Amazon Web Services, Inc.

Amazon Linux 2023 NSS Cryptographic Module

FIPS 140-3 Non-Proprietary Security Policy

Document Version 1.1
Last update: 2025-04-25

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

Copyrights and Trademarks

Amazon is a registered trademark of Amazon Web Services, Inc. or its affiliates.

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for version 3.90.0-2e2ef5732e8af4d7 of the Amazon Linux 2023 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

This Security Policy describes the features and design of the module named Amazon Linux 2023 NSS Cryptographic Module using the terminology contained in the FIPS 140-3 specification. The FIPS 140-3 Security Requirements for Cryptographic Module specifies the security requirements that will be satisfied by a cryptographic module utilized within a security system protecting sensitive but unclassified information. The NIST/CCCS Cryptographic Module Validation Program (CMVP) validates cryptographic module to FIPS 140-3. Validated products are accepted by the Federal agencies of both the USA and Canada for the protection of sensitive or designated information.

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.

The vendor has provided the non-proprietary Security Policy of the cryptographic module, which was further consolidated into this document by atsec information security together with other vendor-supplied

documentation. In preparing the Security Policy document, the laboratory formatted the vendor-supplied documentation for consolidation without altering the technical statements therein contained. The further refining of the Security Policy document was conducted iteratively throughout the conformance testing, wherein the Security Policy was submitted to the vendor, who would then edit, modify, and add technical contents. The vendor would also supply additional documentation, which the laboratory formatted into the existing Security Policy, and resubmitted to the vendor for their final editing.

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The Amazon Linux 2023 NSS Cryptographic Module (hereafter referred to as "the module") 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

Module Characteristics:

Cryptographic Boundary: Figure 1 shows a block diagram that represents the design of the module when the module is operational and providing services to other user space applications. In this diagram, the physical perimeter of the operational environment (a general-purpose computer on which the module is installed) is indicated by a purple dashed line.

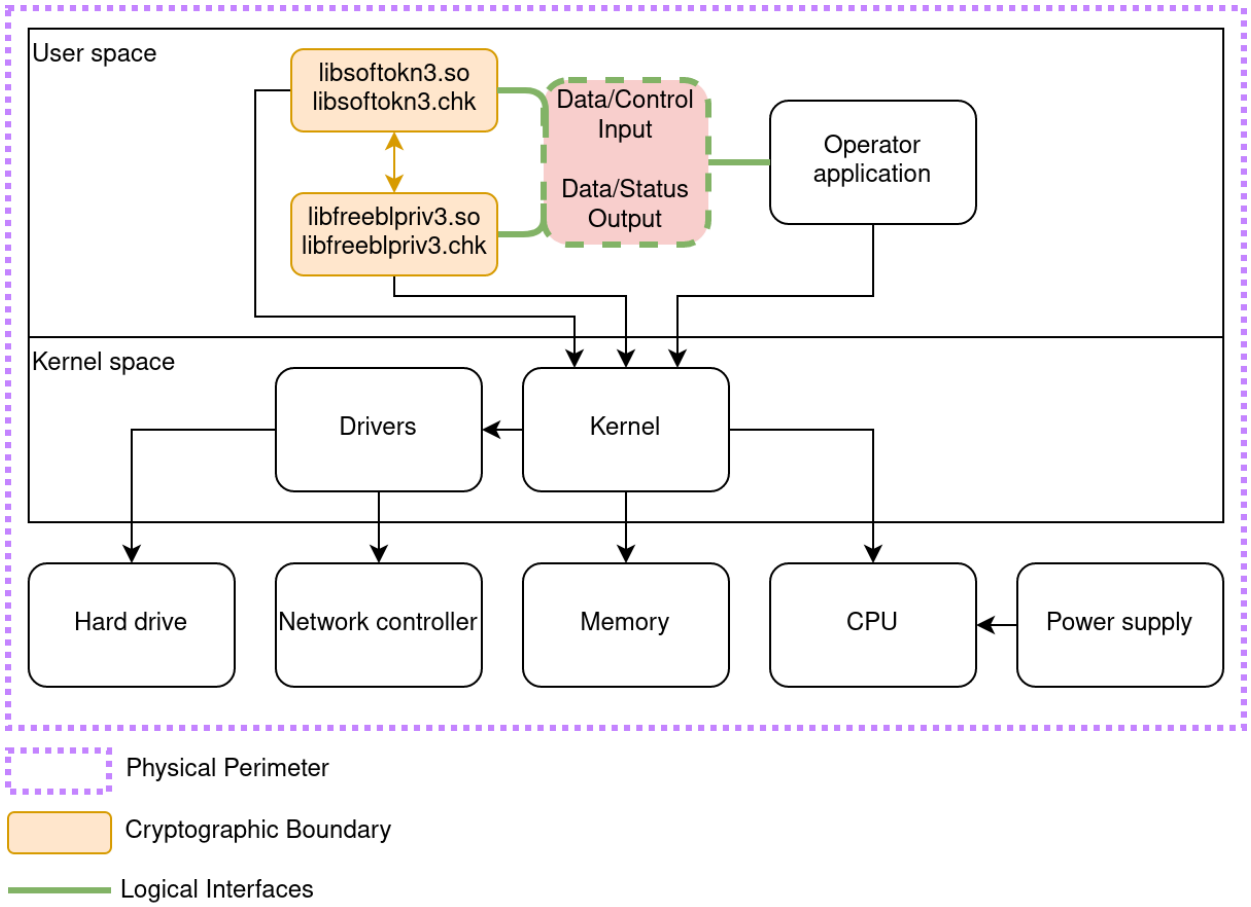

Figure 1: Block Diagram

The cryptographic boundary is represented by the components painted in orange blocks, which consists of two software components:

1. The Softoken library, which provides a PKCS#11 token API (libsoftokn3.so), and its associated integrity check value (libsoftokn3.chk).
2. The Freebl cryptographic library, which implements most cryptographic algorithms used by Softoken (libfreeblpriv3.so), and its associated integrity check value (libfreeblpriv3.chk).

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

Components in white are only included in the diagram for informational purposes. They are not included in the cryptographic boundary (and therefore not part of the module's validation). For example, the kernel is responsible for managing system calls issued by the module itself, as well as other applications using the module for cryptographic services.

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.

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):

The module has been tested on the following platforms with the corresponding module variants and configuration options with and without PAA:

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libsoftokn3.so	3.90.0-2e2ef5732e8af4d7	N/A	DSA signature verification using 2048-bit key and SHA-256
libfreeblpriv3.so	3.90.0-2e2ef5732e8af4d7	N/A	DSA signature verification using 2048-bit key and 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)
Amazon Linux 2023	EC2 c7g.metal	AWS Graviton3	Yes	N/A	3.90.0-2e2ef5732e8af4d7
Amazon Linux 2023	EC2 c6i.metal	Intel Xeon Platinum 8375C	Yes	N/A	3.90.0-2e2ef5732e8af4d7
Amazon Linux 2023	AWS Snowball	AMD EPYC 7702	Yes	N/A	3.90.0-2e2ef5732e8af4d7
Amazon Linux 2023	AWS Snowblade	Intel Xeon Gold 6314U	Yes	N/A	3.90.0-2e2ef5732e8af4d7
Amazon Linux 2023	AWS Snowcone	Intel Atom C3558	Yes	N/A	3.90.0-2e2ef5732e8af4d7

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

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

N/A for this module.

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 excluded from the requirements of the FIPS 140-3 standard.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved mode of operation	Automatically entered whenever an approved service is requested	Approved	Equivalent to the indicator of the requested service as defined in section 4.3
Non-approved mode of operation	Automatically entered whenever a non-approved service is requested	Non-Approved	Equivalent to the indicator of the requested service as defined in section 4.3

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.

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	A4576	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A4583	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A4585	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC-CS1	A4581	Direction - decrypt, encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A4578	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B
AES-CTR	A4576	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CTR	A4585	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A4576	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
AES-ECB	A4583	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A4585	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A4576	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	A4583	Direction - Decrypt, Encrypt IV Generation - External, Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-GCM	A4585	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-KW	A4577	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KW	A4582	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KW	A4584	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KWP	A4577	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KWP	A4582	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KWP	A4584	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
DSA SigVer (FIPS186-4)	A4576	L - 1024, 2048, 3072 N - 160, 224, 256 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
ECDSA KeyGen (FIPS186-4)	A4576	Curve - P-256, P-384, P-521 Secret Generation Mode - Extra Bits	FIPS 186-4
ECDSA SigGen (FIPS186-4)	A4576	Component - No Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A4576	Component - No Curve - P-256, P-384, P-521 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
Hash DRBG	A4576	Prediction Resistance - No, Yes Mode - SHA2-256	SP 800-90A Rev. 1
HMAC-SHA2-224	A4576	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A4576	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A4576	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A4576	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A4576	Domain Parameter Generation Methods - P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
KAS-FFC-SSC Sp800-56Ar3	A4576	Domain Parameter Generation Methods - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Scheme - dhEphem - KAS Role - initiator, responder	SP 800-56A Rev. 3
KDA HKDF Sp800-56Cr1	A4575	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)	A4580	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	A4579	KDF Mode - Counter, Double Pipeline Iteration, Feedback Supported Lengths - Supported Lengths: 8, 72, 128, 776, 3456, 4096	SP 800-108 Rev. 1
KDF TLS (CVL)	A4576	TLS Version - v1.0/1.1	SP 800-135 Rev. 1
PBKDF	A4576	Iteration Count - Iteration Count: 1000-10000 Increment 1 Password Length - Password Length: 8-128 Increment 1	SP 800-132
RSA KeyGen (FIPS186-4)	A4576	Key Generation Mode - B.3.3 Modulo - 2048, 3072, 4096 Primality Tests - Table C.3 Private Key Format - Standard	FIPS 186-4
RSA SigGen (FIPS186-4)	A4576	Signature Type - PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-2)	A4576	Signature Type - PKCS 1.5, PKCSPSS Modulo - 1024, 1536	FIPS 186-4
RSA SigVer (FIPS186-4)	A4576	Signature Type - PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096	FIPS 186-4
Safe Primes Key Generation	A4576	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	A4576	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A4576	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-384	A4576	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A4576	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
TLS v1.2 KDF RFC7627 (CVL)	A4576	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1

Table 5: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
Cryptographic Key Generation (Symmetric keys)	AES, HMAC, key-derivation key sizes:112-256 bits Strength:112-256 bits	Amazon Linux 2023 NSS Cryptographic Module (Generic C)	SP 800-133r2 Section 4 and 6.1

Name	Properties	Implementation	Reference
Cryptographic Key Generation (RSA)	RSA modulus sizes:2048-16384 bits Strength:112-256 bits	Amazon Linux 2023 NSS Cryptographic Module (Generic C)	SP 800-133r2 Section 4 and 5.1
Cryptographic Key Generation (Safe Primes)	Safe Primes:MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 Strength:112-200 bits	Amazon Linux 2023 NSS Cryptographic Module (Generic C)	SP 800-133r2 Section 4 and 5.2
Cryptographic Key Generation (ECDSA)	ECDSA curves:P-224, P-256, P-384, P-521 Strength:112-256 bits	Amazon Linux 2023 NSS Cryptographic Module (Generic C)	SP 800-133r2 Section 4 and 5.1

Table 6: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

Name	Caveat	Use and Function
MD5	Allowed per IG 2.4.A.	Message digest used in TLS 1.0/1.1 KDF only

Table 7: Non-Approved, Allowed Algorithms with No Security Claimed

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
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 (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512), SSL 3 PRF (MD5, SHA-1), IKEv1 PRF (AES XCBC-MAC, MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512)	Key Derivation
KBKDF, HKDF, TLS 1.0/1.1 KDF, TLS 1.2 KDF, IKEv2 KDF (< 112-bit keys)	Key Derivation
KBKDF (MD2, MD5)	Key Derivation
TLS 1.2 KDF (without extended master secret)	Key Derivation
IKEv2 KDF (MD2, MD5)	Key Derivation
PKCS#5 PBE, PKCS#12 PBE	Password-based Key Derivation
PBKDF2 (password<8 characters, salt<128 bits, iteration count<1000, or key<112 bits)	Password-based Key Derivation
J-PAKE	Shared Secret Computation

Name	Use and Function
DH (Shared secret computation; FIPS 186-type groups)	Shared Secret Computation
ECDH (Shared secret computation; P-192)	Shared Secret Computation
DSA (SigGen)	Signature Generation
RSA (SigGen primitive; PKCS#1 v1.5 or PSS with MD2, MD5)	Signature Generation
ECDSA (SigGen; P-192)	Signature Generation
RSA (encryption)	Asymmetric Encryption
DSA (parameter generation)	Parameter Generation
DH (KeyGen, FIPS 186-type groups)	Key Pair Generation
RSA (KeyGen, modulus < 2048 bits)	Key Pair Generation
ECDSA (KeyGen, P-192)	Key Pair Generation
Symmetric Key Generation (< 112 bits)	Secret Key Generation
MD2, MD5, SHA-1	Message Digest
RSA (SigVer primitive; PKCS#1 v1.5 or PSS with MD2, MD5)	Signature Verification
ECDSA (SigVer; P-192)	Signature verification
RSA (decryption)	Asymmetric Decryption
DSA (parameter verification)	Parameter verification
DSA (key pair generation)	Key Pair Generation

Table 8: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
SHA	SHA	Hash function	Reference:FIPS 180-4	SHA2-224 SHA2-256 SHA2-384 SHA2-512
AES (ECB, CBC, CBC-CS1, CTR)	BC-UnAuth	Block cipher	References:FIPS 197, SP 800-38A, SP 800-38A Addendum Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-ECB AES-ECB AES-ECB AES-CBC AES-CBC AES-CBC AES-CBC-CS1 AES-CTR AES-CTR

Name	Type	Description	Properties	Algorithms
AES (KW, KWP)	BC-Auth	Block cipher	References:FIPS 197, SP 800-38F Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-KW AES-KWP AES-KW AES-KWP AES-KW AES-KWP
AES (GCM, encryption, internal IV)	BC-Auth	Block cipher	References:FIPS 197, SP 800-38D Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-GCM AES-GCM AES-GCM
AES (CMAC)	MAC	Message authentication code based on AES	References:FIPS 197, SP 800-38B Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CMAC
HMAC	MAC	Keyed-hash message authentication code	Reference:FIPS 198-1 Keys:112-524288 bits with 112-256 bits of key strength	HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512 HMAC-SHA2-224
KBKDF	KBKDF	Key-based key derivation function	Reference:SP 800-108r1 KBKDF derived keys:112-4096 bits with 112-256 bits of key strength	KDF SP800-108
TLS 1.0/1.1 KDF, TLS 1.2 KDF	KAS-135KDF	KDF component	Reference:SP 800-135r1 TLS derived keys:112-256 bits with 112-256 bits of key strength	KDF TLS TLS v1.2 KDF RFC7627
HKDF	KAS-56CKDF	HMAC-based key derivation function	Reference:SP 800-56Cr1 HKDF derived keys:2048 bits with 112-256 bits of key strength	KDA HKDF Sp800-56Cr1
PBKDF2	PBKDF	Password-based key derivation function	Reference:SP 800-132 PBKDF derived keys:128-2048 bits with 112-256 bits of key strength	PBKDF
Hash DRBG	DRBG	Random number generation	Reference:SP 800-90Ar1 Compliance:SP800-90ARev1	Hash DRBG
KAS-FFC-SSC	KAS-SSC	Diffie-Helman shared secret computation	Reference:SP 800-56Ar3 Compliance:IG D.F Scenario 2(1) Keys:2048, 3072, 4096, 6144, 8192 bits with 112-200 bits of key strength	KAS-FFC-SSC Sp800-56Ar3
KAS-ECC-SSC	KAS-SSC	EC Diffie-Helman shared secret computation	Reference:SP 800-56Ar3 Compliance:IG D.F Scenario 2(1) Curves:P-256, P-384, P-521 with 128, 192, 256 bits of strength	KAS-ECC-SSC Sp800-56Ar3
DSA SigVer	DigSig-SigVer	DSA signature verification	Reference:FIPS 186-4 Keys:(1024, 160), (2048,	DSA SigVer (FIPS186-4)

Name	Type	Description	Properties	Algorithms
			224), (2048, 256), (3072, 256) with 80, 112, 128 bits of key strength	
RSA SigGen	DigSig-SigGen	RSA signature generation	Reference:FIPS 186-4 Keys:2048-4096 bits with 112-150 bits of key strength	RSA SigGen (FIPS186-4)
RSA SigVer	DigSig-SigVer	RSA signature verification	Reference:FIPS 186-4, FIPS 186-2 Keys:1024-4096 bits with 80-150 bits of key strength	RSA SigVer (FIPS186-2) RSA SigVer (FIPS186-4)
ECDSA SigGen	DigSig-SigGen	ECDSA signature generation	Reference:FIPS 186-4 Curves:P-256, P-384, P-521 with 128, 192, 256 bits of strength	ECDSA SigGen (FIPS186-4)
ECDSA SigVer	DigSig-SigVer	ECDSA signature verification	Reference:FIPS 186-4 Curves:P-256, P-384, P-521 with 128, 192, 256 bits of strength	ECDSA SigVer (FIPS186-4)
AES (GCM, decryption, external IV)	BC-Auth	Block cipher	References:FIPS 197, SP 800-38D Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-GCM AES-GCM AES-GCM
AES-KW, AES-KWP, AES-GCM (KTS, key wrapping)	KTS-Wrap	Key wrapping	Reference:SP 800-38F, SP 800-38D Compliance:IG D.G Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-KW AES-KWP AES-KW AES-KWP AES-KW AES-KWP AES-GCM AES-GCM AES-GCM
AES-KW, AES-KWP, AES-GCM (KTS, key unwrapping)	KTS-Wrap	Key unwrapping	Reference:SP 800-38D Compliance:IG D.G Keys :128, 192, 256 bits with 128-256 bits of keys strength	AES-GCM AES-GCM AES-GCM AES-KW AES-KWP AES-KW AES-KWP AES-KW AES-KWP
Hash DRBG (symmetric key generation)	CKG	Symmetric Key Generation	Reference:SP 800-133r2 section 6.1 Keys:112-256 bits with 112-256 bits of key strength	Hash DRBG
RSA KeyGen (key pair generation)	AsymKeyPair-KeyGen	Key pair generation with RSA	Reference:FIPS 186-4 Keys:2048-4096 bits with 112-150 bits of key strength	RSA KeyGen (FIPS186-4)
ECDSA KeyGen (key pair generation)	AsymKeyPair-KeyGen	Key pair generation with ECDSA	Reference:FIPS 186-4 Curves:P-256, P-384, P-	ECDSA KeyGen (FIPS186-4)

Name	Type	Description	Properties	Algorithms
			521 with 128, 192, 256 bits of strength	
Safe Primes Key Generation (key pair generation)	AsymKeyPair-KeyGen	Key pair generation with Safe Primes	Reference:SP800-56Ar3 Keys:2048, 3072, 4096, 6144, 8192 bits with 112-200 bits of key strength	Safe Primes Key Generation
IKEv2 KDF	KAS-135KDF	KDF component	Reference:SP 800-135r1 IKEv2 derived keys:112-256 bits with 112-256 bits of key strength	KDF IKEv2

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. These IVs are always 96 bits and generated using the approved DRBG internal to the module's boundary, compliant to Scenario 2 of FIPS 140-3 IG C.H.

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 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 password or passphrase shall be at least 8 characters, and shall consist of lowercase, uppercase, and numeric 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.

The calling application must observe the rest of the requirements and recommendations specified in [SP800-132].

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 the Approved Services table) 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.

2.8 RBG and Entropy

Cert Number	Vendor Name
E124	Amazon

Table 10: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Amazon Userspace CPU Time Jitter RNG Entropy Source	Non-Physical	Amazon Linux 2023 on EC2 c7g.metal on AWS Graviton3; Amazon Linux 2023 on EC2 c6i.metal on Intel Xeon Platinum 8375C; Amazon Linux 2023 on AWS Snowball on AMD EPYC 7702; Amazon Linux	256 bits	256 bits	SHA3-256 (A4551); HMAC-SHA-512 DRBG (A4551)

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
		2023 on AWS Snowblade on Intel Xeon Gold 6314U; Amazon Linux 2023 on AWS Snowcone on Intel Atom C3558			

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 Entropy Sources table. It does not employ prediction resistance. The DRBG is seeded with 440 bits of entropy and reseeded with 440 bits. 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 directly obtained as output from the SP 800-90Ar1 approved DRBG (without XOR), 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 186-4. The method described in Appendix B.3.3 of FIPS 186-4 ("Probable Primes") is used.
- ECDSA key pair generation: compliant with SP 800-133r2, Section 5.1, which maps to FIPS 186-4. The method described in Appendix B.4.1 of FIPS 186-4 ("Extra Random Bits") is used. Note that this generation method is also used to generated ECDH key pairs. This method is compliant with SP 800-133r2, Section 5.2.
- Intermediate key generation values are not output from the module and are explicitly zeroized after processing the service.

Additionally, the module implements the following key derivation methods:

- KBKDF: compliant with SP 800-108r1. This implementation can be used to derive secret keys from a pre-existing key-derivation-key.
- HKDF: compliant with SP 800-56Cr2. This implementation shall only be used to derive secret keys in the context of an SP 800-56Ar3 key agreement scheme.
- TLS 1.0/1.1 KDF, TLS 1.2 KDF, IKEv2 KDF: compliant with SP 800-135r1. These implementations shall only be used to derive secret keys in the context of the TLS 1.0/1.1, TLS 1.2, or IPSec 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.

2.10 Key Establishment

The module implements the SSP establishment methods as specified in the *Security Function Implementations* table.

2.11 Industry Protocols

GCM with internal IV generation in the approved mode is compliant with versions 1.2 and 1.3 of the TLS protocol (RFC 5288 and 8446) and shall only be used in conjunction with the TLS protocol. Additionally, the module implements the TLS 1.0/1.1 and 1.2 key derivation functions for use in the TLS protocol.

The module implements the IKEv2 key derivation function for use in the IPSec protocol (RFC 5996).

For Diffie-Hellman, the module supports the use of the following safe primes:

- 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)

No other parts of the TLS or IPSec protocols, other than the KDFs mentioned above, have been tested by the CAVP and CMVP.

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 12: Ports and Interfaces

The module does not have a control output interface.

4 Roles, Services, and Authentication

4.1 Authentication Methods

N/A for this module.

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.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Message Digest	Compute a message digest	CKS_NSS_FIPS_OK	Message	Digest value	SHA	Crypto Officer
Encryption	Encrypt a plaintext	CKS_NSS_FIPS_OK	AES key, plaintext	Ciphertext	AES (ECB, CBC, CBC-CS1, CTR)	Crypto Officer - AES key: W,E
Decryption	Decrypt a ciphertext	CKS_NSS_FIPS_OK	AES key, ciphertext	Plaintext	AES (ECB, CBC, CBC-CS1, CTR)	Crypto Officer - AES key: W,E
Authenticated Encryption	Encrypt a plaintext	CKS_NSS_FIPS_OK	Inputs of GCM: AES key, IV, plaintext; Inputs of KW, KWP: AES key, plaintext	Outputs of GCM: Ciphertext, MAC tag; Outputs of KW, KWP: Ciphertext	AES (KW, KWP) AES (GCM, encryption, internal IV)	Crypto Officer - AES key: W,E
Authenticated Decryption	Decrypt a ciphertext	CKS_NSS_FIPS_OK	Inputs of GCM: AES key, IV, MAC tag, ciphertext; Inputs of KW, KWP: AES key, ciphertext	Outputs of GCM: Plaintext or fail; Outputs of KW, KWP: Plaintext	AES (KW, KWP) AES (GCM, decryption, external IV)	Crypto Officer - AES key: W,E
Key Wrapping	Wrap a CSP	CKS_NSS_FIPS_OK	Inputs of KW, KWP: AES key, any CSP (except for password); Inputs of GCM: AES key, IV, any CSP (except for password)	Outputs of KW, KWP: Wrapped CSP; Outputs of GCM: Wrapped CSP, MAC tag	AES-KW, AES-KWP, AES-GCM (KTS, key wrapping)	Crypto Officer - AES key: W,E
Key Unwrapping	Unwrap a CSP	CKS_NSS_FIPS_OK	Inputs of GCM: AES key, IV, MAC tag, wrapped CSP; Inputs of KW, KWP: AES key, wrapped CSP	Outputs of GCM: Unwrapped CSP or fail; Outputs of KW, KWP: Unwrapped CSP	AES-KW, AES-KWP, AES-GCM (KTS, key unwrapping)	Crypto Officer - AES key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Message Authentication	Compute a MAC tag	CKS_NSS_FIPS_OK	Inputs of AES-CMAC: AES key, message. Inputs of HMAC: HMAC key, message	MAC tag	AES (CMAC) HMAC	Crypto Officer - AES key: W,E - HMAC key: W,E
Password-based Key Derivation	Derive a key from a password	CKS_NSS_FIPS_OK	Password, salt, iteration count	PBKDF derived key	PBKDF2	Crypto Officer - Password: W,E - PBKDF derived key: G
Random Number Generation	Generate random bytes	CKR_OK	Output length	Random bytes	Hash DRBG	Crypto Officer - Entropy input: W,E - DRBG seed: G,E - Internal state (V, C): G,W,E
Shared Secret Computation	Compute a shared secret	CKS_NSS_FIPS_OK	Inputs of KAS-FFC-SSC: DH private key (owner), DH public key (peer). Inputs of KAS-ECC-SSC: EC private key (owner), EC public key (peer)	Shared secret	KAS-FFC-SSC KAS-ECC-SSC	Crypto Officer - EC private key: W,E - EC public key: W,E - Shared secret: G - DH private key: W,E - DH public key: W,E
Signature Generation	Generate a signature	CKS_NSS_FIPS_OK	Inputs of RSA SigGen: RSA private key, message. Inputs of ECDSA SigGen: EC private key, message	Signature	RSA SigGen ECDSA SigGen	Crypto Officer - RSA private key: W,E - EC private key: W,E
Signature Verification	Verify a signature	Indicator of DSA SigVer: CKR_OK. Indicator of RSA SigVer: CKS_NSS_FIPS_OK. Indicator of ECDSA SigVer: CKS_NSS_FIPS_OK	Inputs of DSA SigVer: DSA public key, message, signature. Inputs of RSA SigVer: RSA public key, message, signature. Inputs of ECDSA SigVer: EC public key, message, signature.	Pass/fail	DSA SigVer RSA SigVer ECDSA SigVer	Crypto Officer - DSA public key: W,E - RSA public key: W,E - EC public key: W,E
Secret Key Generation	Generate a symmetric key	CKS_NSS_FIPS_OK	Key size	AES key, HMAC key, or Key-derivation key	Hash DRBG (symmetric key generation)	Crypto Officer - AES key: G - HMAC key: G - Key-derivation key: G
Key Pair Generation	Generate a key pair	CKS_NSS_FIPS_OK	Safe Primes Key Generation: Group; RSA KeyGen: Modulus	Safe Primes Key Generation: DH public & private key; RSA KeyGen:	RSA KeyGen (key pair generation) ECDSA	Crypto Officer - DH private key: G - DH public

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
			size; ECDSA KeyGen: Curve	RSA public & private key; ECDSA KeyGen: EC public & private key	KeyGen (key pair generation) Safe Primes Key Generation (key pair generation)	key: G - EC private key: G - EC public key: G - RSA private key: G - RSA public key: G - Intermediate key generation value: G,E,Z
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	SHA AES (ECB, CBC, CBC-CS1, CTR) AES (GCM, encryption, internal IV) AES (CMAC) HMAC KBKDF TLS 1.0/1.1 KDF, TLS 1.2 KDF HKDF PBKDF2 Hash DRBG KAS-FFC-SSC KAS-ECC-SSC DSA SigVer RSA SigGen RSA SigVer ECDSA SigGen ECDSA SigVer AES (GCM, decryption, external IV) RSA KeyGen (key pair generation) ECDSA KeyGen (key pair generation) Safe Primes Key Generation (key pair	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					generation) IKEv2 KDF	
Zeroization	Zeroize any SSP	None	Any SSP	None	None	Crypto Officer - AES key: Z - HMAC key: Z - Key-derivation key: Z - Shared secret: Z - Password: Z - PBKDF 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 - DSA public key: Z - RSA private key: Z - RSA public key: Z - Intermediate key generation value: Z - HKDF Derived key: Z - KBKDF derived key: Z - TLS derived key: Z - IKEv2 derived key: Z
Key-based Key Derivation	Derive a key from a key-derivation key	CKS_NSS_FIPS_OK	Key-derivation key	KBKDF derived key	KBKDF	Crypto Officer - Key-derivation key: W,E - KBKDF derived key: G
HMAC-based Key Derivation	Derive a key from a shared secret	CKS_NSS_FIPS_OK	Shared secret	HKDF derived key	HKDF	Crypto Officer - Shared secret: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- HKDF Derived key: G
IKEv2 Key Derivation	Derived a key from a shared secret	CKS_NSS_FIPS_OK	Shared secret	IKEv2 derived key	IKEv2 KDF	Crypto Officer - Shared secret: W,E - IKEv2 derived key: G
TLS Key Derivation	Derive a key from a shared secret	CKS_NSS_FIPS_OK	Shared secret	TLS derived key	TLS 1.0/1.1 KDF, TLS 1.2 KDF	Crypto Officer - Shared secret: W,E - TLS derived key: G

Table 14: Approved Services

The following convention is used to specify access rights to SSPs:

- **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 Derivation service. 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 Derivation service. 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 Random Number Generation service indicator, which must be used for the Random Number Generation 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. The DSA Signature Verification indicator, which must be used for the DSA Signature Verification service. It can be accessed by invoking the C_VerifyInit function with any CKM_DSA_* mechanism parameter. If this function 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 (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512), SSL 3 PRF (MD5, SHA-1), IKEv1 PRF (AES XCBC-MAC, MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512) KBKDF, HKDF, TLS 1.0/1.1 KDF, TLS 1.2 KDF, IKEv2 KDF (< 112-bit keys) KBKDF (MD2, MD5) TLS 1.2 KDF (without extended master secret) IKEv2 KDF (MD2, MD5)	CO
Password-Based Key Derivation	Derive a key from a password	PKCS#5 PBE, PKCS#12 PBE PBKDF2 (password<8 characters, salt<128 bits, iteration count<1000, or key<112 bits)	CO
Shared Secret Computation	Compute a shared secret	J-PAKE DH (Shared secret computation; FIPS 186-type groups) ECDH (Shared secret computation; P-192)	CO
Signature Generation	Generate a signature	DSA (SigGen) RSA (SigGen primitive; PKCS#1 v1.5 or PSS with MD2, MD5) ECDSA (SigGen; P-192)	CO
Asymmetric Encryption	Encrypt a plaintext	RSA (encryption)	CO
Asymmetric Decryption	Decrypt a plaintext	RSA (decryption)	CO
Parameter Generation	Generate domain parameters	DSA (parameter generation)	CO
Parameter Verification	Verify domain parameters	DSA (parameter verification)	CO
Key Pair Generation	Generate a key pair	DH (KeyGen, FIPS 186-type groups) RSA (KeyGen, modulus < 2048 bits) ECDSA (KeyGen, P-192) DSA (key pair generation)	CO
Secret Key Generation	Generate a symmetric key	Symmetric Key Generation (< 112 bits)	CO
Signature Verification	Verify a signature	RSA (SigVer primitive; PKCS#1 v1.5 or PSS with MD2, MD5) ECDSA (SigVer; P-192)	CO

Table 15: Non-Approved Services

The indicator value for the non-approved services specified in the Non-Approved Services table is CKS_NSS_FIPS_NOT_OK (0).

4.5 External Software/Firmware Loaded

The module does not load external software or firmware.

5 Software/Firmware Security

5.1 Integrity Techniques

The integrity of the module is verified by performing DSA signature verification with a 2048-bit key and SHA-256. Each software component of the module has an associated integrity check value, which contains the DSA signature of the shared library.

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 provided by the Linux kernel, and only the module has control over these SSPs.

6.2 Configuration Settings and Restrictions

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

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.

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.	Dynamic

Table 16: Storage Areas

The module does not perform persistent storage of SSPs. SSPs are provided to the module by the calling process.

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	AES-KW, AES-KWP, AES-GCM (KTS, key unwrapping)
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	AES-KW, AES-KWP, AES-GCM (KTS, key wrapping)

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 (encrypted) 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. The module only supports SSP entry and output to and from the calling application running on the same operational environment.

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

Zeroization Method	Description	Rationale	Operator Initiation
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	Hash DRBG (symmetric key generation)		AES (ECB, CBC, CBC-CS1, CTR) AES (KW, KWP) AES (GCM, encryption, internal IV) AES (CMAC) AES (GCM, decryption, external IV) AES-KW, AES-KWP, AES-GCM (KTS, key wrapping) AES-KW, AES-KWP, AES-GCM (KTS, key unwrapping)
HMAC key	HMAC key used for computing MAC tags	112-524288 bits - 112-256 bits	Symmetric key - CSP	Hash DRBG (symmetric key generation)		HMAC
Key-derivation key	Symmetric key used to derive symmetric keys	112-4096 bits - 112-256 bits	Symmetric key - CSP	Hash DRBG (symmetric key generation)		KBKDF
Shared secret	Shared secret generated by (EC) Diffie-Hellman	256-8192 bits - 112-256 bits	Shared secret - CSP		KAS-FFC-SSC KAS-ECC-SSC	TLS 1.0/1.1 KDF, TLS 1.2 KDF HKDF IKEv2 KDF
Password	Password used to derive symmetric keys	8-128 characters - N/A	Password - CSP			PBKDF2
HKDF Derived key	Symmetric key derived from a shared secret	2048 bits - 112-256 bits	Symmetric key - CSP	HKDF		
Entropy input	Entropy input used to seed the DRBG	440 bits - 440 bits	Entropy input - CSP			Hash DRBG
DRBG seed	DRBG seed derived from entropy input	440 bits - 256 bits	Seed - CSP	Hash DRBG		Hash DRBG
Internal state (V, C)	Internal state of the Hash_DRBG instance	880 bits - 256 bits	Internal state - CSP	Hash DRBG		Hash DRBG

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DH private key	Private key used for Diffie- Hellman	2048-8192 bits - 112-200 bits	Private key - CSP	Safe Primes Key Generation (key pair generation)		KAS-FFC-SSC
DH public key	Public key used for Diffie- Hellman	2048-8192 bits - 112-200 bits	Public key - PSP	Safe Primes Key Generation (key pair generation)		KAS-FFC-SSC
EC private key	Private key used for ECDH and ECDSA	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP	ECDSA KeyGen (key pair generation)		KAS-ECC-SSC ECDSA SigGen
EC public key	Public key used for ECDH and ECDSA	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP	ECDSA KeyGen (key pair generation)		KAS-ECC-SSC ECDSA SigVer
DSA public key	Public key used for DSA signature verification	(1024, 160), (2048, 224), (2048, 256), (3072, 256) - 80, 112, 128 bits	Public key - PSP			DSA SigVer
RSA private key	Private key used for RSA signature generation	2048-4096 bits - 112-150 bits	Private key - CSP	RSA KeyGen (key pair generation)		RSA SigGen
RSA public key	Public key used for RSA signature verification	Signature verification: 1024-4096 bits; Key pair generation: 2048-4096 bits - Signature verification: 80- 150 bits; Key pair generation: 112- 150 bits	Public key - PSP	RSA KeyGen (key pair generation)		RSA SigVer
Intermediate key generation value	Temporary value generated during symmetric key and key pair generation services	112-8192 bits - 112-256 bits	Intermediate value - CSP	RSA KeyGen (key pair generation) ECDSA KeyGen (key pair generation) Safe Primes Key Generation (key pair generation)		RSA KeyGen (key pair generation) ECDSA KeyGen (key pair generation) Safe Primes Key Generation (key pair generation)
KBKDF derived key	Symmetric key derived from a key-derivation key	112-4096 bits - 112-256 bits	Symmetric key - CSP	KBKDF		
TLS derived key	Symmetric key derived from a shared secret	112-256 bits - 112-256 bits	Symmetric key - CSP	TLS 1.0/1.1 KDF, TLS 1.2 KDF		
IKEv2 derived key	Symmetric key derived from a shared secret	112-256 bits - 112-256 bits	Symmetric key - CSP	IKEv2 KDF		
PBKDF derived key	Symmetric key derived from a password	128-2048 bits - 112-256 bits	Symmetric key - CSP	PBKDF2		

Table 19: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES key	API input parameters (encrypted) API output	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
	parameters (encrypted)				
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)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	KBKDF derived key:Used to derive
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:Established using DH public key:Established using EC private key:Established using EC public key:Established using TLS derived key:Used to derive IKEv2 derived key:Used to derive HKDF Derived key:Used to derive
Password	API input parameters (plaintext)	RAM:Plaintext	For the duration of the service	Destroy Object Remove power from the module	PBKDF derived key:Used to derive
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
Entropy input		RAM:Plaintext	From generation until DRBG seed is created	Automatic Remove power from the module	DRBG seed:Used to derive
DRBG seed		RAM:Plaintext	While the DRBG is instantiated	Automatic Remove power from the module	Entropy input:Derived From Internal state (V, C):Used to generate
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)	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

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
	API output parameters (encrypted)				
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
DSA public key	API input parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	
RSA private key	API input parameters (encrypted) API output parameters (encrypted)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	RSA public key:Paired With Intermediate key 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:Created during generation of DH public key:Created during generation of EC private key:Created during generation of EC public key:Created during generation of RSA private key:Created during generation of RSA public key:Created during generation of
KBKDF derived key	API output parameters (encrypted)	RAM:Plaintext	For the duration of the service	Destroy Object Remove power from the module	Key-derivation key:Derived From
TLS derived key	API output parameters (encrypted)	RAM:Plaintext	For the duration of the service	Destroy Object Remove power from the module	Shared secret:Derived From
IKEv2 derived key	API output parameters (encrypted)	RAM:Plaintext	For the duration of the service	Destroy Object Remove power from the module	Shared secret:Derived From
PBKDF derived key	API output parameters (encrypted)	RAM:Plaintext	For the duration of the service	Destroy Object Remove power from the module	Password:Derived From

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.

The ECDSA, and RSA algorithms as implemented by the module conform to FIPS 186-4, which has been superseded by FIPS 186-5. The transition started on July 25, 2023, and ended on February 4, 2024. FIPS 186-4 was withdrawn on February 3, 2024.

10 Self-Tests

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 DSA CAST in the Softoken library, followed by the the pre-operational integrity test on the libsoftokn3.so file using its associated check value. 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 fail 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
DSA SigVer (FIPS186-4) (A4576)	2048-bit key with SHA-256	Signature Verification	SW/FW Integrity	Module becomes operational	Integrity test for libfreeblpriv3.so
DSA SigVer (FIPS186-4) (A4576)	2048-bit key with SHA-256	Signature Verification	SW/FW Integrity	Module becomes operational	Integrity test for libfsoftokn3.so

Table 21: Pre-Operational Self-Tests

Each software component of the module has an associated integrity check value, which contains the DSA signature of the shared library. The software integrity tests ensure that the module is not corrupted. The DSA and SHA-256 algorithms go through their respective CASTs before the software integrity tests are performed.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA-1 (A4576)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Freebl initialization
SHA2-224 (A4576)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Freebl initialization
SHA2-256 (A4576)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Freebl initialization
SHA2-384 (A4576)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Freebl initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA2-512 (A4576)	512-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Freebl initialization
AES-ECB (A4576)	128, 192, 256-bit key, 128-bit plaintext	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	Freebl initialization
AES-ECB (A4583)	128, 192, 256-bit key, 128-bit plaintext	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	Freebl initialization
AES-ECB (A4585)	128, 192, 256-bit key, 128-bit plaintext	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	Freebl initialization
AES-CBC (A4576)	128, 192, 256-bit key, 128-bit plaintext	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	Freebl initialization
AES-CBC (A4581)	128, 192, 256-bit key, 128-bit plaintext	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	Freebl initialization
AES-CBC (A4583)	128, 192, 256-bit key, 128-bit plaintext	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	Freebl initialization
AES-CBC (A4585)	128, 192, 256-bit key, 128-bit plaintext	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	Freebl initialization
AES-GCM (A4576)	128, 192, 256-bit key, 128-bit IV, 128-bit plaintext, 112-bit additional data	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	Freebl initialization
AES-GCM (A4583)	128, 192, 256-bit key, 128-bit IV, 128-bit plaintext, 112-bit additional data	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	Freebl initialization
AES-GCM (A4585)	128, 192, 256-bit key, 128-bit IV, 128-bit plaintext, 112-bit additional data	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	Freebl initialization
AES-CMAC (A4578)	128, 192, 256-bit key, 128-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Freebl initialization
HMAC-SHA-1 (A4576)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Freebl initialization
HMAC-SHA2-224 (A4576)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Freebl initialization
HMAC-SHA2-256 (A4576)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Freebl initialization
HMAC-SHA2-384 (A4576)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Freebl initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-512 (A4576)	288-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Freebl initialization
KDF SP800-108 (A4579)	Counter mode HMAC SHA-256 576-bit input key	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Softoken initialization
KDA HKDF Sp800-56Cr1 (A4575)	SHA-256, 512-bit input secret	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Softoken initialization
KDF TLS (A4576)	288-bit input secret	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Freebl initialization
TLS v1.2 KDF RFC7627 (A4576)	SHA-256, 288-bit input secret	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Freebl initialization
KDF IKEv2 (A4580)	SHA-1, SHA-256, SHA-384, SHA-512; 80, 128, 144-bit input secret	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Softoken initialization
PBKDF (A4576)	SHA-256, 14-character password, 128-bit salt, Iteration count: 5	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Softoken initialization
Hash DRBG (A4576)	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 Section 11.3)	Freebl initialization
KAS-FFC-SSC Sp800-56Ar3 (A4576)	2048-bit key	KAT	CAST	Module becomes operational and services are available for use	Shared Secret Computation	Freebl initialization
KAS-ECC-SSC Sp800-56Ar3 (A4576)	P-256	KAT	CAST	Module becomes operational and services are available for use	Shared Secret Computation	Freebl initialization
DSA SigVer (FIPS186-4) (A4576)	1024-bit key	KAT	CAST	Module becomes operational and services are available for use	Signature verification	Freebl initialization
RSA SigGen (FIPS186-4) (A4576)	PKCS#1 v1.5 with SHA-256, SHA-384, SHA-512, 2048-bit key	KAT	CAST	Module becomes operational and services are available for use	Signature Generation	Softoken initialization
RSA SigVer (FIPS186-4) (A4576)	PKCS#1 v1.5 with SHA-256, SHA-384, SHA-512, 2048-bit key	KAT	CAST	Module becomes operational and services are available for use	Signature Verification	Softoken initialization
ECDSA SigGen (FIPS186-4) (A4576)	SHA-256, P-256	KAT	CAST	Module becomes operational and services are available for use	Signature Generation	Freebl initialization
ECDSA SigVer (FIPS186-4) (A4576)	SHA-256, P-256	KAT	CAST	Module becomes operational and services are available for use	Signature Verification	Freebl initialization
Safe Primes Key Generation (A4576)	N/A	PCT	PCT	Successful key pair generation	SP 800-56Ar3 section 5.6.2.1.4	Key pair generation

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
ECDSA KeyGen (FIPS186-4) (A4576)	N/A (KeyGen PCT for ECDH)	PCT	PCT	Successful key pair generation	SP 800-56Ar3 section 5.6.2.1.4	Key pair generation
RSA KeyGen (FIPS186-4) (A4576)	PKCS#1 v1.5 with SHA-256	PCT	PCT	Successful key pair generation	Signature Generation & Signature Verification	Key pair generation
ECDSA KeyGen (FIPS186-4) (A4576)	SHA-256	PCT	PCT	Successful key pair generation	Signature Generation & Signature Verification	Key pair generation

Table 22: Conditional Self-Tests

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

Upon generation of a key pair, the module will perform a pair-wise consistency test (PCT), as shown in the Conditional Self-Tests table, 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
DSA SigVer (FIPS186-4) (A4576)	Signature Verification	SW/FW Integrity	On demand	Manually
DSA SigVer (FIPS186-4) (A4576)	Signature Verification	SW/FW Integrity	On demand	Manually

Table 23: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
SHA-1 (A4576)	KAT	CAST	On demand	Manually
SHA2-224 (A4576)	KAT	CAST	On demand	Manually
SHA2-256 (A4576)	KAT	CAST	On demand	Manually
SHA2-384 (A4576)	KAT	CAST	On demand	Manually
SHA2-512 (A4576)	KAT	CAST	On demand	Manually
AES-ECB (A4576)	KAT	CAST	On demand	Manually
AES-ECB (A4583)	KAT	CAST	On demand	Manually
AES-ECB (A4585)	KAT	CAST	On demand	Manually
AES-CBC (A4576)	KAT	CAST	On demand	Manually
AES-CBC (A4581)	KAT	CAST	On demand	Manually
AES-CBC (A4583)	KAT	CAST	On demand	Manually
AES-CBC (A4585)	KAT	CAST	On demand	Manually
AES-GCM (A4576)	KAT	CAST	On demand	Manually
AES-GCM (A4583)	KAT	CAST	On demand	Manually
AES-GCM (A4585)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CMAC (A4578)	KAT	CAST	On demand	Manually
HMAC-SHA-1 (A4576)	KAT	CAST	On demand	Manually
HMAC-SHA2-224 (A4576)	KAT	CAST	On demand	Manually
HMAC-SHA2-256 (A4576)	KAT	CAST	On demand	Manually
HMAC-SHA2-384 (A4576)	KAT	CAST	On demand	Manually
HMAC-SHA2-512 (A4576)	KAT	CAST	On demand	Manually
KDF SP800-108 (A4579)	KAT	CAST	On demand	Manually
KDA HKDF Sp800-56Cr1 (A4575)	KAT	CAST	On demand	Manually
KDF TLS (A4576)	KAT	CAST	On demand	Manually
TLS v1.2 KDF RFC7627 (A4576)	KAT	CAST	On demand	Manually
KDF IKEv2 (A4580)	KAT	CAST	On demand	Manually
PBKDF (A4576)	KAT	CAST	On demand	Manually
Hash DRBG (A4576)	KAT	CAST	On demand	Manually
KAS-FFC-SSC Sp800-56Ar3 (A4576)	KAT	CAST	On demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A4576)	KAT	CAST	On demand	Manually
DSA SigVer (FIPS186-4) (A4576)	KAT	CAST	On demand	Manually
RSA SigGen (FIPS186-4) (A4576)	KAT	CAST	On demand	Manually
RSA SigVer (FIPS186-4) (A4576)	KAT	CAST	On demand	Manually
ECDSA SigGen (FIPS186-4) (A4576)	KAT	CAST	On demand	Manually
ECDSA SigVer (FIPS186-4) (A4576)	KAT	CAST	On demand	Manually
Safe Primes Key Generation (A4576)	PCT	PCT	On demand	Manually
ECDSA KeyGen (FIPS186-4) (A4576)	PCT	PCT	On demand	Manually
RSA KeyGen (FIPS186-4) (A4576)	PCT	PCT	On demand	Manually
ECDSA KeyGen (FIPS186-4) (A4576)	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 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 as a part of the Amazon Linux 2023 package in the form of the nss-softokn-3.90.0-3.amzn2023.0.3 and nss-softokn-freebl-3.90.0-3.amzn2023.0.3 RPM packages. The Netscape Portable Runtime (NSPR) package nspr-4.35.0-5.amzn2023.0.3 is a prerequisite for the module.

Before the RPM packages are installed, the Amazon Linux 2023 system must operate in the FIPS validated configuration. To achieve this, the Crypto Officer must execute the fips-mode-setup --enable command, then, restart the system. More information can be found at the vendor documentation.

The Crypto Officer must verify the Amazon Linux 2023 system operates in the FIPS validated configuration by executing the fips-mode-setup --check command, which should output "FIPS mode is enabled."

11.2 Administrator Guidance

After installation of the RPM packages, the Crypto Officer must execute the "Show 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

Amazon Linux 2023 nss 3.90.0-2e2ef5732e8af4d7

Alternatively, the /usr/lib64/nss/unsupported-tools/validation tool is provided as a convenience by the nss-tools-3.90.0-3.amzn2023.0.3 RPM package. This tool performs the same steps, and outputs the FIPS module identifier as above.

The cryptographic boundary consists only of the Softoken and Freebl libraries along with their associated integrity check values as listed in the Tested Module Identification table. 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.

11.3 Non-Administrator Guidance

There is no non-administrator guidance.

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-3.90.0-3.amzn2023.0.3 and nss-softokn-freebl-3.90.0-3.amzn2023.0.3 RPM packages can be uninstalled from the Amazon Linux 2023 systems.

12 Mitigation of Other Attacks

Attack	Mitigation Mechanism	Specific Limit
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.	None
Cache-timing attacks on the modular exponentiation operation used in RSA	**Cache invariant modular exponentiation** This is a variant of a modular exponentiation implementation that Colin Percival showed to defend against cache-timing attacks	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.	None

12.1 Attack List

The following attacks are mitigated:

- Timing attacks on RSA
- Cache-timing attacks on the modular exponentiation operation used in RSA
- Arithmetic errors in RSA signatures

Appendix A. Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
CAVP	Cryptographic Algorithm Validation Program
CAST	Cryptographic Algorithm Self-Test
CBC	Cipher Block Chaining
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
KAT	Known Answer Test
KW	AES Key Wrap
KWP	AES Key Wrap with Padding
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
OS	Operating System
PAA	Processor Algorithm Acceleration
PCT	Pair-Wise Consistency Test
PSP	Public Security Parameter
PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm

Appendix B. References

FIPS140-3	**FIPS PUB 140-3 - Security Requirements for Cryptographic Modules** March 2019 https://doi.org/10.6028/NIST.FIPS.140-3
FIPS140-3_IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program** January 2024 https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements
FIPS180-4	**Secure Hash Standard (SHS)** March 2012 http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS186-4	**Digital Signature Standard (DSS)** July 2013 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
FIPS186-5	**Digital Signature Standard (DSS)** February 2023 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf
FIPS197	**Advanced Encryption Standard** November 2001 http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS198-1	**The Keyed Hash Message Authentication Code (HMAC)** July 2008 http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
PKCS#1	**Public Key Cryptography Standards (PKCS) #1: RSA Cryptography** Specifications Version 2.1 February 2003 http://www.ietf.org/rfc/rfc3447.txt
SP800-38A	**Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
SP800-38B	**NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf
SP800-38D	**NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf

SP800-38F	**NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP800-56Ar3	**NIST Special Publication 800-56A Revision 3 - Recommendation for Pair Wise Key Establishment Schemes Using Discrete Logarithm Cryptography** April 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf
SP800-56Cr2	**NIST Special Publication 800-56C Revision 2 - Recommendation for Key-Derivation Methods in Key-Establishment Schemes** August 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf
SP800-90Ar1	**NIST Special Publication 800-90A - Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP800-90B	**NIST Special Publication 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
SP800-108r1	**NIST Special Publication 800-108 Revision 1 - Transitions: Recommendation for Key Derivation Using Pseudorandom Functions** August 2022 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-108r1.pdf
SP800-131Ar1	**NIST Special Publication 800-131A Revision 1 - Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths** November 2015 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf
SP800-133r2	**NIST Special Publication 800-133rev2 - Recommendation for Cryptographic Key Generation** June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf
SP800-135r1	**NIST Special Publication 800-135 Revision 1 - Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf