Red Hat, Inc

Red Hat Enterprise Linux 9 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

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 3.90.0-4408e3bb8a34af3a of the Red Hat Enterprise Linux 9 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

2 – Cryptographic Module Specification

2.1 Description

Purpose and Use:

The Red Hat Enterprise Linux 9 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: MultiChipStand

Cryptographic Boundary:

The cryptographic boundary consists only of the Softoken and Freebl 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.

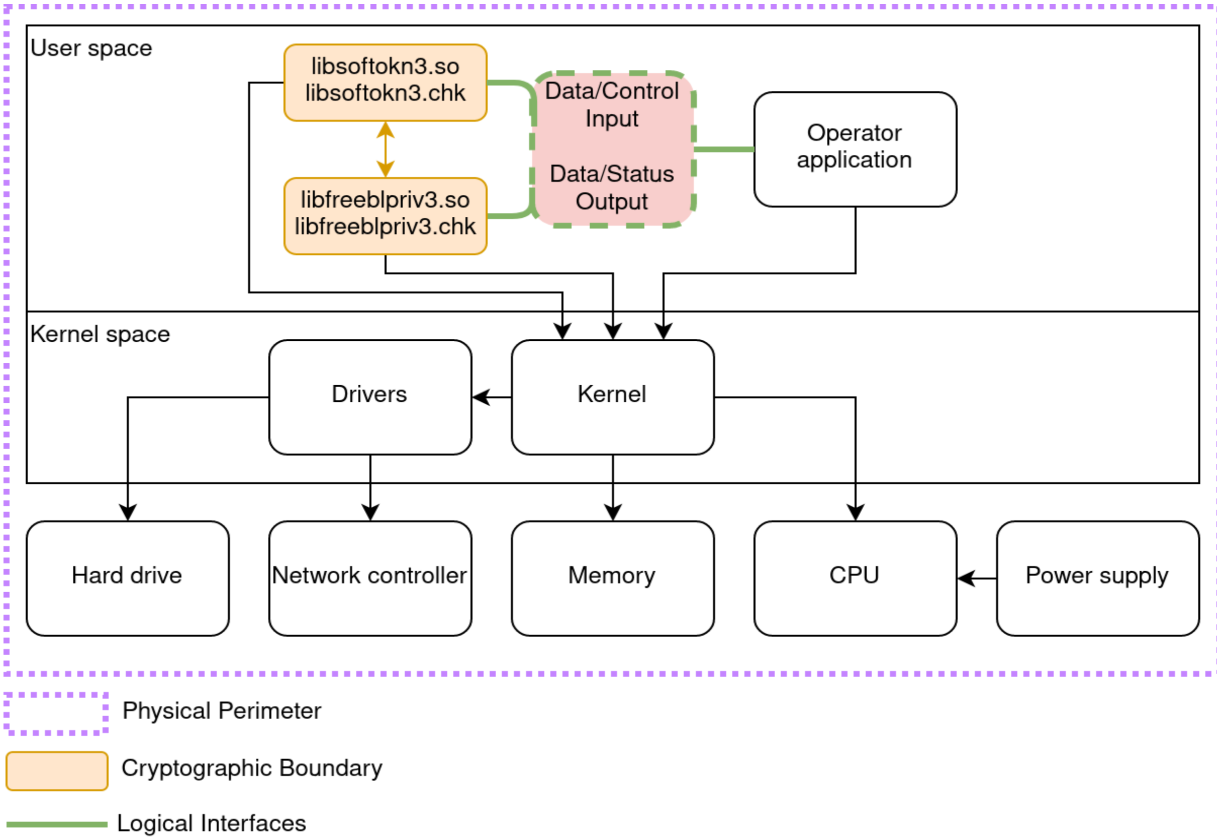

Figure 1: Block Diagram

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 on Dell PowerEdge	3.90.0-4408e3bb8a34af3a	N/A	HMAC-SHA-256
libsoftokn3.so, libfreeblpriv3.so on IBM 9080-HEX	3.90.0-4408e3bb8a34af3a	N/A	HMAC-SHA-256

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libsoftokn3.so, libfreeblpriv3.so on IBM z16 3931-A01	3.90.0-4408e3bb8a34af3a	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 9	Dell PowerEdge R440	Intel Cascade Lake Xeon Silver 4216	Yes	N/A	3.90.0-4408e3bb8a34af3a
Red Hat Enterprise Linux 9	Dell PowerEdge R440	Intel Cascade Lake Xeon Silver 4216	No	N/A	3.90.0-4408e3bb8a34af3a
Red Hat Enterprise Linux 9	IBM 9080-HEX	IBM POWER10	Yes	PowerVM FW1040.00 with VIOS 3.1.3.00	3.90.0-4408e3bb8a34af3a
Red Hat Enterprise Linux 9	IBM 9080-HEX	IBM POWER10	No	PowerVM FW1040.00 with VIOS 3.1.3.00	3.90.0-4408e3bb8a34af3a
Red Hat Enterprise Linux 9	IBM z16 3931-A01	IBM z16	Yes	N/A	3.90.0-4408e3bb8a34af3a
Red Hat Enterprise Linux 9	IBM z16 3931-A01	IBM z16	No	N/A	3.90.0-4408e3bb8a34af3a

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

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

Operating System	Hardware Platform
Red Hat Enterprise Linux 9	Intel(R) Xeon(R) E5

Table 4: Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid

CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

2.3 Excluded Components

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.
Non-Approved	Automatically entered whenever a non-approved service is requested.	Non-Approved	Equivalent to the indicator of the requested service.

Table 5: 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.

Degraded Mode Description:

The module does not implement a degraded mode of operation.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A4987	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A4994	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC-CS1	A4992	Direction - decrypt, encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A4989	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B
AES-CTR	A4987	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CTR	A4994	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A4987	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A4994	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A4987	Direction - Decrypt, Encrypt IV Generation - External, Internal IV Generation Mode - 8.2.1, 8.2.2 Key Length - 128, 192, 256	SP 800-38D
AES-GCM	A4994	Direction - Decrypt, Encrypt IV Generation - External, Internal IV Generation Mode - 8.2.1, 8.2.2 Key Length - 128, 192, 256	SP 800-38D
AES-GCM	A5559	Direction - Decrypt, Encrypt IV Generation - External, Internal Key Length - 128, 192, 256 IV Generation Mode - 8.2.1	SP 800-38D
AES-KW	A4988	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KW	A4993	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F

Algorithm	CAVP Cert	Properties	Reference
AES-KWP	A4988	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KWP	A4993	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
ECDSA KeyGen (FIPS186-5)	A4987	Curve - P-256, P-384, P-521 Secret Generation Mode - testing candidates	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A4987	Curve - P-256, P-384, P-521 Component - No Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-5
ECDSA SigVer (FIPS186-5)	A4987	Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-5
Hash DRBG	A4987	Prediction Resistance - No, Yes Mode - SHA2-256	SP 800-90A Rev. 1
HMAC-SHA-1	A4987	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A4987	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A4987	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A4987	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A4987	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A4987	Domain Parameter Generation Methods - P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-FFC-SSC Sp800-56Ar3	A4987	Domain Parameter Generation Methods - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Scheme -	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
		dhEphem - KAS Role - initiator, responder	
KDA HKDF Sp800-56Cr1	A4986	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-65336 Increment 8 HMAC Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	SP 800-56C Rev. 2
KDF IKEv2 (CVL)	A4991	Diffie-Hellman Shared Secret Length - Diffie-Hellman Shared Secret Length: 224, 2048, 8192 Derived Keying Material Length - Derived Keying Material Length: 1056, 3072 Hash Algorithm - SHA-1, SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
KDF SP800-108	A4990	KDF Mode - Counter, Double Pipeline Iteration, Feedback Supported Lengths - Supported Lengths: 8, 72, 128, 776, 3456, 4096	SP 800-108 Rev. 1
PBKDF	A4987	Iteration Count - Iteration Count: 1000-10000 Increment 1 Password Length - Password Length: 8-128 Increment 1	SP 800-132
RSA KeyGen (FIPS186-5)	A4987	Key Generation Mode - probable Modulo - 2048, 3072, 4096, 8192 Primality Tests - 2pow100 Private Key Format - standard	FIPS 186-5
RSA SigGen (FIPS186-5)	A4987	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
RSA SigVer (FIPS186-2)	A4987	Signature Type - PKCS 1.5, PKCSPSS Modulo - 1536	FIPS 186-4
RSA SigVer (FIPS186-4)	A4987	Signature Type - PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-5)	A4987	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
Safe Primes Key Generation	A4987	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3
SHA2-224	A4987	Large Message Sizes - 1, 2, 4, 8 Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4

Algorithm	CAVP Cert	Properties	Reference
SHA2-256	A4987	Large Message Sizes - 1, 2, 4, 8 Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A4987	Large Message Sizes - 1, 2, 4, 8 Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A4987	Large Message Sizes - 1, 2, 4, 8 Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
TLS v1.2 KDF RFC7627 (CVL)	A4987	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1

Table 6: 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 7: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

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

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

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

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
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
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

Name	Use and Function
RSA	Asymmetric encryption, Asymmetric decryption
DSA	Parameter generation, Parameter verification, Key pair generation
DH (FIPS 186-type groups)	Key pair generation
RSA (< 2048 bits; > 4096 bits)	Key pair generation
Ed25519, X25519	Key pair generation
Symmetric key generation (< 112 bits)	Secret key generation

Table 8: 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	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CBC AES-CTR AES-ECB AES-CBC-CS1 AES-CBC AES-CTR AES-ECB
Decryption with AES	BC-UnAuth	Decryption using AES	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CBC AES-CTR AES-ECB AES-CBC-CS1 AES-CTR AES-CTR AES-ECB
Authenticated Encryption with AES	BC-Auth	Authenticated encryption using AES	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-GCM AES-GCM AES-GCM

Name	Type	Description	Properties	Algorithms
Authenticated Decryption with AES	BC-Auth	Authenticated decryption using AES	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-GCM AES-GCM AES-GCM
Key Derivation with PBKDF	PBKDF	Key derivation using PBKDF	Derived keys:112-256 bits	PBKDF
Key Derivation with KBKDF	KBKDF	Key derivation using KBKDF	Derived keys:112-256 bits	KDF SP800-108
Key Derivation with HKDF	KAS-56CKDF	Key derivation using HKDF	Derived keys:112-256 bits	KDA HKDF Sp800-56Cr1
Key Derivation with TLS 1.2 KDF	KAS-135KDF	Key derivation using TLS 1.2 KDF	Derived keys:112-256 bits	TLS v1.2 KDF RFC7627
Key Derivation with IKEv2 KDF	KAS-135KDF	Key derivation using IKEv2 KDF	Derived keys:112-256 bits	KDF IKEv2
Key Wrapping with AES	KTS-Wrap	Key wrapping using AES	Keys:128, 192, 256 bits with 128-256 bits of key strength; Compliant with IG D.G	AES-KW AES-KWP AES-KW AES-KWP AES-GCM AES-GCM AES-GCM
Key Unwrapping with AES	KTS-Wrap	Key unwrapping using AES	Keys:128, 192, 256 bits with 128-256 bits of key strength; Compliant with IG D.G	AES-KW AES-KWP AES-KW AES-KWP AES-GCM AES-GCM AES-GCM
Message Authentication with HMAC	MAC	Message authentication using HMAC	Keys:112-256 bits with 112-256 bits of key strength	HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512
Message Authentication with CMAC	MAC	Message authentication using CMAC	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CMAC

Name	Type	Description	Properties	Algorithms
Random Number Generation with Hash_DRBG	DRBG	Random number generation using Hash_DRBG	Hash:SHA2-256	Hash DRBG
Shared Secret Computation with KAS-ECC-SSC	KAS-SSC	Shared secret computation using KAS-ECC-SSC	Curves:P-256, P-384, P-521 with 128, 192 and 256 bits of strength; Compliant with IG D.F scenario 2(1)	KAS-ECC-SSC Sp800-56Ar3
Shared Secret Computation with KAS-FFC-SSC	KAS-SSC	Shared secret computation using KAS-FFC-SSC	Keys:MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 with 112-200 bits of key strength; Compliant with IG D.F scenario 2(1)	KAS-FFC-SSC Sp800-56Ar3
Signature Generation with RSA	DigSig-SigGen	Signature generation using RSA	Keys:2048, 3072, 4096 bits with 112-150 bits of key strength	RSA SigGen (FIPS186-5)
Signature Generation with ECDSA	DigSig-SigGen	Signature generation using ECDSA	Curves:P-256, P-384, P-521 with 128, 192 and 256 bits of strength	ECDSA SigGen (FIPS186-5)
Signature Verification with RSA	DigSig-SigVer	Signature verification using RSA	Keys:1024, 1280, 1536, 1792, 2048, 3072, 4096 bits with 80-150 bits of key strength	RSA SigVer (FIPS186-2) RSA SigVer (FIPS186-4) RSA SigVer (FIPS186-5)
Signature Verification with ECDSA	DigSig-SigVer	Signature verification using ECDSA	Curves:P-256, P-384, P-521 with	ECDSA SigVer (FIPS186-5)

Name	Type	Description	Properties	Algorithms
			128, 192, 256 bits of strength	
Symmetric Key Generation with Hash_DRBG	CKG	Direct symmetric key generation using Hash_DRBG	Keys:112-256 bits with 112-256 bits of key strength; Compliant with SP800-133r2 section 6.1	Hash DRBG
Key Pair Generation with RSA	AsymKeyPair-KeyGen	Key pair generation using RSA	Keys:2048, 3072, 4096 bits with 112-150 bits of key strength	RSA KeyGen (FIPS186-5)
Key Pair Generation with ECDSA	AsymKeyPair-KeyGen	Key pair generation using ECDSA	Curves:P-256, P-384, P-521 with 128, 192 and 256 bits of strength	ECDSA KeyGen (FIPS186-5)
Key Pair Generation with Safe Primes	AsymKeyPair-KeyGen	Key pair generation using Safe Primes	Keys:MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 with 112-200 bits of key strength	Safe Primes Key Generation
Message Digest with SHA	SHA	Message digest using SHA		SHA2-256 SHA2-384 SHA2-512 SHA2-224

Table 9: 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 and 8446.

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 generated GCM IV is at least 96 bits in length where the size of the fixed (name) field is at least 32 bits. The module then internally generates a 32 bit or longer deterministic non-repetitive counter. The module increments the counter monotonically at each invocation of the AES-GCM for the same encryption key. The module explicitly checks for the wrap around and returns an error if wrap around condition is reached.

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.

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 RFC8446 of August 2018, using the cipher-suites that explicitly select AES GCM as the encryption/decryption cipher (Appendix B.4 of RFC8446). 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.

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 length of the MK or DPK shall be of 112 bits or more.
- Passwords or passphrases, used as an input for the PBKDF2, shall not be used as cryptographic keys.
- The length of the required password or passphrase is at least 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, with a length of at least 128 bits, shall be generated randomly using the SP 800-90Ar1 DRBG provided by the module.
- The iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users. The minimum value is 1000.

2.7.3 SP 800-56Ar3 Assurances

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 Sections 5.6.2.2.1 and 5.6.2.2.2 of SP 800-56Ar3.

2.7.4 RSA Approved Modulus Size

RSA Signature Verification is approved for 1024, 1280, 1536 and 1792-bit keys in compliance with IG C.F. The 1280 and 1792-bit keys cannot be ACVP tested. However, all modulus sizes in which testing is available have been tested by the CAVP.

2.7.5 Legacy Use

Digital signature verification 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

2.8 RBG and Entropy

Cert Number	Vendor Name
E47	Red Hat, Inc

Table 10: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Userspace CPU Time Jitter RNG Entropy Source Version 2.2.0	Non-Physical	Red Hat Enterprise Linux 9 on Dell PowerEdge R440 on Intel(R) Xeon(R) Silver 4216; Red Hat Enterprise Linux 9 on IBM z16 3931-A01 on IBM z16; Red Hat Enterprise Linux 9 on PowerVM FW1040.00 with VIOS 3.1.3.00 on IBM 9080 HEX on IBM POWER10	256 bits	225 bits	HMAC-SHA2-512

Table 11: 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 module generates SSPs (e.g., keys) whose strengths are modified by available entropy.

The module complies with the Public Use Document for ESV certificate E47 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. There are no maintenance requirements for the entropy source.

The entropy source is located within the module's physical perimeter, but outside of the module's cryptographic boundary.

2.9 Key Generation

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

- Direct generation of symmetric keys: compliant with SP 800-133r2, Section 6.1.
- Safe primes key pair generation: compliant with SP 800-133r2, Section 5.2, which maps to SP 800-56Ar3. The method described in Section 5.6.1.1.4 of SP 800-56Ar3 ("Testing Candidates") is used.
- RSA key pair generation: compliant with SP 800-133r2, Section 5.1, which maps to FIPS 186-5. The method described in Appendix A.1.3 of FIPS 186-5 ("Probable Primes") is used.
- ECC (ECDH and ECDSA) key pair generation: compliant with SP 800-133r2, Section 5.1, which maps to FIPS 186-5. The method described in Appendix A.2.2 of FIPS 186-5 ("Rejection Sampling") is used. Note that this generation method is also used to generate ECDH key pairs.

Additionally, the module implements the following key derivation methods:

- KBKDF: compliant with SP 800-108r1. This implementation can be used to generate secret keys from a pre-existing key-derivation-key.

- HKDF: compliant with SP 800-56Cr2. This implementation shall only be used to generate secret keys in the context of an SP 800-56Ar3 key agreement scheme.
- TLS 1.2 KDF, IKEv2 PRF: compliant with SP 800-135r1. These implementations shall only be used to generate secret keys in the context of the TLS 1.2, and IKEv2 protocols, respectively.
- PBKDF2: compliant with option 1a of SP 800-132. This implementation shall only be used to derive keys for use in storage applications.

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

2.10 Key Establishment

The module provides Diffie-Hellman (DH) and Elliptic Curve Diffie-Hellman (ECDH) shared secret computation compliant with SP800-56Ar3, in accordance with scenario 2 (1) of FIPS 140-3 IG D.F.

For Diffie-Hellman, the module supports the use of the safe primes defined in RFC 3526 (IKE) and RFC 7919 (TLS). Note that the module only implements domain parameter generation, 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 and IKEv2 PRF):

IKE (RFC 3526):

- MODP-2048 (ID = 14)
- MODP-3072 (ID = 15)
- MODP-4096 (ID = 16)
- MODP-6144 (ID = 17)
- MODP-8192 (ID = 18)

TLS (RFC 7919):

- ffdhe2048 (ID = 256)
- ffdhe3072 (ID = 257)
- ffdhe4096 (ID = 258)
- ffdhe6144 (ID = 259)
- ffdhe8192 (ID = 260)

According to FIPS 140-3 IG D.B, the key sizes of DH and ECDH shared secret computation provide 112-200 resp. 128-256 bits of security strength in an approved mode of operation.

The module also provides the following key transport mechanisms:

- Key wrapping using AES KW and AES KWP, with a security strength of 128, 192, or 256 bits, depending on the wrapping key size.
- Key wrapping using AES GCM with a security strength of 128, 192, or 256 bits.

2.11 Industry Protocols

For DH, 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 domain parameter generation, 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). No parts of the TLS and IKE protocols, other than the KDFs, have been tested by the CAVP or CMVP.

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.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
As a software-only module, the module does not have physical ports. Physical Ports are interpreted to be the physical ports of the hardware platform on which it runs.	Data Input	API input parameters
As a software-only module, the module does not have physical ports. Physical Ports are interpreted to be the physical ports of the hardware platform on which it runs.	Data Output	API output parameters
As a software-only module, the module does not have physical ports. Physical Ports are interpreted to be the physical ports of the hardware platform on which it runs.	Control Input	API function calls, API input parameters for control input
As a software-only module, the module does not have physical ports. Physical Ports are interpreted to be the physical ports of the hardware platform on which it runs.	Status Output	API return codes

Table 12: Ports and Interfaces

The logical interfaces are the APIs through which the applications request services. The module does not implement a control output interface.

4 Roles, Services, and Authentication

4.1 Authentication Methods

N/A for this module.

The module does not support authentication for roles.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 13: Roles

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

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 Encryption	Encrypt a plaintext	CKS_NSS_FIPS_OK	AES key, IV, plaintext	Ciphertext, MAC tag	Authenticated Encryption with AES	Crypto Officer - AES Key: W,E
Authenticated Decryption	Decrypt a ciphertext	CKS_NSS_FIPS_OK	AES key, IV, MAC tag, ciphertext	Plaintext or fail	Authenticated Decryption with AES	Crypto Officer - AES 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	Derived key	Key Derivation with KBKDF	Crypto Officer - Key-Derivation Key: W,E - Derived Key : G
Key Derivation	Derive a key from a shared secret	CKS_NSS_FIPS_OK (1)	Shared secret	Derived key	Key Derivation with HKDF Key Derivation with TLS 1.2 KDF Key Derivation with IKEv2 KDF	Crypto Officer - Shared Secret: W,E - Derived Key : G
Password-Based Key Derivation	Derive a key from a password	CKS_NSS_FIPS_OK (1)	Password, salt, iteration count	Derived key	Key Derivation with PBKDF	Crypto Officer - Password: W,E - Derived Key : G
Key Wrapping	Wrap a CSP	CKS_NSS_FIPS_OK (1)	AES key, any CSP	Wrapped CSP	Key Wrapping with AES	Crypto Officer - AES Key: W,E
Key Unwrapping	Unwrap a CSP	CKS_NSS_FIPS_OK (1)	AES key, Wrapped CSP	Any CSP	Key Unwrapping with AES	Crypto Officer - AES Key: W,E
Message Authentication	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 AES	Compute a MAC tag	CKS_NSS_FIPS_OK (1)	AES key	MAC tag	Message Authentication with CMAC	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- 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 - DRBG Seed : G,E - Internal State (V, C) : G,W,E
Shared Secret Computation with DH	Compute a shared secret	CKS_NSS_FIPS_OK	DH private key (owner), DH public key (peer)	Shared secret	Shared Secret Computation with KAS-FFC-SSC	Crypto Officer - DH Private Key : W,E - DH Public Key : W,E - Shared Secret: G
Shared Secret Computation with ECC	Compute a shared secret	CKS_NSS_FIPS_OK	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
Signature Generation with RSA	Generate a signature	CKS_NSS_FIPS_OK	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	EC private	Signature	Signature Generation with ECDSA	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
			key, message			- EC Private Key : W,E
Signature Verification with RSA	Verify a signature	CKS_NSS_FIPS_OK	RSA public key, message, signature	Pass/fail	Signature Verification with RSA	Crypto Officer - RSA Public Key: W,E
Signature Verification with ECDSA	Verify a signature	CKS_NSS_FIPS_OK	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	Group	DH public key, DH private key	Key Pair Generation with Safe Primes	Crypto Officer - DH Private Key : G - DH Public Key : G - Intermediate key generation value : G,E,Z
Key Pair Generation with RSA	Generate a key pair	CKS_NSS_FIPS_OK	Modulus bits	RSA public key, RSA private key	Key Pair Generation with RSA	Crypto Officer - RSA Private Key : G - RSA Public Key: G - Intermediate key generation value : G,E,Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Key Pair Generation with ECDSA	Generate a key pair	CKS_NSS_FIPS_OK	Curve	EC public key, EC private key	Key Pair Generation with ECDSA	Crypto Officer - EC Private Key : G - EC Public Key: G - Intermediate key generation value : G,E,Z
Symmetric Key Generation	Generate a secret key	CKS_NSS_FIPS_OK	Key size	AES key, HMAC key or key-derivation key	Symmetric Key Generation with Hash_DRBG	Crypto Officer - AES Key: G - HMAC Key : G - Key-Derivation Key: G
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 integrity tests	None	N/A	Pass/fail	None	Crypto Officer
Zeroization	Zeroize all SSPs	N/A	Any SSP	None	None	Crypto Officer - AES Key: Z - HMAC

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Key : Z - Key-Derivation Key: Z - Shared Secret: Z - Password: Z - Derived Key : Z - Entropy Input : Z - DRBG Seed : Z - Internal State (V, C) : Z - DH Private Key : Z - DH 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 14: Approved Services

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

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

- **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.
- **N/A:** The module does not access any SSP or key during its operation.

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

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	CO

Name	Description	Algorithms	Role
		keys) KBKDF (MD2, MD5) IKEv2 PRF (MD2, MD5)	
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
Signature Verification	Verify a signature	DSA RSA (< 1024-bit keys) ECDSA (component)	CO
Asymmetric Encryption	Encrypt a plaintext	RSA	CO
Asymmetric Decryption	Decrypt a plaintext	RSA	CO
Parameter Generation	Generate domain parameters	DSA	CO
Parameter Verification	Verify domain parameters	DSA	CO
Key Pair Generation	Generate a key pair	DSA DH (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 15: 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 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 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:

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. 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.

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 16: 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
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

Table 17: 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 or KWP). 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 18: SSP Zeroization Methods

All data output is inhibited during zeroization. Memory is deallocated after zeroization.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES Key	AES key used for encryption, decryption, and computing MAC tags	128, 192, 256 bits - 128, 192, 256 bits	Symmetric key - CSP	Symmetric Key Generation with Hash_DRBG		Encryption with AES Decryption with AES Authenticated Encryption with AES Authenticated Decryption with AES Key Wrapping

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
						with AES Key Unwrapping with AES Message Authentication with CMAC
HMAC Key	HMAC key used for computing MAC tags	112-256 bits - 112-256 bits	Symmetric key - CSP	Symmetric Key Generation with Hash_DRBG		Message Authentication with HMAC
Key-Derivation Key	Symmetric key used to derive symmetric keys	112-4096 bits - 112-256 bits	Symmetric key - CSP	Symmetric Key Generation with Hash_DRBG		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-ECC-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 PBKDF
Derived Key	Symmetric key derived from a key-derivation key, shared secret, or password	112-4096 bits - 112-256 bits	Symmetric key - CSP	Key Derivation with PBKDF Key Derivation with KBKDF Key Derivation with HKDF Key Derivation		

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
				with TLS 1.2 KDF Key Derivation with IKEv2 KDF		
Entropy Input	Entropy input used to seed the DRBG	128-384 bits - 128-256 bits	Entropy input - CSP			Random Number Generation with Hash_DRBG
DRBG Seed	DRBG seed derived from entropy input	440 bits - 256 bits	Seed - CSP	Random Number Generation with Hash_DRBG		Random Number Generation with Hash_DRBG
Internal State (V, C)	Internal state of the Hash_DRBG	880 bits - 256 bits	Internal state - CSP	Random Number Generation with Hash_DRBG		Random Number Generation with Hash_DRBG
DH Private Key	Private key used for Diffie-Hellman	2048-8192 bits - 112-200 bits	Private key - CSP	Key Pair Generation with Safe Primes		Shared Secret Computation with KAS-FFC-SSC
DH Public Key	Public key used for Diffie-Hellman	2048-8192 bits - 112-200 bits	Public key - PSP	Key Pair Generation with Safe Primes		Shared Secret Computation with KAS-FFC-SSC
EC Private Key	Private key used for EC Diffie-Hellman	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP	Key Pair Generation with ECDSA		Shared Secret Computation with KAS-ECC-SSC Signature Generation with ECDSA

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
EC Public Key	Public key used for EC Diffie-Hellman	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP	Key Pair Generation with ECDSA		Shared Secret Computation with KAS-ECC-SSC Signature Verification with ECDSA
RSA Private Key	Private key used for RSA signature generation	2048, 3072, 4096 bits - 112-150 bits	Private key - CSP	Key Pair Generation with RSA		Signature Generation with RSA
RSA Public Key	Public key used for RSA signature verification	KeyGen: 2048, 3072, 4096 bits; SigVer: 1024, 1280, 1536, 1792, 2048, 3072, 4096 bits - KeyGen: 112-150 bits; SigVer: 80-150 bits	Public key - PSP	Key Pair Generation with RSA		Signature Verification 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 ECDSA Key Pair Generation with Safe Primes		Key Pair Generation with RSA Key Pair Generation with ECDSA Key Pair Generation with Safe Primes

Table 19: SSP Table 1

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

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
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 Internal State (V, C) :Generation Of
Internal State (V, C)		RAM:Plaintext	While the module is operational	Remove power from the module	DRBG Seed :Generated From
DH Private Key	API input parameters (encrypted) API output parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	DH Public Key :Paired With Intermediate key generation value :Generated From
DH Public Key	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	DH Private Key :Paired With Intermediate key generation value :Generated From
EC Private Key	API input parameters (encrypted) API output parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	EC Public Key:Paired With Intermediate key generation value :Generated From
EC Public Key	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	EC Private Key :Paired With Intermediate key generation value :Generated From
RSA Private Key	API input parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove	RSA Public Key:Paired With Intermediate key

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
	API output parameters (encrypted)			power from the module	generation value :Generated From
RSA Public Key	API input parameters (plaintext) API output 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 :Generated From
Intermediate key generation value		RAM:Plaintext	For the duration of the service	Automatic	DH Private Key :Generation Of DH Public Key :Generation Of EC Private Key :Generation Of EC Public Key:Generation Of RSA Private Key :Generation Of RSA Public Key:Generation Of

Table 20: 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

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A4987)	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 21: Pre-Operational Self-Tests

Each software component of the module has an associated HMAC-SHA2-256 integrity check value. The software integrity tests ensure that the module is not corrupted. The HMAC-SHA2-256 algorithm goes through a CAST before the software integrity tests are performed.

Upon initialization, the module immediately performs all Freebl cryptographic algorithm self-tests (CASTs) as specified in the Conditional Self-Tests table. 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 Softoken library, followed by the pre-operational integrity test on the libsoftokn3.so file using its associated check value. The CAST for the algorithm used in the pre-operational self-test (i.e., HMAC-SHA2-256) is performed by the Freebl library, before the Softoken library integrity test. Finally, all remaining CASTs for the algorithms implemented in Softoken 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.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA-1 (A4987)	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
SHA2-224 (A4987)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization
SHA2-256 (A4987)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization
SHA2-384 (A4987)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization
SHA2-512 (A4987)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization
HMAC-SHA-1 (A4987)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
HMAC-SHA2-224 (A4987)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
HMAC-SHA2-256 (A4987)	288-bit key	KAT	CAST	Module becomes operational and	Message Authentication	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
HMAC-SHA2-384 (A4987)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
HMAC-SHA2-512 (A4987)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
AES-ECB (A4987)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-ECB (A4987)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-ECB (A4994)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-ECB (A4994)	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 (A4987)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC (A4987)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CBC (A4994)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC (A4994)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM (A4987)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM (A4987)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM (A4994)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and	Encryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
AES-GCM (A4994)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM (A5559)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM (A5559)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CMAC (A4989)	128, 192, 256-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
KDF SP800-108 (A4990)	HMAC-SHA2-256 in counter mode	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Module initialization
KDA HKDF Sp800-56Cr1 (A4986)	SHA2-256	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
TLS v1.2 KDF RFC7627 (A4987)	SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Module initialization
KDF IKEv2 (A4991)	SHA-1, SHA-256, SHA-384, SHA-512	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Module initialization
PBKDF (A4987)	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
Hash DRBG (A4987)	SHA-256 without prediction resistance	KAT	CAST	Module becomes operational and services are available for use	Instantiate Generate; Reseed Generate (compliant to SP 800- 90Ar1 Section 11.3)	Module initialization
KAS-FFC-SSC Sp800-56Ar3 (A4987)	ffdhe2048	KAT	CAST	Module becomes operational and services are available for use	Shared Secret Computation	Module initialization
KAS-ECC-SSC Sp800-56Ar3 (A4987)	P-256	KAT	CAST	Module becomes operational and services are available for use	Shared Secret Computation	Module initialization
RSA SigGen (FIPS186-5) (A4987)	PKCS#1 v1.5 with SHA2-256, SHA2-384,	KAT	CAST	Module becomes operational and	Signature Generation	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
	SHA2-512, and 2048-bit key			services are available for use		
RSA SigVer (FIPS186-5) (A4987)	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) (A4987)	SHA2-256 and P-256	KAT	CAST	Module becomes operational and services are available for use	Signature Generation	Module initialization
ECDSA SigVer (FIPS186-5) (A4987)	SHA2-256 and P-256	KAT	CAST	Module becomes operational and services are available for use	Signature Verification	Module initialization
Safe Primes Key Generation (A4987)	N/A	PCT	PCT	Successful key pair generation	PCT according to section 5.6.2.1.4 of [SP800-56Ar3]	Key Pair Generation
ECDSA KeyGen (FIPS186-5) (A4987)	N/A	PCT	PCT	Successful key pair generation	PCT according to section 5.6.2.1.4 of [SP800-56Ar3]	Key Pair Generation
ECDSA KeyGen (FIPS186-5) (A4987)	SHA-256	PCT	PCT	Successful key pair generation	Signature Generation and Signature Verification	Key Pair Generation
RSA KeyGen (FIPS186-5) (A4987)	PKCS#1 v1.5 with SHA-256	PCT	PCT	Successful key pair generation	Signature Generation and Signature Verification	Key Pair Generation

Table 22: Conditional Self-Tests

The module performs self-tests on all FIPS approved cryptographic algorithms as part of the approved services supported in the approved mode of operation, using the tests shown in the Conditional Self-Tests table above.

Upon generation of a key pair, the module will perform a pair-wise consistency test (PCT) as shown in the table above, which provides some assurance that the generated key pair is well formed. For DH and EC key pairs, these tests consist of the PCT described in Section 5.6.2.1.4 of SP 800-56Ar3. For RSA and EC key pairs, this test consists of a signature generation and a signature verification operation. Note that two PCTs are performed for EC key pairs.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A4987)	Message authentication	SW/FW Integrity	On demand	Manually

Table 23: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
SHA-1 (A4987)	KAT	CAST	On demand	Manually
SHA2-224 (A4987)	KAT	CAST	On demand	Manually
SHA2-256 (A4987)	KAT	CAST	On demand	Manually
SHA2-384 (A4987)	KAT	CAST	On demand	Manually
SHA2-512 (A4987)	KAT	CAST	On demand	Manually
HMAC-SHA-1 (A4987)	KAT	CAST	On demand	Manually
HMAC-SHA2-224 (A4987)	KAT	CAST	On demand	Manually
HMAC-SHA2-256 (A4987)	KAT	CAST	On demand	Manually
HMAC-SHA2-384 (A4987)	KAT	CAST	On demand	Manually
HMAC-SHA2-512 (A4987)	KAT	CAST	On demand	Manually
AES-ECB (A4987)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-ECB (A4987)	KAT	CAST	On demand	Manually
AES-ECB (A4994)	KAT	CAST	On demand	Manually
AES-ECB (A4994)	KAT	CAST	On demand	Manually
AES-CBC (A4987)	KAT	CAST	On demand	Manually
AES-CBC (A4987)	KAT	CAST	On demand	Manually
AES-CBC (A4994)	KAT	CAST	On demand	Manually
AES-CBC (A4994)	KAT	CAST	On demand	Manually
AES-GCM (A4987)	KAT	CAST	On demand	Manually
AES-GCM (A4987)	KAT	CAST	On demand	Manually
AES-GCM (A4994)	KAT	CAST	On demand	Manually
AES-GCM (A4994)	KAT	CAST	On demand	Manually
AES-GCM (A5559)	KAT	CAST	On demand	Manually
AES-GCM (A5559)	KAT	CAST	On demand	Manually
AES-CMAC (A4989)	KAT	CAST	On demand	Manually
KDF SP800-108 (A4990)	KAT	CAST	On demand	Manually
KDA HKDF Sp800-56Cr1 (A4986)	KAT	CAST	On demand	Manually
TLS v1.2 KDF RFC7627 (A4987)	KAT	CAST	On demand	Manually
KDF IKEv2 (A4991)	KAT	CAST	On demand	Manually
PBKDF (A4987)	KAT	CAST	On demand	Manually
Hash DRBG (A4987)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
KAS-FFC-SSC Sp800-56Ar3 (A4987)	KAT	CAST	On demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A4987)	KAT	CAST	On demand	Manually
RSA SigGen (FIPS186-5) (A4987)	KAT	CAST	On demand	Manually
RSA SigVer (FIPS186-5) (A4987)	KAT	CAST	On demand	Manually
ECDSA SigGen (FIPS186-5) (A4987)	KAT	CAST	On demand	Manually
ECDSA SigVer (FIPS186-5) (A4987)	KAT	CAST	On demand	Manually
Safe Primes Key Generation (A4987)	PCT	PCT	On demand	Manually
ECDSA KeyGen (FIPS186-5) (A4987)	PCT	PCT	On demand	Manually
ECDSA KeyGen (FIPS186-5) (A4987)	PCT	PCT	On demand	Manually
RSA KeyGen (FIPS186-5) (A4987)	PCT	PCT	On demand	Manually

Table 24: 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 25: 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 distributed within the following RPM packages for each of the tested operational environments:

- *Dell PowerEdge R440:* nss-softokn-3.90.0-6.el9_2.x86_64.rpm and nss-softokn-freebl-3.90.0-6.el9_2.x86_64.rpm
- *IBM z16 3931-A01:* nss-softokn-3.90.0-6.el9_2.s390x.rpm and nss-softokn-freebl-3.90.0-6.el9_2.s390x.rpm
- *IBM 9080-HEX:* nss-softokn-3.90.0-6.el9_2.ppc64le.rpm and nss-softokn-freebl-3.90.0-6.el9_2.ppc64le.rpm

Before the nss-softokn-3.90.0-6.el9_2 and nss-softokn-freebl-3.90.0-6.el9_2 RPM packages are installed, the RHEL 9 system must operate in the approved mode. 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. More information can be found at [the vendor documentation](#).
- Switching the system into the approved mode after the installation. Execute the `fips-mode-setup --enable` command. Restart the system. More information can be found at [the vendor documentation](#).

In both cases, the Crypto Officer must verify the RHEL 9 system operates in the approved mode by executing the `fips-mode-setup --check` command, which should output "FIPS mode is enabled."

After installation of the nss-softokn-3.90.0-6.el9_2 and nss-softokn-freebl-3.90.0-6.el9_2 RPM packages, the Crypto Officer must execute the "Show module 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 9 nss 3.90.0-4408e3bb8a34af3a`

Alternatively, the /usr/lib64/nss/unsupported-tools/validation tool is provided as a convenience by the nss-tools-3.90.0-6.el9_2 RPM package. This tool performs the same steps, and also outputs the FIPS module identifier as above.

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 Design and Rules

Not applicable.

11.5 Maintenance Requirements

There are no maintenance requirements.

11.6 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, the nss-softokn-3.90.0-6.el9_2 and nss-softokn-freebl-3.90.0-6.el9_2 RPM packages can be uninstalled from the RHEL 9 systems.

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