memory separation mechanisms, and only the module has control over these SSPs.

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.

6.2 Configuration Settings and Restrictions

The module shall be installed as stated in Section 11.1.

Instrumentation tools like the ptrace system call, gdb and strace, 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-validated operational environment.

6.3 Additional Information

The Red Hat Enterprise Linux operating system is used as the basis of other products which include but are not limited to:

- Red Hat Enterprise Linux CoreOS
- Red Hat Ansible Automation Platform
- Red Hat OpenStack Platform
- Red Hat OpenShift
- Red Hat Gluster Storage
- Red Hat Satellite

Compliance is maintained for these products whenever the binary is found unchanged.

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 Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
RAM	Temporary storage for SSPs used by the module as part of service execution. The module does not perform persistent storage of SSPs	Dynamic

Table 15: Storage Areas

SSPs imported, generated, derived, or otherwise established by the module are stored in RAM while the module is operational. The operator application can use these SSPs to perform cryptographic operations, or export them as described in Section 9.2.

The module maintains internal separation of the SSPs (including CSPs) in approved and non-approved modes of operation using an internal isFIPS flag for each SSP. This flag indicates whether the SSP can be used in approved or non-approved services.

The module does not perform persistent storage of SSPs.

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
API input parameters (plaintext)	Calling application within TOEPP	Cryptographic module	Plaintext	Manual	Electronic	
API input parameters (encrypted)	Calling application within TOEPP	Cryptographic module	Encrypted	Manual	Electronic	Key Unwrapping with AES (KTS)
API output parameters (plaintext)	Cryptographic module	Calling application within TOEPP	Plaintext	Manual	Electronic	
API output parameters (encrypted)	Cryptographic module	Calling application within TOEPP	Encrypted	Manual	Electronic	Key Wrapping with AES (KTS)

Table 16: SSP Input-Output Methods

CSPs (with the exception of passwords) can only be imported to and exported from the module when they are wrapped using an approved security function (e.g. AES KW/KWP or AES-GCM). PSPs can be imported and exported in plaintext. Import and export is performed using API input and output parameters.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Destroy Object	Destroys the SSP represented by the object	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable. The completion of the zeroization routine indicates that the zeroization procedure succeeded.	By calling the C_DestroyObject function.
Automatic	Automatically zeroized by the module when no longer needed	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable.	N/A
Remove power from the module	De-allocates the volatile memory used to store SSPs	Volatile memory used by the module is overwritten within nanoseconds when power is removed. Module power off indicates that the zeroization procedure succeeded.	By removing power

Table 17: SSP Zeroization Methods

All data output is inhibited during zeroization.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Symmetric Key	Symmetric key generated by the module	112-4096 bits - 112-256 bits	Symmetric key - CSP	Symmetric Key Generation with Hash_DRBG		Symmetric Key Generation with Hash_DRBG
AES Key	AES key used for encryption, decryption, wrapping, unwrapping, and for message authentication	128, 192, 256 bits - 128, 192, 256 bits	Symmetric key - CSP			Encryption with AES Decryption with AES Authenticated Encryption with AES Authenticated Decryption with AES Message Authenticatio

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
						n with CMAC
AES Key-Encryption Key	AES key used for wrapping CSPs	128, 192, 256 bits - 128, 192, 256 bits	Symmetric key - CSP			Key Wrapping with AES (KTS)
AES Key-Decryption Key	AES key used for unwrapping CSPs	128, 192, 256 bits - 128, 192, 256 bits	Symmetric key - CSP			Key Unwrapping with AES (KTS)
Wrapped CSP	Wrapped CSP imported/exported through AES key wrapping	112-8192 bits - 112-256 bits	Imported/Exported CSP - CSP		Key Wrapping with AES (KTS) Key Unwrapping with AES (KTS)	Key Wrapping with AES (KTS) Key Unwrapping with AES (KTS)
HMAC Key	HMAC key used for message authentication	112-256 bits - 112-256 bits	Symmetric key - CSP			Message Authentication with HMAC
Key-Derivation Key	Symmetric key used to derive symmetric keys	112-4096 bits - 112-256 bits	Symmetric key - CSP			Key Derivation with KBKDF
Shared Secret	Shared secret generated by (EC) Diffie-Hellman	256-8192 bits - 112-256 bits	Shared secret - CSP		Shared Secret Computation with KAS-FFC-SSC Shared Secret Computation with KAS-ECC-SSC	Key Derivation with HKDF Key Derivation with TLS 1.2 KDF Key Derivation with IKEv2 KDF
Password	Password used to derive symmetric keys	8-128 characters - N/A	Password - CSP			Key Derivation with PBKDF2

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
KBKDF Derived Key	Symmetric key derived from a key-derivation key	112-4096 bits - 112-256 bits	Symmetric key - CSP	Key Derivation with KBKDF		
PBKDF2 Derived Key	Symmetric key derived from a password	112-4096 bits - 112-256 bits	Symmetric key - CSP	Key Derivation with PBKDF2		
HKDF Derived Key	Symmetric key derived from a shared secret using HKDF	112-4096 bits - 112-256 bits	Symmetric key - CSP	Key Derivation with HKDF		
TLS Derived Key	Symmetric key derived from a shared secret using TLS 1.2 KDF	112-4096 bits - 112-256 bits	Symmetric key - CSP	Key Derivation with TLS 1.2 KDF		
IKE Derived Key	Symmetric key derived from a shared secret using IKEv2 KDF	112-4096 bits - 112-256 bits	Symmetric key - CSP	Key Derivation with IKEv2 KDF		
Entropy Input	Entropy input used to seed the DRBG (IG D.L compliant)	440-880 bits - 390-781 bits	Entropy input - CSP	Random Number Generation with Hash_DRBG		Random Number Generation with Hash_DRBG
DRBG Seed	DRBG seed derived from entropy input (IG D.L compliant)	440 bits - 256 bits	Seed - CSP	Random Number Generation with Hash_DRBG		Random Number Generation with Hash_DRBG
DRBG Internal State (V, C)	Internal state of the Hash_DRBG (IG D.L compliant)	880 bits - 256 bits	Internal state - CSP	Random Number Generation with Hash_DRBG		Random Number Generation with Hash_DRBG

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Module-generated FFC Private Key	KAS-FFC-SSC Private key generated by the module	2048-8192 bits - 112-200 bits	Private key - CSP	Key Pair Generation with Safe Primes		Key Pair Generation with Safe Primes
Module-generated FFC Public Key	KAS-FFC-SSC Public key generated by the module	2048-8192 bits - 112-200 bits	Public key - PSP	Key Pair Generation with Safe Primes		Key Pair Generation with Safe Primes
FFC Private Key	Private key used for Diffie-Hellman	2048-8192 bits - 112-200 bits	Private key - CSP			Shared Secret Computation with KAS-FFC-SSC
FFC Public Key	Public key used for Diffie-Hellman	2048-8192 bits - 112-200 bits	Public key - PSP			Shared Secret Computation with KAS-FFC-SSC
Module-generated EC Private Key	EC Private key generated by the module	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP	Key Pair Generation with ECDSA		Key Pair Generation with ECDSA
Module-generated EC Public Key	EC Public key generated by the module	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP	Key Pair Generation with ECDSA		Key Pair Generation with ECDSA
EC Private Key	Private key used for EC Diffie-Hellman and ECDSA	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP			Shared Secret Computation with KAS-ECC-SSC Signature Generation with ECDSA
EC Public Key	Public key used for EC Diffie-Hellman and ECDSA	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP			Shared Secret Computation with KAS-ECC-SSC Signature Verification with ECDSA

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Module-generated RSA Private Key	RSA Private key generated by the module	2048, 3072, 4096 bits - 112-150 bits	Private key - CSP	Key Pair Generation with RSA		Key Pair Generation with RSA
Module-generated RSA Public Key	RSA Public key generated by the module	2048, 3072, 4096 bits - 112-150 bits	Public key - PSP	Key Pair Generation with RSA		Key Pair Generation with RSA
RSA Private Key	Private key used for signature generation and asymmetric decryption with RSA	2048, 3072, 4096 bits - 112-150 bits	Private key - CSP			Signature Generation with RSA Authenticated Decryption with RSA
RSA Public Key	Public key used for signature verification and asymmetric encryption with RSA	SigVer: 1024, 1280, 1536, 1792, 2048, 3072, 4096 bits; Other usages: 2048, 3072, 4096 bits - SigVer: 80-150 bits; Other usages: 112-150 bits	Public key - PSP			Signature Verification with RSA Signature Verification with RSA (legacy use) Authenticated Encryption with RSA
Intermediate key generation value	Temporary value generated during key generation services	256-8192 bits - 112-256 bits	Intermediate value - CSP	Key Pair Generation with RSA Key Pair Generation with		Key Pair Generation with RSA Key Pair Generation with ECDSA

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
				ECDSA Key Pair Generation with Safe Primes		Key Pair Generation with Safe Primes

Table 18: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Symmetric Key	API output parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	
AES Key	API input parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	
AES Key-Encryption Key	API input parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	
AES Key-Decryption Key	API input parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	
Wrapped CSP	API input parameters (encrypted) API output parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	AES Key-Encryption Key:Wrapped by AES Key-Decryption Key:Unwrapped by
HMAC Key	API input parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Key-Derivation Key	API input parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	KBKDF Derived Key:Derivation Of
Shared Secret	API input parameters (encrypted) API output parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	FFC Private Key:Derived From FFC Public Key:Derived From EC Private Key:Derived From EC Public Key:Derived From HKDF Derived Key:Derivation Of TLS Derived Key:Derivation Of IKE Derived Key:Derivation Of
Password	API input parameters (plaintext)	RAM:Plaintext	For the duration of the service	Destroy Object Remove power from the module	PBKDF2 Derived Key:Derivation Of
KBKDF Derived Key	API output parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Key-Derivation Key:Derived From
PBKDF2 Derived Key	API output parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Password:Derived From
HKDF Derived Key	API output parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Shared Secret:Derived From
TLS Derived Key	API output parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove	Shared Secret:Derived From

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
				power from the module	
IKE Derived Key	API output parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Shared Secret:Derived From
Entropy Input		RAM:Plaintext	From generation until DRBG Seed is created	Automatic Remove power from the module	DRBG Seed:Derivation Of
DRBG Seed		RAM:Plaintext	While the DRBG is instantiated	Automatic Remove power from the module	Entropy Input:Derived From DRBG Internal State (V, C):Generation Of
DRBG Internal State (V, C)		RAM:Plaintext	While the module is operational	Remove power from the module	DRBG Seed:Generated From
Module-generated FFC Private Key	API output parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Module-generated FFC Public Key:Paired With Intermediate key generation value:Generated From
Module-generated FFC Public Key	API output parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Module-generated FFC Private Key:Paired With Intermediate key generation value:Generated From
FFC Private Key	API input parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	FFC Public Key:Paired With Shared Secret:Derivation Of
FFC Public Key	API input parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove	FFC Private Key:Paired With Shared Secret:Derivation Of

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
				power from the module	
Module-generated EC Private Key	API output parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Module-generated EC Public Key:Paired With Intermediate key generation value:Generated From
Module-generated EC Public Key	API output parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Module-generated EC Private Key:Paired With Intermediate key generation value:Generated From
EC Private Key	API input parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	EC Public Key:Paired With Shared Secret:Derivation Of
EC Public Key	API input parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	EC Private Key:Paired With Shared Secret:Derivation Of
Module-generated RSA Private Key	API output parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Module-generated RSA Public Key:Paired With Intermediate key generation value:Generated From
Module-generated RSA Public Key	API output parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Module-generated RSA Private Key:Paired With Intermediate key generation value:Generated From

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
RSA Private Key	API input parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	RSA Public Key:Paired With
RSA Public Key	API input parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	RSA Private Key:Paired With
Intermediate key generation value		RAM:Plaintext	For the duration of the service	Automatic	Module-generated FFC Private Key:Generation Of Module-generated FFC Public Key:Generation Of Module-generated EC Private Key:Generation Of Module-generated EC Public Key:Generation Of Module-generated RSA Private Key:Generation Of Module-generated RSA Public Key:Generation Of

Table 19: SSP Table 2

9.5 Transitions

The SHA-1 algorithm as implemented by the module will be non-approved for all purposes, starting January 1, 2031.

10 Self-Tests

Upon initialization, the module immediately performs all libfreeblpriv3.so cryptographic algorithm self-tests (CASTs) as specified in the Conditional Self-Tests table. These self-tests include the CAST for the HMAC-SHA2-256 algorithm that is used to verify the integrity of the module's shared libraries.

When all those self-tests pass successfully, the module automatically performs the pre-operational integrity test on the libfreeblpriv3.so file using its associated check value.

Then, the module performs the RSA CAST in the libsoftokn3.so library, followed by the pre-operational integrity test on the libsoftokn3.so file using its associated check value.

Finally, all remaining CASTs for the algorithms implemented in libsoftokn3.so are executed (see the Conditional Self-Tests table).

Only if all CASTs and pre-operational integrity tests passed successfully, the module transitions to the operational state. No operator intervention is required to reach this point.

While the module is executing the self-tests, services are not available, and data output (via the data output interface) is inhibited until the tests are successfully completed. If any of the self-tests fails, an error message is returned, and the module transitions to an error state.

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256	256-bit key	Message authentication	SW/FW Integrity	Module becomes operational and services are available for use	Integrity test for libsoftokn3.so and libfreeblpriv3.so

Table 20: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA2-224 (A6443)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization
SHA2-224 (A6447)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA2-256 (A6443)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization
SHA2-256 (A6447)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization
SHA2-384 (A6443)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization
SHA2-384 (A6447)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization
SHA2-512 (A6443)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization
SHA2-512 (A6447)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization
HMAC-SHA2-224 (A6443)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-224 (A6447)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
HMAC-SHA2-256 (A6443)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
HMAC-SHA2-256 (A6447)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
HMAC-SHA2-384 (A6443)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
HMAC-SHA2-384 (A6447)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
HMAC-SHA2-512 (A6443)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
HMAC-SHA2-512 (A6447)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-ECB - Encrypt (A6443)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-ECB - Encrypt (A6445)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-ECB - Decrypt (A6443)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-ECB - Decrypt (A6445)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CBC - Encrypt (A6443)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC - Encrypt (A6445)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC - Decrypt (A6443)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC - Decrypt (A6445)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM - Encrypt (A6443)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Encrypt (A6445)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Encrypt (A6446)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Decrypt (A6443)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM - Decrypt (A6445)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM - Decrypt (A6446)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CMAC (A6443)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
AES-CMAC (A6445)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
KDF SP800-108 (A6443)	HMAC-SHA2-256 in counter mode	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Module initialization
KDA HKDF SP800-56Cr2 (A6442)	SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Module initialization
TLS v1.2 KDF RFC7627 (A6443)	SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Module initialization
KDF IKEv2 (A6444)	SHA-1, SHA-256, SHA-384, SHA-512	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Module initialization
PBKDF (A6443)	SHA2-256 with 5 iterations, 128-bit salt and 14 characters password	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
Hash DRBG (A6443)	SHA-256 without prediction resistance	KAT	CAST	Module becomes operational and services are available for use	Instantiate Generate; Reseed Generate (compliant to SP 800-90A Rev. 1, Section 11.3)	Module initialization
KAS-FFC-SSC Sp800-56Ar3 (A6443)	ffdhe2048	KAT	CAST	Module becomes operational and services are available for use	Shared Secret Computation	Module initialization
KAS-ECC-SSC Sp800-56Ar3 (A6443)	P-256	KAT	CAST	Module becomes operational and services are available for use	Shared Secret Computation	Module initialization
RSA SigGen (FIPS186-5) (A6443)	PKCS#1 v1.5 with SHA2-256, SHA2-384, SHA2-512, and 2048-bit key	KAT	CAST	Module becomes operational and services are available for use	Signature Generation	Module initialization
RSA SigVer (FIPS186-5) (A6443)	PKCS#1 v1.5 with SHA2-256, SHA2-384, SHA2-512, and 2048-bit key	KAT	CAST	Module becomes operational and services are available for use	Signature Verification	Module initialization
ECDSA SigGen (FIPS186-5) (A6443)	SHA2-256 and P-256	KAT	CAST	Module becomes operational and services are available for use	Signature Generation	Module initialization
ECDSA SigVer (FIPS186-5) (A6443)	SHA2-256 and P-256	KAT	CAST	Module becomes operational and services are available for use	Signature Verification	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
RSA KeyGen (FIPS186-5) (A6443)	PKCS#1 v1.5 with SHA-256	PCT	PCT	Successful key pair generation	Signature Generation and Signature Verification	Key Pair Generation
ECDSA KeyGen (FIPS186-5) - SSC (A6443)	N/A	PCT	PCT	Successful key pair generation	PCT according to section 5.6.2.1.4 of SP 800-56A Rev. 3	Key Pair Generation
ECDSA KeyGen (FIPS186-5) - SigGen / SigVer (A6443)	SHA-256	PCT	PCT	Successful key pair generation	Signature Generation and Signature Verification	Key Pair Generation
Safe Primes Key Generation (A6443)	N/A	PCT	PCT	Successful key pair generation	PCT according to section 5.6.2.1.4 of SP 800-56A Rev. 3	Key Pair Generation

Table 21: Conditional Self-Tests

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256	Message authentication	SW/FW Integrity	On demand	Manually

Table 22: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
SHA2-224 (A6443)	KAT	CAST	On demand	Manually
SHA2-224 (A6447)	KAT	CAST	On demand	Manually
SHA2-256 (A6443)	KAT	CAST	On demand	Manually
SHA2-256 (A6447)	KAT	CAST	On demand	Manually
SHA2-384 (A6443)	KAT	CAST	On demand	Manually
SHA2-384 (A6447)	KAT	CAST	On demand	Manually
SHA2-512 (A6443)	KAT	CAST	On demand	Manually
SHA2-512 (A6447)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-224 (A6443)	KAT	CAST	On demand	Manually
HMAC-SHA2-224 (A6447)	KAT	CAST	On demand	Manually
HMAC-SHA2-256 (A6443)	KAT	CAST	On demand	Manually
HMAC-SHA2-256 (A6447)	KAT	CAST	On demand	Manually
HMAC-SHA2-384 (A6443)	KAT	CAST	On demand	Manually
HMAC-SHA2-384 (A6447)	KAT	CAST	On demand	Manually
HMAC-SHA2-512 (A6443)	KAT	CAST	On demand	Manually
HMAC-SHA2-512 (A6447)	KAT	CAST	On demand	Manually
AES-ECB - Encrypt (A6443)	KAT	CAST	On demand	Manually
AES-ECB - Encrypt (A6445)	KAT	CAST	On demand	Manually
AES-ECB - Decrypt (A6443)	KAT	CAST	On demand	Manually
AES-ECB - Decrypt (A6445)	KAT	CAST	On demand	Manually
AES-CBC - Encrypt (A6443)	KAT	CAST	On demand	Manually
AES-CBC - Encrypt (A6445)	KAT	CAST	On demand	Manually
AES-CBC - Decrypt (A6443)	KAT	CAST	On demand	Manually
AES-CBC - Decrypt (A6445)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A6443)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A6445)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-GCM - Encrypt (A6446)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A6443)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A6445)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A6446)	KAT	CAST	On demand	Manually
AES-CMAC (A6443)	KAT	CAST	On demand	Manually
AES-CMAC (A6445)	KAT	CAST	On demand	Manually
KDF SP800-108 (A6443)	KAT	CAST	On demand	Manually
KDA HKDF SP800-56Cr2 (A6442)	KAT	CAST	On demand	Manually
TLS v1.2 KDF RFC7627 (A6443)	KAT	CAST	On demand	Manually
KDF IKEv2 (A6444)	KAT	CAST	On demand	Manually
PBKDF (A6443)	KAT	CAST	On demand	Manually
Hash DRBG (A6443)	KAT	CAST	On demand	Manually
KAS-FFC-SSC Sp800-56Ar3 (A6443)	KAT	CAST	On demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A6443)	KAT	CAST	On demand	Manually
RSA SigGen (FIPS186-5) (A6443)	KAT	CAST	On demand	Manually
RSA SigVer (FIPS186-5) (A6443)	KAT	CAST	On demand	Manually
ECDSA SigGen (FIPS186-5) (A6443)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
ECDSA SigVer (FIPS186-5) (A6443)	KAT	CAST	On demand	Manually
RSA KeyGen (FIPS186-5) (A6443)	PCT	PCT	On demand	Manually
ECDSA KeyGen (FIPS186-5) - SSC (A6443)	PCT	PCT	On demand	Manually
ECDSA KeyGen (FIPS186-5) - SigGen / SigVer (A6443)	PCT	PCT	On demand	Manually
Safe Primes Key Generation (A6443)	PCT	PCT	On demand	Manually

Table 23: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Power-On Error	An error occurred during the self-tests executed on power-on	Software integrity test failure or CAST failure	Restart of the module	Module will not load
PCT Error	An error occurred during a PCT	PCT failure	Restart of the module	Module stops functioning (sftk_fatalError is set to TRUE)

Table 24: Error States

In any error state, the output interface is inhibited, and the module accepts no more inputs or requests.

10.5 Operator Initiation of Self-Tests

The software integrity tests and CASTs can be invoked on demand by unloading and subsequently re-initializing the module. The PCTs can be invoked on demand by requesting the Key Pair Generation service.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

The module is delivered as part of the following RPM packages:

- Red Hat Enterprise Linux 8 system:
  - nss-softokn-3.101.0-11.el8_8.x86_64
  - nss-softokn-freebl-3.101.0-11.el8_8.x86_64

Before these packages are installed, the Red Hat Enterprise Linux 8 system must operate in the FIPS validated configuration. This can be achieved by:

- Adding the fips=1 option to the kernel command line during the system installation. During the software selection stage, do not install any third-party software.
- Switching the system into the FIPS validated configuration after the installation. Execute the fips-mode-setup --enable command. Restart the system.

In both cases, once the system has rebooted, the Crypto Officer must verify the system operates in the FIPS validated configuration by executing the fips-mode-setup --check command, which should output "FIPS mode is enabled."

After the completion of the above-mentioned system configuration steps, the nss-softokn-3.101.0-11.el8_8.x86_64 and the nss-softokn-freebl-3.101.0-11.el8_8.x86_64 RPM packages can be installed using any of the package manager tools specified in Section 11.2.

After the RPM packages are installed, the Crypto Officer must execute the "Show name and version" service by accessing the CKA_NSS_VALIDATION_MODULE_ID attribute of the CKO_NSS_VALIDATION object in the default slot. The object attribute must contain the value:

Red Hat Enterprise Linux 8 nss 3.101.0-36157fa50e4c0485

Alternatively, the /usr/lib64/nss/unsupported-tools/validation tool is provided as a convenience by the nss-tools-3.101.0-11.el8_8.x86_64 package. This tool performs the same steps, and also outputs the FIPS module identifier as above.

To operate as the FIPS 140-3 validated module, the output of both the `fips-mode-setup --check` command and the "Show Name and Version" service must match the values specified above. If either output differs, the module cannot be considered the FIPS 140-3 validated module.

11.2 Administrator Guidance

The version of the RPMs containing the FIPS validated Module is stated in Section 11.1. The RPM packages forming the Module can be installed by standard tools recommended for the installation of RPM packages on a Red Hat Enterprise Linux system (for example, dnf, rpm, and the RHN remote management tool). All RPM packages are signed with the Red Hat build key, which is an RSA 4096-bit key using SHA-256 signatures. The signature is automatically verified upon installation of the RPM package. If the signature cannot be validated, the RPM tool rejects the installation of the package. In such a case, the Crypto Officer is requested to obtain a new copy of the module's RPMs from Red Hat.

11.3 Non-Administrator Guidance

There is no non-administrator guidance.

11.4 End of Life

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. Then, if desired, both the nss-softokn-3.101.0-11.el8_8.x86_64 and the nss-softokn-freebl-3.101.0-11.el8_8.x86_64 RPM packages can be uninstalled from the Red Hat Enterprise Linux 8 system.

12 Mitigation of Other Attacks

12.1 Attack List

Timing attacks on RSA

- RSA blinding: timing attack on RSA was first demonstrated by Paul Kocher in 1996, who contributed the mitigation code to our module. Most recently Boneh and Brumley showed that RSA blinding is an effective defense against timing attacks on RSA.
  - Specific Limit: None

Cache-timing attacks on the modular exponentiation operation used in RSA

- Cache invariant module exponentiation: this is a variant of a modular exponentiation implementation that Colin Percival showed to defend against cache-timing attacks
  - Specific Limit: this mechanism requires intimate knowledge of the cache line sizes of the processor. The mechanism may be ineffective when the module is running on a processor whose cache line sizes are unknown.

Arithmetic errors in RSA signatures

- Double-checking RSA signatures: arithmetic errors in RSA signatures might leak the private key. Ferguson and Schneier recommend that every RSA signature generation should verify the signature just generated.
  - Specific Limit: None

Appendix A. Glossary and abbreviations

AES	Advanced Encryption Standard
API	Application Programming Interface
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CKG	Cryptographic Key Generation
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter
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
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
HKDF	HMAC-based Key Derivation Function
HMAC	Keyed-Hash Message Authentication Code
IFC	Integer Factorization Cryptography
IKE	Internet Key Exchange
KAS	Key Agreement Scheme
KAT	Known Answer Test
KBKDF	Key-based Key Derivation Function
KTS	Key Transport Scheme
KW	Key Wrap
KWP	Key Wrap with Padding
MAC	Message Authentication Code
NIST	National Institute of Science and Technology

PAA	Processor Algorithm Acceleration
PAI	Processor Algorithm Implementation
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
SSC	Shared Secret Computation
SSP	Sensitive Security Parameter
TLS	Transport Layer Security
XOF	Extendable Output Function

Appendix B. References

FIPS 140-3 **FIPS PUB 140-3 - Security Requirements For Cryptographic Modules**

March 2019

https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf

FIPS 140-3 IG **Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program**

23 October 2024

https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements

FIPS 180-4 **Secure Hash Standard (SHS)**
August 2015
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

FIPS 186-2 **Digital Signature Standard (DSS)**
July 2013
https://csrc.nist.gov/files/pubs/fips/186-2/final/docs/fips186-2.pdf

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

FIPS 186-5 **Digital Signature Standard (DSS)**
February 2023
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf

FIPS 197 **Advanced Encryption Standard**
May 2023
https://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

FIPS 198-1 **The Keyed Hash Message Authentication Code (HMAC)**
July 2008
https://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 2.2**
November 2016
https://www.rfc-editor.org/rfc/rfc8017.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 5288	**AES Galois Counter Mode (GCM) Cipher Suites for TLS** August 2008 https://www.ietf.org/rfc/rfc5288.txt
RFC 7919	**Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)** August 2016 https://www.ietf.org/rfc/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	**Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 https://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
SP 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	**Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 https://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf
SP 800-38D	**Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 https://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf
SP 800-38F	**Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP 800-52r2	**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	**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 800-56Br2	**Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography** March 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Br2.pdf
SP 800-56Cr2	**Recommendation for Key-Derivation Methods in Key-Establishment Schemes** August 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf
SP 800-90Ar1	**Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP 800-90B	**Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf
SP 800-108r1	**NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions** August 2022 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-108r1-upd1.pdf
SP 800-131Ar2	**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	**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	**Recommendation for Cryptographic Key Generation** June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf
SP 800-135r1	**Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf