ORACLE
Linux

FIPS 140-3 Non-Proprietary Security Policy

Oracle Corporation

Oracle Linux 9 NSS Cryptographic Module

Software Version: 4.35.0-381552536e763d0c

Prepared by:

atsec information security corporation

4516 Seton Center Pkwy, Suite 250

Austin, TX 78759

www.atsec.com

Title: Oracle Linux 9 NSS Cryptographic Module Security Policy

Date: August 28th, 2025

Contributing Authors:

Oracle Linux Engineering

Security Evaluations – Global Product Security

atsec information security

Oracle Corporation

World Headquarters

2300 Oracle Way

Austin, TX 78741

U.S.A.

Worldwide Inquiries:

Phone: +1.650.506.7000

Fax: +1.650.506.7200

www.oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together

ORACLE

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 software version 4.35.0-381552536e763d0c of the Oracle 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.

1.1.1. How This Security Policy was Prepared

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

1.2 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 Oracle 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. The software components of the cryptographic module are listed in the Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets) table.

Module Type: Software

Module Embodiment: Multi-Chip Standalone

Cryptographic Boundary:

Figure 1 shows the cryptographic boundary of the module, its interfaces with the operational environment and the flow of information between the module and operator (depicted through the arrows).

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.

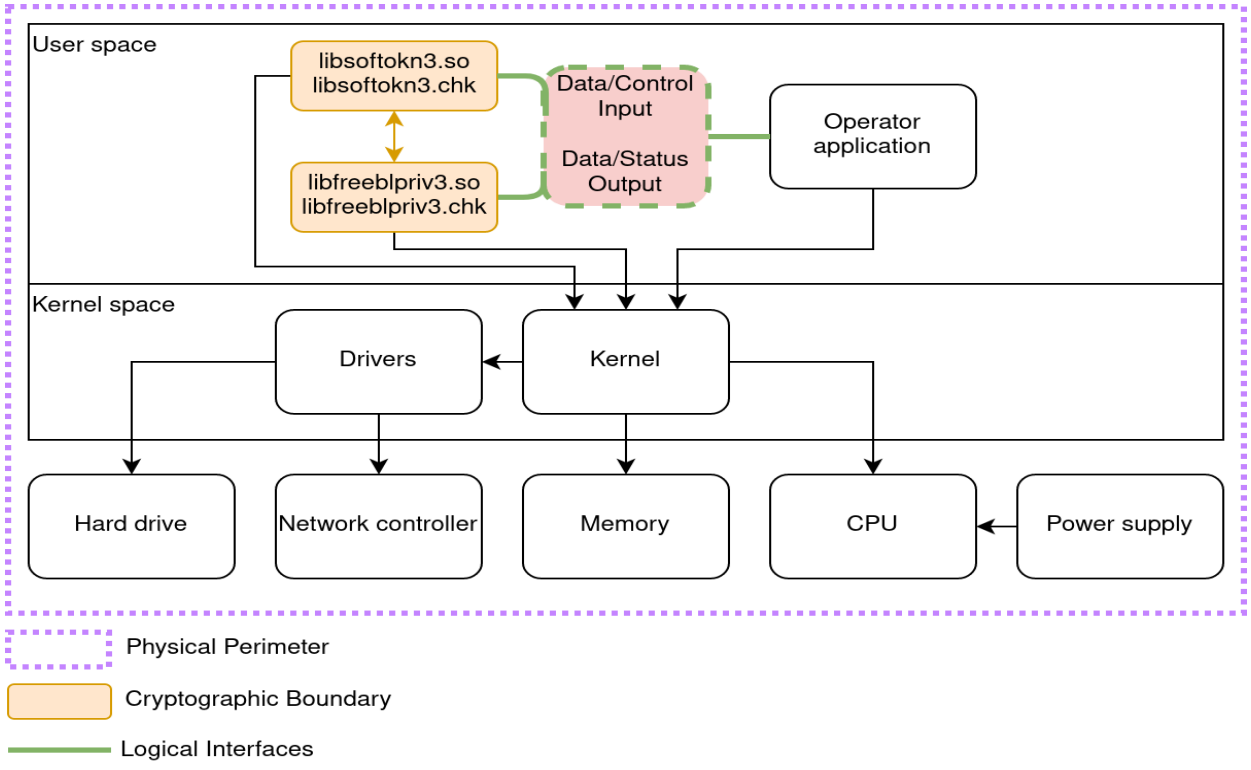

Figure 1: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libsoftokn3.so and libfreeblpriv3.so on ORACLE SERVER X9-2c with Intel(R) Xeon(R) Platinum 8358	4.35.0-381552536e763d0c	N/A	HMAC-SHA-256
libsoftokn3.so and libfreeblpriv3.so on ORACLE SERVER E4-2c with AMD EPYC 7J13	4.35.0-381552536e763d0c	N/A	HMAC-SHA-256
libsoftokn3.so and libfreeblpriv3.so on ORACLE SERVER A1-2c with Ampere(R) Altra(R) Q80-30	4.35.0-381552536e763d0c	N/A	HMAC-SHA-256

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

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Oracle Linux 9	ORACLE SERVER X9-2c	Intel® Xeon® Platinum 8358	Yes	KVM on Oracle Linux 8	4.35.0-381552536e763d0c
Oracle Linux 9	ORACLE SERVER E4-2c	AMD EPYC 7J13	Yes	KVM on Oracle Linux 8	4.35.0-381552536e763d0c
Oracle Linux 9	ORACLE SERVER A1-2c	Ampere® Altra® Q80- 30	Yes	KVM on Oracle Linux 8	4.35.0-381552536e763d0c
Oracle Linux 9	ORACLE SERVER X9-2c	Intel® Xeon® Platinum 8358	No	KVM on Oracle Linux 8	4.35.0-381552536e763d0c
Oracle Linux 9	ORACLE SERVER E4-2c	AMD EPYC 7J13	No	KVM on Oracle Linux 8	4.35.0-381552536e763d0c
Oracle Linux 9	ORACLE SERVER A1-2c	Ampere® Altra® Q80- 30	No	KVM on Oracle Linux 8	4.35.0-381552536e763d0c

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

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

Operating System	Hardware Platform
Oracle Linux 9	Oracle X Series Servers
Oracle Linux 9	Oracle E Series Servers
Oracle Linux 9	Oracle A Series Servers
Oracle Linux 9	Marvell T93 LiquidIO III (ARM v8.x) SmartNIC
Oracle Linux 9	Pensando DSC-200-R (ARM v8.x) SmartNIC

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 mode	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	Automatically entered whenever a non-approved service is requested	Non-Approved	Equivalent to the indicator of the requested service as defined in section 4.4

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. 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.

Mode Change Instructions and Status:

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

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A4760	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A4767	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A4769	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC-CS1	A4765	Direction - decrypt, encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 128-65536 Increment 8	SP 800-38A
AES-CMAC	A4762	Direction - Generation, Verification Key Length - 128, 192, 256 MAC Length - MAC Length: 128 Message Length - Message Length: 8-524288 Increment 8	SP 800-38B

Algorithm	CAVP Cert	Properties	Reference
AES-CTR	A4760	Direction - Decrypt, Encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 8-128 Increment 8 Supports Counter larger than maximum value - No Incremental Counter - Yes Counter Tests Performed - Yes	SP 800-38A
AES-CTR	A4769	Direction - Decrypt, Encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 8-128 Increment 8 Supports Counter larger than maximum value - No Incremental Counter - Yes Counter Tests Performed - Yes	SP 800-38A
AES-ECB	A4760	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A4767	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A4769	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A4760	Direction - Decrypt, Encrypt IV Generation - External, Internal IV Generation Mode - 8.2.1, 8.2.2 Key Length - 128, 192, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96, 128 Payload Length - Payload Length: 128, 1024, 120, 248 AAD Length - AAD Length: 0, 128, 1024, 120, 248	SP 800-38D
AES-GCM	A4767	Direction - Decrypt, Encrypt IV Generation - External, Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96, 128 Payload Length - Payload Length: 128, 1024, 120, 248 AAD Length - AAD Length: 0, 128, 1024, 120, 248	SP 800-38D
AES-GCM	A4769	Direction - Decrypt, Encrypt IV Generation - External, Internal IV Generation Mode - 8.2.1, 8.2.2 Key Length - 128, 192, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96, 128 Payload Length - Payload Length: 128, 1024, 120, 248 AAD Length - AAD Length: 0, 128, 1024, 120, 248	SP 800-38D
AES-KW	A4761	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 128-4096 Increment 128	SP 800-38F
AES-KW	A4766	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 128-4096 Increment 128	SP 800-38F

Algorithm	CAVP Cert	Properties	Reference
AES-KW	A4768	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 128-4096 Increment 128	SP 800-38F
AES-KWP	A4761	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 8-4096 Increment 8	SP 800-38F
AES-KWP	A4766	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 8-4096 Increment 8	SP 800-38F
AES-KWP	A4768	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 8-4096 Increment 8	SP 800-38F
DSA SigVer (FIPS186-4)	A4760	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)	A4760	Curve - P-256, P-384, P-521 Secret Generation Mode - Extra Bits	FIPS 186-4
ECDSA KeyVer (FIPS186-4)	A4760	Curve - P-256, P-384, P-521	FIPS 186-4
ECDSA SigGen (FIPS186-4)	A4760	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)	A4760	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	A4760	Prediction Resistance - No, Yes Supports Reseed - No Mode - SHA2-256 Entropy Input - Entropy Input: 256 Nonce - Nonce: 256 Personalization String Length - Personalization String Length: 0, 256 Additional Input - Additional Input: 0, 256 Returned Bits - 1024	SP 800-90A Rev. 1
HMAC-SHA2-224	A4760	MAC - MAC: 224 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A4760	MAC - MAC: 256 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A4760	MAC - MAC: 384 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A4760	MAC - MAC: 512 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A4760	Domain Parameter Generation Methods - P-256, P-384, P-521 Scheme -	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
		ephemeralUnified - KAS Role - initiator, responder	
KAS-FFC-SSC Sp800-56Ar3	A4760	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	A4759	Fixed Info Pattern - uPartyInfo||vPartyInfo Fixed Info Encoding - concatenation 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)	A4764	Initiator Nonce Length - Initiator Nonce Length: 128, 256, 512, 2048 Responder Nonce Length - Responder Nonce Length: 128, 256, 512, 2048 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	A4763	KDF Mode - Counter, Double Pipeline Iteration, Feedback MAC Mode - CMAC-AES128, CMAC-AES192, CMAC-AES256, HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512 Supported Lengths - Supported Lengths: 8, 72, 128, 776, 3456, 4096 Fixed Data Order - After Fixed Data, Before Fixed Data Counter Length - 16, 24, 32, 8 Supports Empty IV - No Requires Empty IV - No Custom Key In Length - 0	SP 800-108 Rev. 1
KDF TLS (CVL)	A4760	TLS Version - v1.0/1.1	SP 800-135 Rev. 1
PBKDF	A4760	Iteration Count - Iteration Count: 1000-10000 Increment 1 HMAC Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512 Password Length - Password Length: 8-128 Increment 1 Salt Length - Salt Length: 128-4096 Increment 8 Key Data Length - Key Data Length: 128-256 Increment 8	SP 800-132
RSA KeyGen (FIPS186-4)	A4760	Key Generation Mode - B.3.3 Modulo - 2048, 3072, 4096 Primality Tests - Table C.3 Info Generated By Server - No Public Exponent Mode - Fixed Fixed Public Exponent - 010001 Private Key Format - Standard	FIPS 186-4

Algorithm	CAVP Cert	Properties	Reference
RSA SigGen (FIPS186-4)	A4760	Signature Type - PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA2-224	FIPS 186-4
RSA SigVer (FIPS186-2)	A4760	Public Exponent Mode - Fixed Fixed Public Exponent - 010001 Signature Type - PKCS 1.5, PKCSPSS Modulo - 1024, 1536 Hash Pair - Hash Algorithm - SHA-1	FIPS 186-4
RSA SigVer (FIPS186-4)	A4760	Signature Type - PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA-1 Public Exponent Mode - Fixed Fixed Public Exponent - 010001	FIPS 186-4
Safe Primes Key Generation	A4760	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	A4760	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A4760	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-384	A4760	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A4760	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
TLS v1.2 KDF RFC7627 (CVL)	A4760	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512 Key Block Length - Key Block Length: 1024	SP 800-135 Rev. 1

Table 6: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
Cryptographic Key Generation (CKG)	Key Type:Symmetric and Asymmetric Symmetric Key Generation:112-256 bits with 112-256 bits of key strength. RSA:2048, 3072, 4096 bits with 112, 128, 149 bits of key strength. ECDSA: P-256, P-384, P-521 elliptic curves with 128-256 bits of key strength Safe Primes Key Generation:MODP-2048, MODP-3072, MODP-4096, MODP-6144, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 with 112-200 bits of key strength	N/A	SP 800-133Rev2 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 allowed in the approved mode of operation.

Non-Approved, Allowed Algorithms with No Security Claimed:

Name	Caveat	Use and Function
MD5	Only allowed as the PRF in TLSv1.0 and v1.1 per IG 2.4.A	Message digest used in TLS 1.0/1.1 KDF only

Table 8: Non-Approved, Allowed Algorithms 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), AES GCM (external IV)	Encryption
RC2, RC4, DES, Triple-DES, CDMF, Camellia, SEED, ChaCha20(-Poly1305)	Decryption
CBC-MAC, AES XCBC-MAC, AES XCBC-MAC-96, HMAC-SHA-1, HMAC (MD2, MD5; < 112-bit keys), HMAC/SSLv3 MAC (constant-time implementation)	Message Authentication
RSA OAEP	Key Encapsulation/Decapsulation
SEED, ANS X9.63 KDF, SSL 3 PRF, 40v1 PRF, KBKDF, HKDF, TLS 1.0/1.1 KDF, TLS 1.2 KDF, IKEv2 PRF (< 112-bit keys), KBKDF (MD2, MD5), TLS 1.2 KDF (without extended master secret), IKEv1 KDF, IKEv2 PRF (MD2, MD5)	Key Derivation
PKCS#5 PBE, PKCS#12 PBE, PBKDF2 (short password; short salt; insufficient iterations; < 112-bit keys)	Password-Based Key Derivation
J-PAKE, KAS-FFC-SSC (FIPS 186-type groups), KAS-ECC-SSC (P-192)	Shared Secret Computation
DSA SigGen, RSA with SHA-1, RSA (primitive; PKCS#1 v1.5 or PSS with MD2, MD5), ECDSA (P-192)	Signature Generation
RSA with SHA-1, RSA (primitive; PKCS#1 v1.5 or PSS with MD2, MD5), ECDSA (P-192)	Signature Verification
RSA Encryption	Asymmetric Encryption
RSA Decryption	Asymmetric Decryption
DH (FIPS 186-type groups), RSA (< 2048 bits), DSA, ECDSA (P-192)	Key Pair Generation
Symmetric key generation (< 112 bits)	Secret Key Generation
DSA ParmGen	Parameter Generation
DSA ParmVer	Parameter Verification

Table 9: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Key Derivation	KAS-56CKDF KBKDF	Key Derivation		KDA HKDF Sp800-56Cr1: (A4759) KDF SP800-108: (A4763)
Encryption UnAuth	BC-UnAuth	Encryption		AES-CBC: (A4769, A4767, A4760) AES-CTR: (A4769, A4760) AES-ECB: (A4769, A4767, A4760) AES-CBC-CS1: (A4765)
Decryption UnAuth	BC-UnAuth	Decryption		AES-CBC: (A4769, A4767, A4760) AES-CTR: (A4769,

Name	Type	Description	Properties	Algorithms
				A4760) AES-ECB: (A4769, A4767, A4760) AES-CBC-CS1: (A4765)
Encryption	BC-Auth	Encryption		AES-GCM: (A4769, A4767, A4760)
Decryption	BC-Auth	Decryption		AES-GCM: (A4769, A4767, A4760)
Key Pair Generation	AsymKeyPair-KeyGen CKG	Key Pair Generation		ECDSA KeyGen (FIPS186-4): (A4760) RSA KeyGen (FIPS186-4): (A4760) Safe Primes Key Generation: (A4760)
Key Pair Verification	AsymKeyPair-KeyVer	Key Pair Verification		ECDSA KeyVer (FIPS186-4): (A4760)
Signature Generation	DigSig-SigGen	Signature Generation		ECDSA SigGen (FIPS186-4): (A4760) RSA SigGen (FIPS186-4): (A4760)
Signature Verification	DigSig-SigVer	Signature Verification		ECDSA SigVer (FIPS186-4): (A4760) RSA SigVer (FIPS186-2): (A4760) RSA SigVer (FIPS186-4): (A4760) DSA SigVer (FIPS186-4): (A4760)
Message Authentication	MAC	Message Authentication		HMAC-SHA2-224: (A4760) HMAC-SHA2-256: (A4760) HMAC-SHA2-384: (A4760) HMAC-SHA2-512: (A4760) AES-CMAC: (A4762)
Random Number Generation	DRBG	Random Number Generation		Hash DRBG: (A4760)

Name	Type	Description	Properties	Algorithms
Shared Secret Computation	KAS-SSC	Shared Secret Computation		KAS-ECC-SSC Sp800-56Ar3: (A4760) KAS-FFC-SSC Sp800-56Ar3: (A4760)
Password-Based Key Derivation	PBKDF	Password-Based Key Derivation		PBKDF: (A4760)
Message Digest	SHA	Message Digest		SHA2-224: (A4760) SHA2-256: (A4760) SHA2-384: (A4760) SHA2-512: (A4760)
Key Derivation (CVL)	KAS-135KDF	Key Derivation (CVL)		KDF TLS: (A4760) TLS v1.2 KDF RFC7627: (A4760) KDF IKEv2: (A4764)
Key Wrapping	KTS-Wrap BC-Auth	Key Wrapping	Standard:SP800-38F IG D.G:approved AES KW/KWP (SP800-38F) key wrapping Caveat:Key establishment methodology provides between 128 and 256 bits of security strength	AES-KW: (A4761, A4766, A4768) AES-KWP: (A4761, A4766, A4768)
Key Unwrapping	KTS-Wrap BC-Auth	Key Unwrapping	Standard:SP800-38F IG D.G:approved AES KW/KWP (SP800-38F) key unwrapping Caveat:Key establishment methodology provides between 128 and 256 bits of security strength	AES-KW: (A4761, A4766, A4768) AES-KWP: (A4761, A4766, A4768)

Table 10: 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 compliant with Scenario 2 of FIPS 140-3 IG C.H. These IVs are always 96 bits and generated using the approved DRBG internal to the module's boundary.

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

Finally, for TLS 1.3, the AES GCM implementation uses the context of Scenario 5 of FIPS 140-3 IG C.H. The protocol that provides this compliance is TLS 1.3, defined in 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 with SP 800-132 and FIPS 140-3 IG D.N, the following requirements shall be met:

- Derived keys shall only be used in storage applications. The MK shall not be used for other purposes. The module accepts a minimum length of 112 bits for the MK or DPK.
- 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 10^{-8}. Combined with the minimum iteration count as described below, this provides an acceptable trade-off between user experience and security against brute-force attacks.
- A portion of the salt, with a length of at least 128 bits (this is verified by the module to determine the service is approved), shall be generated randomly using the SP 800-90Ar1 DRBG provided by the module.
- The iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users. The module only allows minimum iteration count to be 1000.

2.7.3 SP 800-56Ar3 Assurances

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

- The entity using the module, must use module's "Key pair generation" service for generating DH/ECDH ephemeral keys. This meets the assurances required by key pair owner defined in the section 5.6.2.1 of SP 800-56Ar3.
- As part of the module's shared secret computation (SSC) service, the module internally performs the public key validation the peer's public key passed in as input to the SSC function. This meets the public key validity assurance required by the sections 5.6.2.2.1/5.6.2.2.2 of SP 800-56Ar3.
- The module does not support static keys therefore the "assurance of peer's possession of private key" is not applicable.

2.7.4 KAS-SSC

The module offers as a service to an external operator (e.g., calling application) CAVP-tested KAS SSC that map to IG D.F Scenario 2 path (1) without establishing a module key/SSP used by the module for cryptographic protection. The module offers KAS-ECC-SSC as a service that receives as inputs all keying material necessary to compute and output the shared secret without applying a KDF/KDA which is a separate module service.

2.7.5 SHA-1

SHA-1 from Message Digest is only approved for non-digital-signature uses (see Table 8 of SP 800-131Ar2).

2.7.6 RSA

All supported modulus sizes for RSA signature verification have been CAVP tested. CAVP testing performed on the module's legacy implementations, specifically digital signature verification using SHA-1 or modulus sizes with a strength < 112-bits, were not testable under FIPS 186-5, but have been tested under FIPS 186-4 per IG C.K additional comment 2.

2.8 RBG and Entropy

Cert Number	Vendor Name
E99	Oracle Corp

Table 11: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Oracle Userspace CPU Time Jitter RNG Entropy Source (Cert. #E99)	Non-Physical	Oracle Linux 9 on Oracle SERVER X9-2c; Oracle Linux 9 on ORACLE SERVER E4-2c; Oracle Linux 9 on ORACLE SERVER A1-2c	256	256 bits sample size with full entropy	LFSR, HMAC_DRBG_SHA2-512

Table 12: Entropy Sources

RNG Information:

The module implements a Deterministic Random Bit Generator (Hash_DRBG) implementation compliant with 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 module DRBG implemented is a SHA-256 Hash_DRBG, seeded by the entropy source specified in the table above.

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. 2.10 Key Establishment. The key generation methods are specified in the Vendor Affirmed Algorithms table and the Security Function Implementations table.

2.10 Key Establishment

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

2.11 Industry Protocols

For DH, 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.

TLS 1.0/1.1 KDF, TLS 1.2 KDF (RFC 7627), IKEv2 implementations shall only be used to generate secret keys in the context of the TLS 1.0/1.1, TLS 1.2, IKE protocols respectively. No parts of this protocol, other than the approved cryptographic algorithms and the KDFs, 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 data input parameters
N/A	Data Output	API output parameters
N/A	Control Input	API function calls, API control input parameters
N/A	Status Output	API return codes, error queue

Table 13: Ports and Interfaces

The logical interfaces are the APIs through which the applications request services. These logical interfaces are logically separated from each other by the API design.

The module does not implement the control output interface.

4 Roles, Services, and Authentication

4.1 Authentication Methods

The module does not implement authentication.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	Crypto Officer	None

Table 14: 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	Message digest	Message Digest	Crypto Officer
Encryption	Encrypt a plaintext	CKS_NSS_FIPS_OK	AES Key, plaintext, IV	Ciphertext	Encryption UnAuth Encryption	Crypto Officer - AES Key: W,E
Decryption	Decrypt a ciphertext	CKS_NSS_FIPS_OK	AES Key, ciphertext, IV	Plaintext	Decryption UnAuth Decryption	Crypto Officer - AES Key: W,E
Key Wrapping	Wrap a key	CKS_NSS_FIPS_OK	AES key, key to be wrapped	Wrapped key	Key Wrapping	Crypto Officer - Key Wrapping Key: W,E - Wrapped key: R - Key To Be Wrapped: W,E
Key Unwrapping	Unwrap a key	CKS_NSS_FIPS_OK	AES key, key to unwrap	Unwrapped key	Key Unwrapping	Crypto Officer - Key Unwrapping Key: W,E - Unwrapped key: R - Key To Be Unwrapped: W,E
Message Authentication CMAC	Compute a MAC tag	CKS_NSS_FIPS_OK	AES key, message	MAC tag	Message Authentication	Crypto Officer - AES Key: W,E
Message Authentication HMAC	Compute a MAC tag	CKS_NSS_FIPS_OK	HMAC key, message	MAC tag	Message Authentication	Crypto Officer - HMAC Key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Shared Secret Computation	Compute a Shared Secret	CKS_NSS_FIPS_OK	Private Key and Public Key	Shared Secret	Shared Secret Computation	Crypto Officer - DH Private Key: W,E - DH Public Key: W,E - EC Private Key: W,E - EC Public Key: W,E - Shared Secret: G,R
Signature Generation	Generate a signature	CKS_NSS_FIPS_OK	Private Key, message, hash	Signature	Signature Generation	Crypto Officer - EC Private Key: W,E - RSA Private Key: W,E
Signature Verification	Verify a signature	CKS_NSS_FIPS_OK; CKS_NSS_FIPS_OK; CKR_OK	Public Key, message, signature, hash	Pass or Fail	Signature Verification	Crypto Officer - EC Public Key: W,E - RSA Public Key: W,E
Key Pair Generation	Generate a key pair	CKS_NSS_FIPS_OK	modulus bits, Curve or Group	Generated Private Key and Public Key	Key Pair Generation	Crypto Officer - EC Private Key: G,R - EC Public Key: G,R - DH Private Key: G,R - DH Public Key: G,R - RSA Public Key: G,R - RSA Private Key: G,R - Intermediate Key Generation Value: G,E,Z
Key Pair Verification	Verify a key pair	CKS_NSS_FIPS_OK	Private Key, Public Key	Pass or Fail	Key Pair Verification	Crypto Officer - EC Private Key: G,R - EC Public Key: G,R
Secret Key Generation	Generate a secret key	CKS_NSS_FIPS_OK	random bits	Symmetric Key	None	Crypto Officer - AES Key: G,R - HMAC Key: G,R
Key Derivation	Derive a key	CKS_NSS_FIPS_OK	Shared Secret	Derived Key	Key Derivation Key	Crypto Officer - Shared Secret: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					Derivation (CVL)	- Derived Key: G,R
Key-Based Key Derivation	Derive a key from a key	CKS_NSS_FIPS_OK	Derivation Key	Derived Key	Key Derivation	Crypto Officer - Key-Derivation Key: W,E - Derived Key: G,R
Password-Based Key Derivation	Derive a key from a password	CKS_NSS_FIPS_OK	Password	Derived Key	Password-Based Key Derivation	Crypto Officer - Password: W,E - Derived Key: G,R
Random Number Generation	Generate random bytes	CKR_OK	number of bits	Random Number	Random Number Generation	Crypto Officer - Entropy Input: W,E - DRBG Seed: G,E - Internal State (V,C): G,E
Show Version	Return the name and version information	None	None	Module name and version	None	Unauthenticated
Show Status	Return the module status	None	None	Module status	None	Unauthenticated
Self-Test	Perform the CASTs and integrity test	None	None	Pass or Fail	None	Unauthenticated
Zeroization	Zeroize all SSPs	None	N/A	N/A	None	Crypto Officer - Entropy Input: Z - DRBG Seed: Z - Internal State (V,C): Z - AES Key: Z - Wrapped key: Z - Unwrapped key: Z - Key To Be Wrapped: Z - Key To Be Unwrapped: Z - HMAC Key: Z - Key-Derivation Key: Z - Shared Secret:

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Z - Password: Z - Derived Key: Z - DH Public Key: Z - DH Private Key: Z - EC Public Key: Z - EC Private Key: Z - RSA Public Key: Z - RSA Private Key: Z - Intermediate Key Generation Value: Z - Key Wrapping Key: Z - Key Unwrapping Key: Z

Table 15: 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.

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 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-SHA-1, HMAC (MD2, MD5; < 112-bit keys), HMAC/SSLv3 MAC (constant-time implementation)	CO
Key Encapsulation/Decapsulation	Encapsulate/Decapsulate a key	RSA OAEP	CO
Key Derivation	Derive a key	SEED, ANS X9.63 KDF, SSL 3 PRF, 40v1 PRF, KBKDF, HKDF, TLS 1.0/1.1 KDF, TLS 1.2 KDF, IKEv2 PRF (< 112-bit keys), KBKDF (MD2, MD5), TLS 1.2 KDF (without extended master secret), IKEv1 KDF, IKEv2 PRF (MD2, MD5)	CO
Password-Based Key Derivation	Derive a key from a password	PKCS#5 PBE, PKCS#12 PBE, PBKDF2 (short password; short salt; insufficient iterations; < 112-bit keys)	CO
Shared Secret Computation	Compute a shared secret	J-PAKE, KAS-FFC-SSC (FIPS 186-type groups), KAS-ECC-SSC (P-192)	CO
Signature Generation	Generate a signature	DSA SigGen, RSA with SHA-1, RSA (primitive; PKCS#1 v1.5 or PSS with MD2, MD5), ECDSA (P-192)	CO
Signature Verification	Verify a signature	RSA with SHA-1, RSA (primitive; PKCS#1 v1.5 or PSS with MD2, MD5), ECDSA (P-192)	CO
Asymmetric Encryption	Encrypt a plaintext	RSA Encryption	CO
Asymmetric Decryption	Decrypt a ciphertext	RSA Decryption	CO
Parameter Generation	Generate domain parameters	DSA ParmGen	CO
Parameter Verification	Verify domain parameters	DSA ParmVer	CO
Key Pair Generation	Generate a key pair	DH (FIPS 186-type groups), RSA (< 2048 bits), DSA, ECDSA (P-192)	CO
Secret Key Generation	Generate a secret key	Symmetric key generation (< 112 bits)	CO

Table 16: Non-Approved Services

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 comparing a HMAC-SHA-256 value calculated at run time with the HMAC-SHA-256 value embedded in the module that was 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 test may be invoked on-demand by unloading and subsequently re-initializing the module.

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.

There are no concurrent operators.

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

Instrumentation tools like the ptrace system call, gdb and strace, 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. The module does not perform persistent storage of SSPs	Dynamic

Table 17: Storage Areas

The module does not perform persistent storage of SSPs. The SSPs are temporarily stored in the RAM in plaintext form. SSPs are provided to the module by the calling process and are destroyed when released by the appropriate zeroization function calls.

9.2 SSP Input-Output Methods

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

Table 18: SSP Input-Output Methods

The module does not support entry and output of SSPs beyond the physical perimeter of the operational environment. The SSPs are provided to the module via API input parameters in the plaintext form and output via API output parameters in the plaintext form to and from the calling application running on the same operational environment.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Wipe and Free memory block allocated	Zeroizes the SSPs contained within the cipher handle.	Memory occupied by SSPs is overwritten with zeroes and then it is released, which renders the SSP values irretrievable. The completion of the zeroization routine indicates that the zeroization procedure succeeded.	By calling the cipher related zeroization API
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
Module Reset	De-allocates the volatile memory used to store SSPs	Volatile memory used by the module is overwritten within nanoseconds when power is removed.	By unloading and reloading the module

Table 19: 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 (CSP) used for: Use: Encryption, Decryption, Message Authentication, Message Authentication Verification;	128-256 bits - 128-256 bits	Symmetric Key - CSP - CSP			Encryption UnAuth Decryption UnAuth Encryption Decryption
Wrapped key	Wrapped key (CSP) used for: Use: Key Wrapping;	128, 192, 256 bit keys - 112-256 bits	Wrapped Key - CSP - CSP			
Unwrapped key	Unwrapped key (CSP) used for: Use: Key Unwrapping;	128, 192, 256 bit keys - 112-256 bits	Unwrapped Key - CSP - CSP			
Key To Be Wrapped	Key To Be Wrapped (CSP) used for: Use: Key Wrapping;	128-256 bits - 112-256 bits	Key - CSP - CSP			Key Wrapping
Key To Be Unwrapped	Key To Be Unwrapped (CSP) used for: Use: Key Unwrapping;	128-256 bits - 112-256 bits	Key - CSP - CSP			Key Unwrapping
Key Wrapping Key	Key Wrapping key (CSP) used for: Use: Key Wrapping;	128, 192, 256 bit keys - 128-256 bits	Key Wrapping Key - CSP - CSP			Key Wrapping
Key Unwrapping Key	Key Unwrapping Key (CSP) used for: Use: Key Unwrapping;	128, 192, 256 bit keys - 128-256 bits	Key Unwrapping Key - CSP - CSP			Key Unwrapping
HMAC Key	HMAC Key (CSP) used for: Use: Message Authentication, Message	112-256 bits - 112-256 bits	HMAC key - CSP - CSP			Message Authentication

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	Authentication Verification;					
Key-Derivation Key	Key-Derivation Key (CSP) used for: Use: Key-Based Key Derivation;	112-256 bits - 112-256 bits	Key - CSP - CSP			Key Derivation
Shared Secret	Shared Secret (CSP) used for: Use: Shared Secret Computation, Key Derivation;	ECDH:128-256 bits; DH:112-200 bits - ECDH:128-256 bits; DH:112-200 bits	Shared Secret - CSP - CSP	Shared Secret Computation		
Password	Password (CSP) used for: Use: Password-Based Key Derivation;	at least 8 characters - N/A	Password - CSP - CSP			Password-Based Key Derivation
Derived Key	Derived Key (CSP) used for: Use: Key Derivation, Key-Based Key Derivation, Password-Based Key Derivation;	112-4096 bits - 112-256 bits	Symmetric key - CSP - CSP	Key Derivation Password-Based Key Derivation		
Entropy Input	Entropy Input (CSP) used for: Use: Random Number Generation;Compliant with IG D.L	128-448 bits - 128-256 bits	Entropy Input - CSP - CSP			Random Number Generation
DRBG Seed	DRBG Seed (CSP) used for: Use: Random Number Generation;Compliant with IG D.L	128-256 bits - 128-256 bits	Seed - CSP - CSP	Random Number Generation		Random Number Generation
Internal State (V,C)	Internal State (V, C) (CSP) used for: Use: Random Number Generation;Compliant with IG D.L	128-256 bits - 128-256 bits	Internal state - CSP - CSP	Random Number Generation		Random Number Generation
DH Public Key	DH Public Key (PSP) used for: Use: Shared Secret Computation, Key Pair Generation;	2048, 3072, 4096, 6144, 8192 bits - 112-200 bits	Public key - PSP			Shared Secret Computation
DH Private Key	DH Private Key (CSP) used for: Use: Shared Secret Computation, Key Pair Generation;	2048, 3072, 4096, 6144, 8192 bits -	Private key - CSP			Shared Secret Computation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		112-200 bits				
EC Public Key	EC Public Key (PSP) used for: Use: Signature Verification, Shared Secret Computation;	P-256-P-521 bits - 128-256 bits	Public key - PSP			Shared Secret Computation
EC Private Key	EC Private Key (CSP) used for: Use: Signature Generation, Shared Secret Computation;	P-256-P-521 bits - 128-256 bits	Private key - CSP			Shared Secret Computation
RSA Public Key	RSA Public Key (PSP) used for: Use: Signature Verification;	1024, 2048, 3072, 4096 bits - 80-150 bits	Public key - PSP			Key Pair Verification Signature Generation
RSA Private Key	RSA Private Key (CSP) used for: Use: Signature Generation;	1024, 2048, 3072, 4096 bits - 112-150 bits	Private key - CSP			Key Pair Verification Signature Verification
Intermediate Key Generation Value	Intermediate Key Generation Value (CSP) used for: Use: Key Pair Generation;	224-4096 bits - 112-256 bits	Key - CSP - CSP	Shared Secret Computation		

Table 20: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES Key	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	
Wrapped key	API output parameters (encrypted)	RAM:Plaintext	From service invocation to service completion	Module Reset	Key Wrapping Key:Wrapped by Key To Be Wrapped:Wrapped from
Unwrapped key	API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	Key Unwrapping Key:Unwrapped by Key To Be Unwrapped:Unwrapped from
Key To Be Wrapped	API input parameters (plaintext)	RAM:Plaintext	From service invocation to	Module Reset	Key Wrapping Key: Wrapped by

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
			service completion		
Key To Be Unwrapped	API input parameters (encrypted)	RAM:Plaintext	From service invocation to service completion	Module Reset	Key Unwrapping Key:Unwrapped by
Key Wrapping Key	API input parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	Key To Be Wrapped:Wraps
Key Unwrapping Key	API input parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	Key To Be Unwrapped:Unwraps
HMAC Key	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	
Key-Derivation Key	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	Derived Key:Derives
Shared Secret	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	DH Public Key:Established by DH Private Key:Established by EC Public Key:Established by EC Private Key:Established by
Password	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	Derived Key:Derived by
Derived Key	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	Key-Derivation Key:Derived From Shared Secret:Derived From Password:Derived From
Entropy Input	API input parameters (plaintext) API output	RAM:Plaintext	From service invocation to service completion	Module Reset	DRBG Seed:Derives

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
	parameters (plaintext)				
DRBG Seed	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	Entropy Input:Derived From Internal State (V,C):Used With
Internal State (V,C)	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion		DRBG Seed:Generated from
DH Public Key	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	DH Private Key:Paired With Shared Secret:Establishes
DH Private Key	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	DH Public Key:Paired With Shared Secret:Establishes
EC Public Key	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	EC Private Key:Paired With Shared Secret:Establishes
EC Private Key	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	EC Public Key:Paired With Shared Secret:Estabksihes
RSA Public Key	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	RSA Private Key:Paired With
RSA Private Key	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion	Module Reset	RSA Public Key:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Intermediate Key Generation Value	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	From service invocation to service completion		DH Public Key:Generated with DH Private Key:Generated with EC Public Key:Generated with EC Private Key:Generated with RSA Public Key:Generated with RSA Private Key:Generated with

Table 21: 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 RSA, ECDSA algorithm as implemented by the module conforms to FIPS 186-4, which has been superseded by FIPS 186-5. FIPS 186-4 was withdrawn on February 3, 2024.

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A4760)	SHA2-256	HMAC	SW/FW Integrity	Module becomes operational and services are available for use.	Integrity test for libfreeblpriv3.so and libfsoftokn3.so

Table 22: Pre-Operational Self-Tests

The pre-operational software integrity test is performed automatically, after the CASTs, when the module is powered on before the module transitions into the operational state. 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. The module transitions to the operational state only after the pre-operational self-test has passed successfully. If the pre-operational self-test fails, the module transitions to the error (i.e., Power-On Error) state.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-GCM (A4769)	128-bit key, encrypt	KAT	CAST	Module becomes operational and services are available for use.	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A4767)	128-bit key, encrypt	KAT	CAST	Module becomes operational and services are available for use.	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A4760)	128-bit key, encrypt	KAT	CAST	Module becomes operational and services are available for use.	Symmetric operation	Test runs at power-on before the integrity test
SHA2-224 (A4760)	SHA2-224	KAT	CAST	Module becomes operational and services are available for use.	Message authentication	Test runs at power-on before the integrity test
SHA2-256 (A4760)	SHA2-256	KAT	CAST	Module becomes operational and services are available for use.	Message authentication	Test runs at power-on before the integrity test
SHA2-384 (A4760)	SHA2-384	KAT	CAST	Module becomes operational and services are available for use.	Message authentication	Test runs at power-on before the integrity test
SHA2-512 (A4760)	SHA2-512	KAT	CAST	Module becomes operational and services are available for use.	Message authentication	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-224 (A4760)	SHA2-224, HMAC key size 288-bit	KAT	CAST	Module becomes operational and services are available for use.	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-256 (A4760)	SHA2-256, HMAC key size 288-bit	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-384 (A4760)	SHA2-384, HMAC key size 288-bit	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-512 (A4760)	SHA2-512, HMAC key size 288-bit	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Test runs at power-on before the integrity test
RSA KeyGen (FIPS186-4) (A4760)	SHA2-256 and respective keys	PCT	PCT	Successful key pair generation	Signature generation & verification	Key pair generation
RSA SigGen (FIPS186-4) (A4760)	PKCS#1 v1.5 with 2048 bit key and SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Digital signature generation	Test runs at power-on before the integrity test
RSA SigVer (FIPS186-4) (A4760)	PKCS#1 v1.5 with 2048 bit key and SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Digital signature verification	Test runs at power-on before the integrity test
ECDSA KeyGen (FIPS186-4) (A4760)	SHA2-256 and respective keys	PCT	PCT	Successful key pair generation	Signature generation & verification	Key pair generation
ECDSA SigGen (FIPS186-4) (A4760)	P-224 with SHA-224	KAT	CAST	Module becomes operational and services are available for use	Digital signature generation	Test runs at power-on before the integrity test
ECDSA SigVer (FIPS186-4) (A4760)	P-224 with SHA-224	KAT	CAST	Module becomes operational and services are available for use	Digital signature verification	Test runs at power-on before the integrity test
KAS-ECC-SSC Sp800-56Ar3 (A4760)	P-224 curve	KAT	CAST	Module becomes operational and services are available for use	Shared secret computation	Test runs at power-on before the integrity test
KAS-FFC-SSC Sp800-56Ar3 (A4760)	MODP-2048	KAT	CAST	Module becomes operational and services are available for use	Shared secret computation	Test runs at power-on before the integrity test
AES-CBC (A4769)	128-bit key encrypt	KAT	CAST	Module becomes operational and	Symmetric operation	Test runs at power-on

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		before the integrity test
AES-CBC (A4767)	128-bit key encrypt	KAT	CAST	Module becomes operational and services are available for use	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A4760)	128-bit key encrypt	KAT	CAST	Module becomes operational and services are available for use	Symmetric operation	Test runs at power-on before the integrity test
AES-ECB (A4769)	128-bit key decrypt	KAT	CAST	Module becomes operational and services are available for use	Symmetric operation	Test runs at power-on before the integrity test
AES-ECB (A4767)	128-bit key decrypt	KAT	CAST	Module becomes operational and services are available for use	Symmetric operation	Test runs at power-on before the integrity test
AES-ECB (A4760)	128-bit key decrypt	KAT	CAST	Module becomes operational and services are available for use	Symmetric operation	Test runs at power-on before the integrity test
AES-CMAC (A4762)	128, 192, 256-bit key 128-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Test runs at power-on before the integrity test
Safe Primes Key Generation (A4760)	N/A	PCT	PCT	Module becomes operational and services are available for use	Key Generation, SP 800-56Ar3 section 5.6.2.1.4	Key pair generation
KDF SP800-108 (A4763)	Counter mode HMAC SHA-256 576-bit input key	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Test runs at power-on before the integrity test
Hash DRBG (A4760)	SHA-256 without prediction resistance	KAT	CAST	Module becomes operational and services are available for use	Random Number Generation	Test runs at power-on before the integrity test
KDA HKDF Sp800-56Cr1 (A4759)	SHA-256 512-bit input secret	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Test runs at power-on before the integrity test
KDF TLS (A4760)	MD5-SHA-1 288-bit input secret	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Test runs at power-on before the integrity test
TLS v1.2 KDF RFC7627 (A4760)	SHA-256 288-bit input secret	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
PBKDF (A4760)	SHA-256 14-character password 128-bit salt Iteration count: 5	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Test runs at power-on before the integrity test
DSA SigVer (FIPS186-4) (A4760)	1024-bit key	KAT	CAST	Module becomes operational and services are available for use	Digital signature verification	Test runs at power-on before the integrity test

Table 23: 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. Data output through the data output interface is inhibited during the self-tests. If any of these tests fails, the module transitions to the Power-On Error state.

The pair-wise consistency tests (PCT), provide some assurance that the generated key pairs are well formed. Services are not available, and data output (via the data output interface) is inhibited during execution of a PCT. If a PCT test fails, the module transitions to the PCT Error state.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A4760)	HMAC	SW/FW Integrity	On Demand	Module restart

Table 24: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-GCM (A4769)	KAT	CAST	On Demand	Manually
AES-GCM (A4767)	KAT	CAST	On Demand	Manually
AES-GCM (A4760)	KAT	CAST	On Demand	Manually
SHA2-224 (A4760)	KAT	CAST	On Demand	Manually
SHA2-256 (A4760)	KAT	CAST	On Demand	Manually
SHA2-384 (A4760)	KAT	CAST	On Demand	Manually
SHA2-512 (A4760)	KAT	CAST	On Demand	Manually
HMAC-SHA2-224 (A4760)	KAT	CAST	On Demand	Manually
HMAC-SHA2-256 (A4760)	KAT	CAST	On Demand	Manually
HMAC-SHA2-384 (A4760)	KAT	CAST	On Demand	Manually
HMAC-SHA2-512 (A4760)	KAT	CAST	On Demand	Manually
RSA KeyGen (FIPS186-4) (A4760)	PCT	PCT	On Demand	Manually
RSA SigGen (FIPS186-4) (A4760)	KAT	CAST	On Demand	Manually
RSA SigVer (FIPS186-4) (A4760)	KAT	CAST	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
ECDSA KeyGen (FIPS186-4) (A4760)	PCT	PCT	On Demand	Manually
ECDSA SigGen (FIPS186-4) (A4760)	KAT	CAST	On Demand	Manually
ECDSA SigVer (FIPS186-4) (A4760)	KAT	CAST	On Demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A4760)	KAT	CAST	On Demand	Manually
KAS-FFC-SSC Sp800-56Ar3 (A4760)	KAT	CAST	On Demand	Manually
AES-CBC (A4769)	KAT	CAST	On Demand	Manually
AES-CBC (A4767)	KAT	CAST	On Demand	Manually
AES-CBC (A4760)	KAT	CAST	On Demand	Manually
AES-ECB (A4769)	KAT	CAST	On Demand	Manually
AES-ECB (A4767)	KAT	CAST	On Demand	Manually
AES-ECB (A4760)	KAT	CAST	On Demand	Manually
AES-CMAC (A4762)	KAT	CAST	On Demand	Manually
Safe Primes Key Generation (A4760)	PCT	PCT	On Demand	Manually
KDF SP800-108 (A4763)	KAT	CAST	On Demand	Manually
Hash DRBG (A4760)	KAT	CAST	On Demand	Manually
KDA HKDF Sp800-56Cr1 (A4759)	KAT	CAST	On Demand	Manually
KDF TLS (A4760)	KAT	CAST	On Demand	Manually
TLS v1.2 KDF RFC7627 (A4760)	KAT	CAST	On Demand	Manually
PBKDF (A4760)	KAT	CAST	On Demand	Manually
DSA SigVer (FIPS186-4) (A4760)	KAT	CAST	On Demand	Manually

Table 25: 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	The module must be restarted and successfully perform the pre-operational self-test and the CASTs to recover from these errors.	Module will not load
PCT Error	An error occurred during a PCT	PCT failure	The module must be restarted and successfully perform the PCT tests to recover from these errors.	Module stops functioning (sftk_fatalError is set to TRUE)

Table 26: Error States

In any error state, the output interface is inhibited, and the module accepts no more inputs or requests (as the module is no longer running).

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 part of the Oracle Linux 9 (OL9) RPM package in the form of nss-softokn-3.90.0-3.0.1.el9_2_fips and nss-softokn-freebl-3.90.0-3.0.1.el9_2_fips RPM packages that are located in the "Oracle Linux 9 Security Validation (Update 3)" yum repository (ol9_u3_security_validation).

The module can achieve the FIPS validated configuration by:

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

In both cases, the Crypto Officer must verify the Oracle Linux 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-3.90.0-3.0.1.el9_2_fips and nss-softokn-freebl-3.90.0-3.0.1.el9_2_fips 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:

Oracle Linux 9 NSS Cryptographic Module 4.35.0-381552536e763d0c

Alternatively, the /usr/lib64/nss/unsupported-tools/validation tool is provided as a convenience by the nss-tools-3.90.0-3.0.1.el9_2 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. 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.

11.2 Administrator Guidance

The Approved and non-Approved modes of operation are specified in section 2.4. The administrative functions are specified in the Approved Services table. All the logical interfaces are specified in section 3.1. The requirements and restrictions that shall be considered when operating the module in approved mode are specified in section 2.7 and section 6. The installation, initialization, and startup procedures specified in section 11.1 shall be followed.

11.3 Non-Administrator Guidance

There is no non-administrator guidance.

11.4 End of Life

As the module does not persistently store SSPs, secure sanitization of the module consists of unloading the module. This will zeroize all SSPs in volatile memory. Then, if desired, the nss-softokn-3.90.0-3.0.1.el9_2_fips and nss-softokn-freebl-3.90.0-3.0.1.el9_2_fips RPM packages can be uninstalled from the Oracle Linux 9 system.

12 Mitigation of Other Attacks

12.1 Attack List

Timing attacks on RSA:

RSA blinding: Timing attack on RSA was first demonstrated by Paul Kocher in 1996, who contributed the mitigation code to our module. Most recently Boneh and Brumley showed that RSA blinding is an effective defense against timing attacks on RSA.

Specific Limit: None

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

Cache invariant modular exponentiation: This is a variant of a modular exponentiation implementation that Colin Percival showed to defend against cache-timing attacks.

Specific Limit: This mechanism requires intimate knowledge of the cache line sizes of the processor. The mechanism may be ineffective when the module is running on a processor whose cache line sizes are unknown.

Arithmetic errors in RSA signatures:

Double-checking RSA signatures: Arithmetic errors in RSA signatures might leak the private key. Ferguson and Schneier recommend that every RSA signature generation should verify the signature just generated.

Specific Limit: None

Appendix A Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
API	Application Programming Interface
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CKG	Cryptographic Key Generation
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
GMAC	Galois Counter Mode Message Authentication Code
HKDF	HMAC-based Key Derivation Function
HMAC	Keyed-Hash Message Authentication Code
IV	Initialization Vector
KAT	Known Answer Test
KBKDF	Key-based Key Derivation Function
KDA	Key Derivation Algorithm
KDF	Key Derivation Function
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
PAA	Processor Algorithm Acceleration
PBKDF2	Password-based Key Derivation Function v2
PCT	Pairwise Consistency Test
PKCS	Public-Key Cryptography Standards
PSP	Public Security Parameter
RSA	Rivest, Shamir, Adleman
SHA	Secure Hash Algorithm
SSC	Shared Secret Computation
SSP	Sensitive Security Parameter
TOEPP	Tested Operational Environment’s Physical Perimeter

Appendix B References

FIPS 140-3	FIPS PUB 140-3 - Security Requirements For Cryptographic Modules March 2019 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf
FIPS 140-3 IG	Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program December 20, 2024 https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements
FIPS 180-4	Secure Hash Standard (SHS) March 2012 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS 186-4	Digital Signature Standard (DSS) July 2013 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
FIPS 186-5	Digital Signature Standard (DSS) February 2023 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf
FIPS 197	Advanced Encryption Standard November 2001 https://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS 198-1	The Keyed Hash Message Authentication Code (HMAC) July 2008 https://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
FIPS 202	SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions August 2015 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
PKCS#1	Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1 February 2003 https://www.ietf.org/rfc/rfc3447.txt
RFC 3526	More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE) May 2003 https://www.ietf.org/rfc/rfc3526.txt
RFC 5288	AES Galois Counter Mode (GCM) Cipher Suites for TLS August 2008 https://www.ietf.org/rfc/rfc5288.txt
RFC 7919	Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS) August 2016 https://www.ietf.org/rfc/rfc7919.txt
RFC 8446	The Transport Layer Security (TLS) Protocol Version 1.3 August 2018 https://www.ietf.org/rfc/rfc8446.txt
SP 800-38A	Recommendation for Block Cipher Modes of Operation Methods and Techniques December 2001 https://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
SP 800-38A Addendum	Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode October 2010 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a-add.pdf
SP 800-38B	Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication May 2005 https://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

SP 800-38F	Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping December 2012 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP 800-52r2	Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations August 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r2.pdf
SP 800-56Ar3	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography April 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf
SP 800-56Cr1	Recommendation for Key-Derivation Methods in Key-Establishment Schemes August 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr1.pdf
SP 800-56Cr2	Recommendation for Key-Derivation Methods in Key-Establishment Schemes August 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf
SP 800-90Ar1	Recommendation for Random Number Generation Using Deterministic Random Bit Generators June 2015 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP 800-90B	Recommendation for the Entropy Sources Used for Random Bit Generation January 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf
SP 800-108r1	NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions August 2022 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-108r1.pdf
SP 800-131Ar2	Transitioning the Use of Cryptographic Algorithms and Key Lengths March 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf
SP 800-132	Recommendation for Password-Based Key Derivation - Part 1: Storage Applications December 2010 https://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf
SP 800-133r2	Recommendation for Cryptographic Key Generation June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf
SP 800-135r1	Recommendation for Existing Application-Specific Key Derivation Functions December 2011 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf
SP 800-140Br1	CMVP Security Policy Requirements November 2023 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140Br1.pdf