

Ctrl IQ, Inc.

Rocky Linux 9 NSS Cryptographic Module

FIPS 140-3 Non-Proprietary Security Policy

Prepared by:

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

Document version: 1.0
Last update: 2026-09-18

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Table of Contents

1 General 5
1.1 Overview 5
1.2 Security Levels 5
1.3 Additional Information 5
2 Cryptographic Module Specification 7
2.1 Description 7
2.2 Tested and Vendor Affirmed Module Version and Identification 8
2.3 Excluded Components 8
2.4 Modes of Operation 8
2.5 Algorithms 9
2.6 Security Function Implementations 15
2.7 Algorithm Specific Information 20
2.7.1 AES-GCM IV 20
2.7.2 Key Derivation using SP 800-132 PBKDF2 21
2.7.3 SP 800-56A Rev. 3 Assurances 21
2.7.4 RSA Encapsulation and Decapsulation 22
2.7.5 Key Agreement 22
2.7.6 Key Encapsulation 22
2.8 RBG and Entropy 22
2.9 Key Generation 23
2.10 Key Establishment 23
2.11 Industry Protocols 23
3 Cryptographic Module Interfaces 25
3.1 Ports and Interfaces 25
4 Roles, Services, and Authentication 26
4.1 Authentication Methods 26
4.2 Roles 26
4.3 Approved Services 26
4.4 Non-Approved Services 38
4.5 External Software/Firmware Loaded 39
5 Software/Firmware Security 40

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

5.1 Integrity Techniques 40
5.2 Initiate on Demand 40
6 Operational Environment 41
6.1 Operational Environment Type and Requirements 41
6.2 Configuration Settings and Restrictions 41
7 Physical Security 42
8 Non-Invasive Security 43
9 Sensitive Security Parameters Management 44
9.1 Storage Areas 44
9.2 SSP Input-Output Methods 44
9.3 SSP Zeroization Methods 45
9.4 SSPs 45
10 Self-Tests 61
10.1 Pre-Operational Self-Tests 61
10.2 Conditional Self-Tests 61
10.3 Periodic Self-Test Information 68
10.4 Error States 70
10.5 Operator Initiation of Self-Tests 70
11 Life-Cycle Assurance 71
11.1 Installation, Initialization, and Startup Procedures 71
11.2 Administrator Guidance 71
11.3 Non-Administrator Guidance 71
11.4 End of Life 71
12 Mitigation of Other Attacks 73
12.1 Attack List 73
Appendix A Glossary and Abbreviations 74
Appendix B References 76

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

List of Tables

Table 1: Security Levels 5
Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets) 8
Table 3: Tested Operational Environments - Software, Firmware, Hybrid 8
Table 4: Modes List and Description 9
Table 5: Approved Algorithms 13
Table 6: Vendor-Affirmed Algorithms 13
Table 7: Non-Approved, Not Allowed Algorithms 14
Table 8: Security Function Implementations 20
Table 9: Entropy Certificates 22
Table 10: Entropy Sources 23
Table 11: Ports and Interfaces 25
Table 12: Roles 26
Table 13: Approved Services 37
Table 14: Non-Approved Services 39
Table 15: Storage Areas 44
Table 16: SSP Input-Output Methods 45
Table 17: SSP Zeroization Methods 45
Table 18: SSP Table 1 52
Table 19: SSP Table 2 60
Table 20: Pre-Operational Self-Tests 61
Table 21: Conditional Self-Tests 67
Table 22: Pre-Operational Periodic Information 68
Table 23: Conditional Periodic Information 69
Table 24: Error States 70

List of Figures

Figure 1: Block Diagram 7

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for version rocky9.20251215 of the Rocky Linux 9 NSS Cryptographic Module. It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-3 (Federal Information Processing Standards Publication 140-3) for an overall Security Level 1 module.

This Non-Proprietary Security Policy may be reproduced and distributed, but only whole and intact and including this notice. Other documentation is proprietary to their authors.

1.2 Security Levels

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

Table 1: Security Levels

1.3 Additional Information

In preparing the Security Policy document, the laboratory formatted the vendor-supplied documentation for consolidation without altering the technical statements therein contained. The further refining of the Security

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

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.

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The Rocky Linux 9 NSS Cryptographic Module (hereafter referred to as "the module") is defined as a software module in a multi-chip standalone embodiment. It provides a C language application program interface (API) designed to support cross-platform development of security-enabled client and server applications. Applications built with NSS can support SSLv3, TLS, IKEv2, PKCS#5, PKCS#7, PKCS#11, PKCS#12, S/MIME, X.509 v3 certificates, and other security standards supporting FIPS 140-3 validated cryptographic algorithms. It combines a vertical stack of Linux components intended to limit the external interface each separate component may provide.

Module Type: Software

Module Embodiment: Multi-Chip Standalone

Cryptographic Boundary:

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

Tested Operational Environment's Physical Perimeter (TOEPP):

The TOEPP of the module is defined as the general-purpose computer on which the module is installed.

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

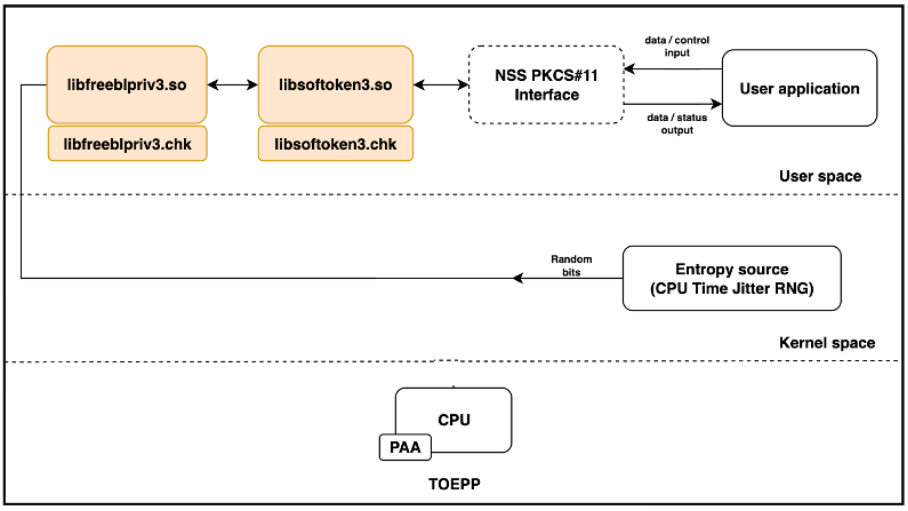


Figure 1: Block Diagram

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Hardware:

N/A for this module.

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libsoftokn3.so, libfreeblpriv3.so	rocky9.20251215	N/A	HMAC-SHA-256

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

Tested Module Identification – Hybrid Disjoint Hardware:

N/A for this module.

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Rocky Linux 9	SuperMicro SuperServer 5039MS	Intel Kaby Lake Xeon E3-1270 v6	Yes	N/A	rocky9.20251215
Rocky Linux 9	SuperMicro SuperServer 5039MS	Intel Kaby Lake Xeon E3-1270 v6	No	N/A	rocky9.20251215

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

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

N/A for this module.

2.3 Excluded Components

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

2.4 Modes of Operation

Modes List and Description:

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Mode Name	Description	Types of Services	Status Indicator
Approved	Automatically entered whenever an approved service is requested.	Approved	Equivalent to the indicator of the requested service (for all services NSC_NSSGetFIPSStatus returns CKS_NSS_FIPS_OK (1); for DRBG, the service API returns CKR_OK).
Non-Approved	Automatically entered whenever a non-approved service is requested.	Non-Approved	Equivalent to the indicator of the requested service (for all services NSC_NSSGetFIPSStatus returns CKS_NSS_FIPS_NOT_OK; for DRBG, the service API returns a value different from CKR_OK).

Table 4: Modes List and Description

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

Mode Change Instructions and Status:

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

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A7786, A7788	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC-CS1	A7786, A7788	Direction - decrypt, encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A7786, A7788	Direction - Generation Key Length - 128, 192, 256	SP 800-38B
AES-CTR	A7786, A7788	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A7786, A7788	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	CAVP Cert	Properties	Reference
AES-GCM	A7786, A7788, A7789	Direction - Decrypt, Encrypt IV Generation - External, Internal Key Length - 128, 192, 256 IV Generation Mode - 8.2.1, 8.2.2	SP 800-38D
AES-KW	A7786, A7788	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KWP	A7786, A7788	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
ECDSA KeyGen (FIPS186-5)	A7786	Curve - P-256, P-384, P-521 Secret Generation Mode - testing candidates	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A7786	Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512 Component - No	FIPS 186-5
ECDSA SigVer (FIPS186-5)	A7786	Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-5
Hash DRBG	A7786	Prediction Resistance - No, Yes Mode - SHA2-256	SP 800-90A Rev. 1
HMAC-SHA2-224	A7786, A7790	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A7786, A7790	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A7786, A7790	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A7786, A7790	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A7786	Domain Parameter Generation Methods - P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	CAVP Cert	Properties	Reference
KAS-FFC-SSC Sp800-56Ar3	A7786	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 SP800-56Cr2	A7785	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-8192 Increment 8 HMAC Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	SP 800-56C Rev. 2
KDF IKEv2 (CVL)	A7787	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 - SHA2-224, SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
KDF SP800-108	A7786	KDF Mode - Counter, Double Pipeline Iteration, Feedback Supported Lengths - Supported Lengths: 112-4096 Increment 8	SP 800-108 Rev. 1
KTS-IFC	A7786	Modulo - 2048, 3072, 4096, 6144, 8192 Key Generation Methods - rsakpg1-basic Scheme - KTS-OAEP-basic - KAS Role - initiator, responder Key Transport Method - Key Length - 512	SP 800-56B Rev. 2
ML-DSA KeyGen	A7786	Parameter Sets - ML-DSA-44, ML-DSA-65, ML-DSA-87	FIPS 204
ML-DSA SigGen	A7786	Signature Interfaces - external Pre Hash - pure Deterministic - No, Yes External Mu - No Parameter Sets - ML-DSA-44, ML-DSA-65, ML-DSA-87 Message Length - Message Length: 8-65536 Increment 8 Context Length - Context Length: 0-248 Increment 8	FIPS 204

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	CAVP Cert	Properties	Reference
ML-DSA SigVer	A7786	Signature Interfaces - external Pre Hash - pure External Mu - No Parameter Sets - ML-DSA-44, ML-DSA-65, ML-DSA-87 Message Length - Message Length: 8-65536 Increment 8 Context Length - Context Length: 0-248 Increment 8	FIPS 204
ML-KEM EncapDecap	A7786	Parameter Sets - ML-KEM-1024, ML-KEM-768 Functions - Decapsulation, Decapsulation Key Check, Encapsulation, Encapsulation Key Check	FIPS 203
ML-KEM KeyGen	A7786	Parameter Sets - ML-KEM-1024, ML-KEM-768	FIPS 203
PBKDF	A7786	Iteration Count - Iteration Count: 1000-10000 Increment 1 Password Length - Password Length: 8-128 Increment 1	SP 800-132
RSA KeyGen (FIPS186-5)	A7786	Key Generation Mode - probable Modulo - 2048, 3072, 4096, 6144, 8192 Primality Tests - 2pow100 Private Key Format - standard	FIPS 186-5
RSA SigGen (FIPS186-5)	A7786	Hash Pair - Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512 Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
RSA SigVer (FIPS186-5)	A7786	Hash Pair - Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512 Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
Safe Primes Key Generation	A7786	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	A7786, A7790	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A7786, A7790	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	CAVP Cert	Properties	Reference
SHA2-384	A7786, A7790	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A7786, A7790	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA3-256	A8940	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-512	A8940	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHAKE-128	A8940, A8941	Message Length - Message Length: 0-65536 Increment 8 Output Length - Output Length: 16-65536 Increment 8	FIPS 202
SHAKE-256	A8940, A8941	Message Length - Message Length: 0-65536 Increment 8 Output Length - Output Length: 16-65536 Increment 8	FIPS 202
TLS v1.2 KDF RFC7627 (CVL)	A7786	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1

Table 5: Approved Algorithms

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

Vendor-Affirmed Algorithms:

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

Table 6: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

Non-Approved, Not Allowed Algorithms:

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Use and Function
MD2, MD5, SHA-1	Message digest
RC2, RC4, DES, Triple-DES, CDMF, Camellia, SEED, ChaCha20(-Poly1305)	Encryption, Decryption
AES GCM (external IV)	Encryption
CBC-MAC, AES XCBC-MAC, AES XCBC-MAC-96, HMAC (MD2, MD5, SHA-1), HMAC (< 112-bit keys), HMAC/SSLv3 MAC (constant-time implementation)	Message authentication
KDF using only MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, DES, Triple-DES, AES, Camellia, SEED	Key derivation
ANS X9.63 KDF, SSL 3 PRF, IKEv1 PRF, TLS 1.0/1.1 KDF, TLS KDF without extended master secret	Key derivation
KBKDF, HKDF, TLS 1.2 KDF, IKEv2 PRF (< 112-bit keys)	Key derivation
KBKDF and IKEv2 PRF (MD2, MD5)	Key derivation
PBKDF2 with parameters (password, salt, iterations) not complying with section 2.7.2 or < 112-bit keys, PKCS#5 PBE, PKCS#12 PBE	Password-based key derivation
J-PAKE, KAS-FFC-SSC with FIPS 186-type groups, X25519	Shared secret computation
RSA (data encryption / decryption)	Asymmetric encryption, Asymmetric decryption
DSA, RSA (primitive), RSA (PKCS#1 v1.5 or PSS with MD2, MD5, SHA-1), RSA (< 2048-bit keys), ECDSA (component)	Signature generation, Signature verification
DSA	Parameter generation, Parameter verification
DSA, Diffie-Hellman (FIPS 186-type groups), RSA (< 2048 bits), RSA (> 4096 bits), Ed25519, X25519	Key pair generation
Symmetric key generation (< 112 bits)	Secret key generation

Table 7: Non-Approved, Not Allowed Algorithms

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Encryption with AES	BC-UnAuth	Encryption using AES		AES-CBC: (A7786, A7788) AES-CBC-CS1: (A7786, A7788) AES-CTR: (A7786, A7788) AES-ECB: (A7786, A7788)
Decryption with AES	BC-UnAuth	Decryption using AES		AES-CBC: (A7786, A7788) AES-CBC-CS1: (A7786, A7788) AES-CTR: (A7786, A7788) AES-ECB: (A7786, A7788)
Authenticated Encryption with AES	BC-Auth	Authenticated encryption using AES		AES-GCM: (A7786, A7788, A7789) AES-KW: (A7786, A7788) AES-KWP: (A7786, A7788)
Authenticated Decryption with AES	BC-Auth	Authenticated decryption using AES		AES-GCM: (A7786, A7788, A7789) AES-KW: (A7786, A7788) AES-KWP: (A7786, A7788)
Key Derivation with PBKDF2	PBKDF	Key derivation using PBKDF2		PBKDF: (A7786)
Key Derivation with KBKDF	KBKDF	Key derivation using KBKDF		KDF SP800-108: (A7786)
Key Derivation with HKDF	KAS-56CKDF	Key derivation using HKDF		KDA HKDF SP800-56Cr2: (A7785)

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Type	Description	Properties	Algorithms
Key Derivation with TLS 1.2 KDF	KAS-135KDF	Key derivation using TLS 1.2 KDF		TLS v1.2 KDF RFC7627: (A7786)
Key Derivation with IKEv2 KDF	KAS-135KDF	Key derivation using IKEv2 KDF		KDF IKEv2: (A7787)
Key Wrapping with AES (KTS)	KTS-Wrap	Key wrapping using AES	Standard:SP 800-38F IG D.G:approved key wrapping method Caveat:Key establishment methodology provides between 128 and 256 bits of security strength	AES-KW: (A7786, A7788) AES-KWP: (A7786, A7788)
Key Unwrapping with AES (KTS)	KTS-Unwrap	Key unwrapping using AES	Standard:SP 800-38F or SP 800-38D IG D.G:approved key wrapping method Caveat:Key establishment methodology provides between 128 and 256 bits of security strength	AES-KW: (A7786, A7788) AES-KWP: (A7786, A7788) AES-GCM: (A7786, A7788, A7789)
Message Authentication with HMAC	MAC	Message authentication using HMAC		HMAC-SHA2-224: (A7786, A7790) HMAC-SHA2-256: (A7786, A7790) HMAC-SHA2-384: (A7786, A7790) HMAC-SHA2-512: (A7786, A7790)
Message Authentication with CMAC	MAC	Message authentication using CMAC		AES-CMAC: (A7786, A7788)
Random Number Generation with Hash_DRBG	DRBG	Random number generation using Hash_DRBG		Hash DRBG: (A7786)

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Type	Description	Properties	Algorithms
Shared Secret Computation with KAS-ECC-SSC	KAS-SSC	Shared secret computation using KAS-ECC-SSC	Compliance:FIPS 140-3 IG D.F scenario 2(1)	KAS-ECC-SSC Sp800-56Ar3: (A7786)
Shared Secret Computation with KAS-FFC-SSC	KAS-SSC	Shared secret computation using KAS-FFC-SSC	Compliance:FIPS 140-3 IG D.F scenario 2(1)	KAS-FFC-SSC Sp800-56Ar3: (A7786)
Signature Generation with RSA	DigSig-SigGen	Signature generation using RSA		RSA SigGen (FIPS186-5): (A7786)
Signature Generation with ECDSA	DigSig-SigGen	Signature generation using ECDSA		ECDSA SigGen (FIPS186-5): (A7786)
Signature Verification with RSA	DigSig-SigVer	Signature verification using RSA		RSA SigVer (FIPS186-5): (A7786)
Signature Verification with ECDSA	DigSig-SigVer	Signature verification using ECDSA		ECDSA SigVer (FIPS186-5): (A7786)
Symmetric Key Generation with Hash_DRBG	CKG	Direct symmetric key generation using Hash_DRBG		Hash DRBG: (A7786) Symmetric Cryptographic Key Generation (CKG): ()
Key Pair Generation with RSA	AsymKeyPair-KeyGen CKG	Key pair generation using RSA		RSA KeyGen (FIPS186-5): (A7786) Asymmetric Cryptographic Key Generation (CKG): ()
Key Pair Generation with ECDSA	AsymKeyPair-KeyGen CKG	Key pair generation using ECDSA		ECDSA KeyGen (FIPS186-5): (A7786) Asymmetric Cryptographic Key

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Type	Description	Properties	Algorithms
				Generation (CKG): ()
Key Pair Generation with Safe Primes	AsymKeyPair-KeyGen CKG	Key pair generation using Safe Primes		Safe Primes Key Generation: (A7786) Asymmetric Cryptographic Key Generation (CKG): ()
Message Digest with SHA	SHA	Message digest using SHA		SHA2-224: (A7786, A7790) SHA2-256: (A7786, A7790) SHA2-384: (A7786, A7790) SHA2-512: (A7786, A7790)
Key Encapsulation with RSA (KTS)	KTS-Encap	Key Encapsulation using RSA	Standard:SP 800-56B Rev. 2 IG D.G:approved key encapsulation method Caveat:Key establishment methodology provides between 112 and 256 bits of security strength	KTS-IFC: (A7786)
Key Decapsulation with RSA (KTS)	KTS-Decap	Key Decapsulation using RSA	Standard:SP 800-56B Rev. 2 IG D.G:approved key encapsulation method Caveat:Key establishment methodology provides between 112 and 256 bits of security strength	KTS-IFC: (A7786)
Key Pair Generation with ML-KEM	AsymKeyPair-KeyGen CKG	Key pair generation using ML-KEM		ML-KEM KeyGen: (A7786) Asymmetric Cryptographic Key Generation (CKG): () SHA3-256:

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Type	Description	Properties	Algorithms
				(A8940) SHA3-512: (A8940) SHAKE-128: (A8940) SHAKE-256: (A8940)
Key Encapsulation with ML-KEM	KEM-Encap	Key encapsulation using ML-KEM	IG:IG D.S Scenario 1 Caveat:Key establishment methodology provides between 192 and 256 bits of security strength	ML-KEM EncapDecap: (A7786) SHA3-256: (A8940) SHA3-512: (A8940) SHAKE-128: (A8940) SHAKE-256: (A8940)
Key Decapsulation with ML-KEM	KEM-Decap	Key decapsulation using ML-KEM	IG:IG D.S Scenario 1 Caveat:Key establishment methodology provides between 192 and 256 bits of security strength	ML-KEM EncapDecap: (A7786) SHA3-256: (A8940) SHA3-512: (A8940) SHAKE-128: (A8940) SHAKE-256: (A8940)
Key Pair Generation with ML-DSA	AsymKeyPair-KeyGen CKG	Key pair generation using ML-DSA		ML-DSA KeyGen: (A7786) Asymmetric Cryptographic Key Generation (CKG): () SHAKE-128: (A8941) SHAKE-256: (A8941)

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Type	Description	Properties	Algorithms
Signature Generation with ML-DSA	DigSig-SigGen	Signature generation using ML-DSA		ML-DSA SigGen: (A7786) SHAKE-128: (A8941) SHAKE-256: (A8941)
Signature Verification with ML-DSA	DigSig-SigVer	Signature verification using ML-DSA		ML-DSA SigVer: (A7786) SHAKE-128: (A8941) SHAKE-256: (A8941)

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 AES-GCM IV

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

For TLS 1.2, the module offers the AES-GCM implementation and uses the context of Scenario 1 of FIPS 140-3 IG C.H. NSS is compliant with SP 800-52 Rev. 2 Section 3.3.1 and the mechanism for IV generation is compliant with RFC 5288.

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

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

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

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

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Finally, for TLS 1.3, the module offers the AES-GCM implementation and uses the context of Scenario 1 of FIPS 140-3 IG C.H. The protocol that provides this compliance is TLS 1.3, defined in RFC 8446 of August 2018, using the cipher-suites that explicitly select AES-GCM as the encryption/decryption cipher (Appendix B.4 of RFC 8446). The module supports acceptable AES-GCM cipher suites from Section 3.3.1 of SP800-52 Rev. 2.

TLS 1.3 employs separate 64-bit sequence numbers, one for protocol records that are received, and one for protocol records that are sent to a peer. These sequence numbers are set at zero at the beginning of a TLS 1.3 connection and each time when the AES-GCM key is changed. After reading or writing a record, the respective sequence number is incremented by one. The protocol specification determines that the sequence number should not wrap, and if this condition is observed, then the protocol implementation must either trigger a re-key of the session (i.e., a new key for AES-GCM), or terminate the connection.

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

2.7.2 Key Derivation using SP 800-132 PBKDF2

The module provides password-based key derivation (PBKDF2), compliant with SP 800-132. The module supports option 1a from Section 5.4 of SP 800-132, in which the Master Key (MK) or a segment of it is used directly as the Data Protection Key (DPK). In accordance to SP 800-132 and FIPS 140-3 IG D.N, the following requirements shall be met:

- Derived keys shall only be used in storage applications. The MK shall not be used for other purposes. The module enforces the length of the MK or DPK to be of 112 bits or more for the service to be approved.
- Passwords or passphrases, used as an input for the PBKDF2, shall not be used as cryptographic keys.
- The minimum length of the password or passphrase accepted by the module is 8 characters. The probability of guessing the value is estimated to be at most $1/62^8 = 4 \times 10^{-15}$, when the password is a combination of lowercase, uppercase, and numeric characters. If the password solely consists of digits, the probability of guessing the value is estimated to be 10^{-8}. Combined with the minimum iteration count as described below, this provides an acceptable trade-off between user experience and security against brute-force attacks.
- A portion of the salt shall be generated randomly using the SP 800-90A Rev. 1 DRBG provided by the module. The module restricts minimum length to 128 bits.
- The iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users. The module only allows minimum iteration count to be 1000.

2.7.3 SP 800-56A Rev. 3 Assurances

To comply with the assurances found in Section 5.6.2 of SP 800-56A Rev. 3, the operator must use the module together with an application that implements the TLS protocol. Additionally, the module's approved Key Pair Generation service (see Section 4.3) must be used to generate ephemeral FFC or ECC 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 800-56A Rev. 3.

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

2.7.4 RSA Encapsulation and Decapsulation

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

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

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

2.7.5 Key Agreement

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

2.7.6 Key Encapsulation

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

When using an externally obtained ML-KEM static public encapsulation key, the operator shall ensure that the ownership of the private key has been established prior to invoking the ML-KEM encapsulation service. This assurance may be obtained using a trusted third party (e.g., a certificate authority) or another mechanism appropriate for the deployment, consistent with the guidance of SP 800-227 Section 4.5.

The module shall be deployed on an appropriately secured device and operational environment that protects sensitive KEM data from unauthorized access or modification, in accordance with SP 800-227 Section 4.2, RM4.

2.8 RBG and Entropy

Cert Number	Vendor Name
E210	Ctrl IQ, Inc.

Table 9: Entropy Certificates

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Rocky Linux Userspace CPU Time Jitter RNG Entropy Source	Non-Physical	Rocky Linux 9 on Intel Xeon E3-1270 v6	256 bits	256 bits	SHA3-256 (A5837)

Table 10: Entropy Sources

The module employs a Deterministic Random Bit Generator (DRBG) implementation based on SP 800-90A Rev. 1. This DRBG is used internally by the module (e.g. to generate symmetric keys, seeds for asymmetric key pairs, and random numbers for security functions). It can also be accessed using the specified API functions.

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

The module complies with the Public Use Document for ESV certificate E210 by reading entropy data from the get_random() function with the GRND_RANDOM flag set, which corresponds to the GetEntropy() conceptual interface. This function outputs 256 bits of full entropy.

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

The operational environment on the ESV certificate is identical to the operating system described in this document, which implements the entropy source (outside the module's cryptographic boundary). Thus, the module is compliant with scenario 1 (b) of IG 9.3.A. There are no maintenance requirements for the entropy source.

2.9 Key Generation

The module implements Cryptographic Key Generation (CKG, vendor affirmed), compliant with SP 800-133 Rev. 2 as listed in the Security Function Implementation table in Section 2.6.

When random values are required, they are obtained from the SP 800-90A Rev. 1 approved DRBG, compliant with Section 4 of SP 800-133 Rev. 2.

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

2.10 Key Establishment

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

2.11 Industry Protocols

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

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

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

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

Table 11: Ports and Interfaces

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

4 Roles, Services, and Authentication

4.1 Authentication Methods

The module does not support authentication for roles.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 12: Roles

No support is provided for multiple concurrent operators or a maintenance role.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Encryption	Encrypt a plaintext	CKS_NSS_FIPS_OK (1)	AES key, IV, plaintext	Ciphertext	Encryption with AES	Crypto Officer - AES Key: W,E
Decryption	Decrypt a ciphertext	CKS_NSS_FIPS_OK (1)	AES key, IV, ciphertext	Plaintext	Decryption with AES	Crypto Officer - AES Key: W,E
Authenticated Encryption	Encrypt a plaintext	CKS_NSS_FIPS_OK (1)	AES key, IV, plaintext	Ciphertext, MAC tag	Authenticated Encryption with AES	Crypto Officer - AES Key: W,E
Authenticated Decryption	Decrypt a ciphertext	CKS_NSS_FIPS_OK (1)	AES key, IV, MAC tag, ciphertext	Plaintext or fail	Authenticated Decryption with AES	Crypto Officer - AES Key: W,E
Key Encapsulation with RSA	Encapsulate a key with RSA	CKS_NSS_FIPS_OK (1)	RSA public key, any CSP (other than Password)	Encapsulated CSP	Key Encapsulation with RSA (KTS)	Crypto Officer - RSA Public Key: W,E

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
			than Password)		with AES (KTS)	- AES Key: W,E
Key Unwrapping	Unwrap a CSP	CKS_NSS_FIPS_OK (1)	AES Key-Decryption Key, Wrapped CSP	Any CSP (other than Password)	Key Unwrapping with AES (KTS)	Crypto Officer - AES Key: W,E
ML-KEM Encapsulation	Encapsulation using ML-KEM	CKS_NSS_FIPS_OK (1)	ML-KEM encapsulation key	Ciphertext, ML-KEM shared secret	Key Encapsulation with ML-KEM	Crypto Officer - ML-KEM Encapsulation Key: W,E - ML-KEM Shared Secret: G,R
ML-KEM Decapsulation	Decapsulation using ML-KEM	CKS_NSS_FIPS_OK (1)	ML-KEM decapsulation key, ciphertext	ML-KEM shared secret	Key Decapsulation with ML-KEM	Crypto Officer - ML-KEM Decapsulation Key: W,E - ML-KEM Shared Secret: G,R
Message Authentication with HMAC	Compute a MAC tag	CKS_NSS_FIPS_OK (1)	HMAC key, message	MAC tag	Message Authentication with HMAC	Crypto Officer - HMAC Key: W,E
Message Authentication with CMAC	Compute a MAC tag	CKS_NSS_FIPS_OK (1)	AES key, message	MAC tag	Message Authentication with CMAC	Crypto Officer - AES Key: W,E
Message Digest	Compute a message digest	CKS_NSS_FIPS_OK (1)	Message	Digest value	Message Digest with SHA	Crypto Officer

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Random Number Generation	Generate random bytes	CKR_OK	Output length	Random bytes	Random Number Generation with Hash_DRBG	Crypto Officer - Entropy Input: W,E,Z - DRBG Seed: G,E,Z - Internal State (V, C): G,W,E
KAS-FFC-SSC Shared Secret Computation	Compute a shared secret	CKS_NSS_FIPS_OK (1)	FFC private key (owner), FFC public key (peer)	Shared secret	Shared Secret Computation with KAS-FFC-SSC	Crypto Officer - FFC Private Key: W,E - FFC Public Key: W,E - Shared Secret: G
KAS-ECC-SSC Shared Secret Computation	Compute a shared secret	CKS_NSS_FIPS_OK (1)	EC private key (owner), EC public key (peer)	Shared secret	Shared Secret Computation with KAS-ECC-SSC	Crypto Officer - EC Private Key: W,E - EC Public Key: W,E - Shared Secret: G
Signature Generation with RSA	Generate a signature	CKS_NSS_FIPS_OK (1)	RSA private key, message	Signature	Signature Generation with RSA	Crypto Officer - RSA Private Key: W,E
Signature Generation with ECDSA	Generate a signature	CKS_NSS_FIPS_OK (1)	EC private key, message	Signature	Signature Generation with ECDSA	Crypto Officer - EC Private Key: W,E
Signature Generation	Generate a signature	CKS_NSS_FIPS_OK (1)	ML-DSA private key,	Signature	Signature Generation	Crypto Officer

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Key: G - Module-generated RSA Public Key: G - Intermediate key generation value: G,E,Z
Key Pair Generation with ECDSA	Generate a key pair	CKS_NSS_FIPS_OK (1)	Curve	EC public key, EC private key	Key Pair Generation with ECDSA	Crypto Officer - Module-generated EC Private Key: G - Module-generated EC Public Key: G - Intermediate key generation value: G,E,Z
Key Pair Generation with ML-DSA	Generate a key pair	CKS_NSS_FIPS_OK (1)	Parameter set	ML-DSA public key, ML-DSA private key	Key Pair Generation with ML-DSA	Crypto Officer - Module-generated ML-DSA Private Key: G - Module-generated ML-DSA Public Key: G - ML-DSA Seed: G,E

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Key Pair Generation with ML-KEM	Generate a key pair	CKS_NSS_FIPS_OK (1)	Parameter set	ML-KEM Private Key, ML-KEM Public Key	Key Pair Generation with ML-KEM	Crypto Officer - Module-generated ML-KEM Encapsulation Key: G - Module-generated ML-KEM Decapsulation Key: G - ML-KEM Seed: G,E
Symmetric Key Generation	Generate a secret key	CKS_NSS_FIPS_OK (1)	Key size	Symmetric Key	Symmetric Key Generation with Hash_DRBG	Crypto Officer - Symmetric Key: G
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	Message Digest with SHA Message Authentication with HMAC Encryption with AES Decryption with AES Authenticated Encryption	Crypto Officer

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					with ECDSA Signature Verification with ECDSA Key Pair Generation with ML-KEM Key Pair Generation with ML-DSA Key Encapsulation with RSA (KTS) Key Decapsulation with RSA (KTS) Key Encapsulation with ML-KEM Key Decapsulation with ML-KEM Signature Generation with ML-DSA Signature Verification with ML-DSA	
Zeroization	Zeroize all SSPs	N/A	Any SSP	None	None	Crypto Officer - Symmetric Key: Z - AES Key: Z

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- HMAC Key: Z - Key-Derivation Key: Z - Shared Secret: Z - Password: Z - KBKDF Derived Key: Z - PBKDF2 Derived Key: Z - HKDF Derived Key: Z - TLS Derived Key: Z - IKE Derived Key: Z - Entropy Input: Z - DRBG Seed: Z - Internal State (V, C): Z - Module-generated FFC Private Key: Z - Module-generated FFC Public Key: Z - FFC Private Key: Z - FFC Public Key:

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Z - Module-generated EC Private Key: Z - Module-generated EC Public Key: Z - EC Private Key: Z - EC Public Key: Z - Module-generated RSA Private Key: Z - Module-generated RSA Public Key: Z - RSA Private Key: Z - RSA Public Key: Z - Module-generated ML-DSA Private Key: Z - Module-generated ML-DSA Public Key: Z - ML-DSA Private Key: Z - ML-DSA Public Key: Z - ML-DSA

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Seed: Z - Module-generated ML-KEM Encapsulation Key: Z - Module-generated ML-KEM Decapsulation Key: Z - ML-KEM Encapsulation Key: Z - ML-KEM Decapsulation Key: Z - ML-KEM Shared Secret: Z - ML-KEM Seed: Z - Intermediate key generation value: Z

Table 13: Approved Services

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

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

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

1. The session indicator, which must be used for all cryptographic services except the key (pair) generation and key derivation services. It can be accessed by invoking the NSC_NSSGetFIPSStatus

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

function with the CKT_NSS_SESSION_LAST_CHECK parameter. If the output parameter is set to CKS_NSS_FIPS_OK (1), the service was approved.

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

4.4 Non-Approved Services

Name	Description	Algorithms	Role
Message Digest	Compute a message digest	MD2, MD5, SHA-1	CO
Encryption	Encrypt a plaintext	RC2, RC4, DES, Triple-DES, CDMF, Camellia, SEED, ChaCha20(-Poly1305) AES GCM (external IV)	CO
Decryption	Decrypt a ciphertext	RC2, RC4, DES, Triple-DES, CDMF, Camellia, SEED, ChaCha20(-Poly1305)	CO
Message Authentication	Compute a MAC tag	CBC-MAC, AES XCBC-MAC, AES XCBC-MAC-96, HMAC (MD2, MD5, SHA-1), HMAC (< 112-bit keys), HMAC/SSLv3 MAC (constant-time implementation)	CO
Key Derivation	Derive a key from a key-derivation key or a shared secret	KDF using only MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, DES, Triple-DES, AES, Camellia, SEED ANS X9.63 KDF, SSL 3 PRF, IKEv1 PRF, TLS 1.0/1.1 KDF, TLS KDF without extended master secret KBKDF, HKDF, TLS 1.2 KDF, IKEv2 PRF (< 112-bit keys) KBKDF and IKEv2 PRF (MD2, MD5)	CO
Password-Based Key Derivation	Derive a key from a password	PBKDF2 with parameters (password, salt, iterations) not complying with section 2.7.2 or < 112-bit keys, PKCS#5 PBE, PKCS#12 PBE	CO
Shared Secret Computation	Compute a shared secret	J-PAKE, KAS-FFC-SSC with FIPS 186-type groups, X25519	CO

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Algorithms	Role
Signature Generation	Generate a signature	DSA, RSA (primitive), RSA (PKCS#1 v1.5 or PSS with MD2, MD5, SHA-1), RSA (< 2048-bit keys), ECDSA (component)	CO
Signature Verification	Verify a signature	DSA, RSA (primitive), RSA (PKCS#1 v1.5 or PSS with MD2, MD5, SHA-1), RSA (< 2048-bit keys), ECDSA (component)	CO
Asymmetric Encryption	Encrypt a plaintext	RSA (data encryption / decryption)	CO
Asymmetric Decryption	Decrypt a plaintext	RSA (data encryption / decryption)	CO
Parameter Generation	Generate domain parameters	DSA	CO
Parameter Verification	Verify domain parameters	DSA	CO
Key Pair Generation	Generate a key pair	DSA, Diffie-Hellman (FIPS 186-type groups), RSA (< 2048 bits), RSA (> 4096 bits), Ed25519, X25519	CO
Secret Key Generation	Generate a secret key	Symmetric key generation (< 112 bits)	CO

Table 14: Non-Approved Services

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

4.5 External Software/Firmware Loaded

The module does not load external software or firmware.

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

5 Software/Firmware Security

5.1 Integrity Techniques

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

The CAST for the HMAC-SHA2-256 algorithm used for the integrity check of the module's components is run before the execution of the integrity check itself.

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.

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

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

6.2 Configuration Settings and Restrictions

The module shall be installed as stated in Section 11.1.

Instrumentation tools like the ptrace system call, gdb and strace, userspace live patching, as well as other tracing mechanisms offered by the Linux environment such as ftrace or systemtap, shall not be used in the operational environment. The use of any of these tools implies that the cryptographic module is running in a non-validated operational environment.

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

7 Physical Security

The module is comprised of software only and therefore this Section is not applicable.

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

8 Non-Invasive Security

This module does not implement any non-invasive security mechanism and therefore this Section is not applicable.

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

9 Sensitive Security Parameters Management

9.1 Storage Areas

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

Table 15: Storage Areas

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

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) - AES	Calling application within TOEPP	Cryptographic module	Encrypted	Manual	Electronic	Key Unwrapping with AES (KTS)
API input parameters (encrypted) - RSA	Calling application within TOEPP	Cryptographic module	Encrypted	Manual	Electronic	Key Decapsulation with RSA (KTS)
API output parameters (plaintext)	Cryptographic module	Calling application within TOEPP	Plaintext	Manual	Electronic	
API output parameters (encrypted) - AES	Cryptographic module	Calling application within TOEPP	Encrypted	Manual	Electronic	Key Wrapping with AES (KTS)

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
API output parameters (encrypted) - RSA	Cryptographic module	Calling application within TOEPP	Encrypted	Manual	Electronic	Key Encapsulation with RSA (KTS)

Table 16: SSP Input-Output Methods

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

9.3 SSP Zeroization Methods

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

Table 17: SSP Zeroization Methods

All data output is inhibited during zeroization.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Symmetric Key	Symmetric key generated	112-512 bits -	Symmetric key - CSP	Symmetric Key		Symmetric Key

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	by the module	112-256 bits		Generation with Hash_DRBG		Generation with Hash_DRBG
AES Key	AES Key used for encryption, decryption, and for message authentication	128, 192, 256 bits - 128, 192, 256 bits	Symmetric key - CSP			Encryption with AES Decryption with AES Authenticated Encryption with AES Authenticated Decryption with AES Message Authentication with CMAC Key Wrapping with AES (KTS) Key Unwrapping with AES (KTS)
HMAC Key	HMAC key used for message authentication	112-512 bits - 112-256 bits	Symmetric key - CSP			Message Authentication with HMAC
Key-Derivation Key	Symmetric key used to derive symmetric keys	112-4096 bits - 112-256 bits	Symmetric key - CSP	Symmetric Key Generation with Hash_DRBG		Key Derivation with KBKDF
Shared Secret	Shared secret generated by KAS-ECC-SSC or KAS-FFC-SSC	256-8192 bits - 112-256 bits	Shared secret - CSP		Shared Secret Computation with KAS-FFC-SSC Shared Secret	Key Derivation with HKDF Key Derivation

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
					Computation with KAS-ECC-SSC	with TLS 1.2 KDF Key Derivation with IKEv2 KDF
Password	Password used to derive symmetric keys	8-128 characters - N/A	Password - CSP			Key Derivation with PBKDF2
KBKDF Derived Key	Symmetric key derived from a key-derivation key	112-4096 bits - 112-256 bits	Symmetric key - CSP	Key Derivation with KBKDF		
PBKDF2 Derived Key	Symmetric key derived from a password	112-4096 bits - 112-256 bits	Symmetric key - CSP	Key Derivation with PBKDF2		
HKDF Derived Key	Symmetric key derived from a shared secret using HKDF	112-4096 bits - 112-256 bits	Symmetric key - CSP	Key Derivation with HKDF		
TLS Derived Key	Symmetric key derived from a shared secret using TLS 1.2 KDF	112-4096 bits - 112-256 bits	Symmetric key - CSP	Key Derivation with TLS 1.2 KDF		
IKE Derived Key	Symmetric key derived from a shared secret using IKEv2 KDF	112-4096 bits - 112-256 bits	Symmetric key - CSP	Key Derivation with IKEv2 KDF		
Entropy Input	Entropy input used to seed the DRBG	440-880 bits -	Entropy input - CSP	Random Number Generation with		Random Number Generation

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		440-880 bits		Hash_DRBG		with Hash_DRBG
DRBG Seed	DRBG seed derived from entropy input	440 bits - 256 bits	Seed - CSP	Random Number Generation with Hash_DRBG		Random Number Generation with Hash_DRBG
Internal State (V, C)	Internal state of the Hash_DRBG	880 bits - 256 bits	Internal state - CSP	Random Number Generation with Hash_DRBG		Random Number Generation with Hash_DRBG
Module-generated FFC Private Key	FFC private key generated by the module	2048-8192 bits - 112-200 bits	Private key - CSP	Key Pair Generation with Safe Primes		Key Pair Generation with Safe Primes
Module-generated FFC Public Key	FFC public key generated by the module	2048-8192 bits - 112-200 bits	Public key - PSP	Key Pair Generation with Safe Primes		Key Pair Generation with Safe Primes
FFC Private Key	Private key used for FFC Diffie-Hellman	2048-8192 bits - 112-200 bits	Private key - CSP			Shared Secret Computation with KAS-FFC-SSC
FFC Public Key	Public key used for FFC Diffie-Hellman	2048-8192 bits - 112-200 bits	Public key - PSP			Shared Secret Computation with KAS-FFC-SSC
Module-generated EC Private Key	EC private key generated by the module	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP	Key Pair Generation with ECDSA		Key Pair Generation with ECDSA

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Module-generated EC Public Key	EC public key generated by the module	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP	Key Pair Generation with ECDSA		Key Pair Generation with ECDSA
EC Private Key	Private key used for EC Diffie-Hellman and ECDSA	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP			Shared Secret Computation with KAS-ECC-SSC Signature Generation with ECDSA
EC Public Key	Public key used for EC Diffie-Hellman and ECDSA	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP			Shared Secret Computation with KAS-ECC-SSC Signature Verification with ECDSA
Module-generated RSA Private Key	RSA private key generated by the module	2048, 3072, 4096 bits - 112, 128, 149 bits	Private key - CSP	Key Pair Generation with RSA		Key Pair Generation with RSA
Module-generated RSA Public Key	RSA public key generated by the module	2048, 3072, 4096 bits - 112, 128, 149 bits bits	Public key - PSP	Key Pair Generation with RSA		Key Pair Generation with RSA
RSA Private Key	RSA private key written to the module	2048, 3072, 4096 bits - 112, 128, 149 bits	Private key - CSP			Signature Generation with RSA Key Decapsulation with RSA (KTS)

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
RSA Public Key	RSA public key written to the module	2048, 3072, 4096 bits - 112, 128, 149 bits	Public key - PSP			Signature Verification with RSA Key Encapsulation with RSA (KTS)
Module-generated ML-DSA Private Key	ML-DSA private key generated by the module	2560, 4032, 4896 bytes - 128, 192, 256 bits	Private key - CSP	Key Pair Generation with ML-DSA		Key Pair Generation with ML-DSA
Module-generated ML-DSA Public Key	ML-DSA public key generated by the module	1312, 1952, 2592 bytes - 128, 192, 256 bits	Public key - PSP	Key Pair Generation with ML-DSA		Key Pair Generation with ML-DSA
ML-DSA Private Key	Private key used for ML-DSA signature generation	2560, 4032, 4896 bytes - 128, 192, 256 bits	Private key - CSP			Signature Generation with ML-DSA
ML-DSA Public Key	Public key used for ML-DSA signature verification	1312, 1952, 2592 bytes - 128, 192, 256 bits	Public key - PSP			Signature Verification with ML-DSA
ML-DSA Seed	ML-DSA Seed used for key pair generation with ML-DSA.	256 bits - 128, 192, and 256 bits	Seed - CSP	Key Pair Generation with ML-DSA		Key Pair Generation with ML-DSA

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Module-generated ML-KEM Encapsulation Key	ML-KEM encapsulation key generated by the module	1184, 1568 bytes - 192, 256 bits	Encapsulation key - PSP	Key Pair Generation with ML-KEM		Key Pair Generation with ML-KEM
Module-generated ML-KEM Decapsulation Key	ML-KEM decapsulation key generated by the module	2400, 3168 bytes - 192, 256 bits	Decapsulation key - CSP	Key Pair Generation with ML-KEM		Key Pair Generation with ML-KEM
ML-KEM Encapsulation Key	ML-KEM encapsulation key written to the module	1184, 1568 bytes - 192, 256 bits	Encapsulation key - PSP			Key Encapsulation with ML-KEM
ML-KEM Decapsulation Key	ML-KEM decapsulation key written to the module	2400, 3168 bytes - 192, 256 bits	Decapsulation key - CSP			Key Decapsulation with ML-KEM
ML-KEM Shared Secret	Shared secret outputted by ML-KEM encapsulation / decapsulation	256 bits - 192, 256 bits	Shared secret - CSP		Key Encapsulation with ML-KEM Key Decapsulation with ML-KEM	Key Encapsulation with ML-KEM Key Decapsulation with ML-KEM
ML-KEM Seed	ML-KEM Seed used for key pair generation with ML-KEM.	512 bits - 192, 256 bits	Seed - CSP	Key Pair Generation with ML-KEM		Key Pair Generation with ML-KEM
Intermediate key generation value	Temporary value generated during key	ML-KEM, ML-DSA: 32-2048	Intermediate value - CSP	Key Pair Generation with RSA Key Pair		Key Pair Generation with RSA Key Pair

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	generation services	bytes; Others: 256-8192 bits - ML-KEM: 192, 256 bits; ML-DSA: 128, 192, 256 bits; Others: 112-256 bits		Generation with ECDSA Key Pair Generation with Safe Primes Key Pair Generation with ML-KEM Key Pair Generation with ML-DSA		Generation with ECDSA Key Pair Generation with Safe Primes Key Pair Generation with ML-KEM Key Pair Generation with ML-DSA

Table 18: SSP Table 1

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
KBKDF Derived Key	API output parameters (encrypted) - AES API output parameters (encrypted) - RSA	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Key-Derivation Key:Derived From
PBKDF2 Derived Key	API output parameters (encrypted) - AES API output parameters (encrypted) - RSA	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Password:Derived From
HKDF Derived Key	API output parameters (encrypted) - AES API output parameters (encrypted) - RSA	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Shared Secret:Derived From
TLS Derived Key	API output parameters (encrypted) - AES API output parameters (encrypted) - RSA	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Shared Secret:Derived From
IKE Derived Key	API output parameters (encrypted) - AES API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Shared Secret:Derived From

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
	AES API input parameters (encrypted) - RSA		zeroized by operator	power from the module	
ML-DSA Public Key	API input parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	ML-DSA Private Key:Paired With
ML-DSA Seed		RAM:Plaintext	From service invocation until service completion	Automatic Remove power from the module	Module-generated ML-DSA Private Key:Generation Of Module-generated ML-DSA Public Key:Generation Of ML-DSA Private Key:Used with
Module-generated ML-KEM Encapsulation Key	API output parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Module-generated ML-KEM Decapsulation Key:Paired With ML-KEM Seed:Generated From
Module-generated ML-KEM Decapsulation Key	API output parameters (encrypted) - AES API output parameters (encrypted) - RSA	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	Module-generated ML-KEM Encapsulation Key:Paired With ML-KEM Seed:Generated From
ML-KEM Encapsulation Key	API input parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	ML-KEM Decapsulation Key:Paired With ML-KEM Shared Secret:Establishes

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
ML-KEM Decapsulation Key	API input parameters (encrypted) - AES API input parameters (encrypted) - RSA	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	ML-KEM Encapsulation Key:Paired With ML-KEM Shared Secret:Establishes
ML-KEM Shared Secret	API input parameters (encrypted) - AES API input parameters (encrypted) - RSA API output parameters (encrypted) - AES API output parameters (encrypted) - RSA	RAM:Plaintext	Until explicitly zeroized by operator	Destroy Object Remove power from the module	ML-KEM Encapsulation Key:Established by ML-KEM Decapsulation Key:Established by
ML-KEM Seed		RAM:Plaintext	From service invocation until service completion	Automatic Remove power from the module	Module-generated ML-KEM Encapsulation Key:Generation Of Module-generated ML-KEM Decapsulation Key:Generation Of ML-KEM Encapsulation Key:Used with ML-KEM Shared Secret:Derives
Intermediate key generation value		RAM:Plaintext	For the duration of the service	Automatic	Module-generated FFC Private Key:Generation Of Module-generated

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
					FFC Public Key:Generation Of Module-generated EC Private Key:Generation Of Module-generated EC Public Key:Generation Of Module-generated RSA Private Key:Generation Of Module-generated RSA Public Key:Generation Of Module-generated ML-KEM Encapsulation Key:Generation Of Module-generated ML-KEM Decapsulation Key:Generation Of Module-generated ML-DSA Private Key:Generation Of Module-generated ML-DSA Public Key:Generation Of

Table 19: SSP Table 2

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

10 Self-Tests

Upon initialization, the module immediately performs all libfreeblpriv3.so cryptographic algorithm self-tests (CASTs) as specified in the Conditional Self-Tests table. 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. Consequently, the HMAC-SHA2-256 algorithm goes through a CAST before the software integrity tests are performed.

Then, the module performs the RSA CAST in the libsoftokn3.so library, followed by the pre-operational integrity test on the libsoftokn3.so file using its associated check value. The CAST for the algorithm used in the pre-operational self-test (i.e., HMAC-SHA2-256) was already performed by the libfreeblpriv3.so library, before the libsoftokn3.so library integrity test. Finally, all remaining CASTs for the algorithms implemented in libsoftokn3.so are executed (see the Conditional Self-Tests table).

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

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

10.1 Pre-Operational Self-Tests

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

Table 20: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				operational and services are available for use		
KDF SP800-108	HMAC-SHA2-256 in counter mode	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Module initialization
KDA HKDF SP800-56Cr2	SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Module initialization
TLS v1.2 KDF RFC7627	SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Module initialization
KDF IKEv2	SHA-1, SHA-256, SHA-384, SHA-512	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Module initialization
PBKDF	SHA2-256 with 5 iterations, 128-bit salt and 14 characters password	KAT	CAST	Module becomes operational and services are available for use	Key Derivation	Module initialization
Hash DRBG	SHA-256 without	KAT	CAST	Module becomes operational	Instantiate Generate; Reseed Generate	Module initialization

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
	prediction resistance			and services are available for use	(compliant to SP 800-90A Rev. 1, Section 11.3)	
KAS-FFC-SSC Sp800-56Ar3	ffdhe2048	KAT	CAST	Module becomes operational and services are available for use	Shared Secret Computation	Module initialization
KAS-ECC-SSC Sp800-56Ar3	P-256	KAT	CAST	Module becomes operational and services are available for use	Shared Secret Computation	Module initialization
RSA SigGen (FIPS186-5)	PKCS#1 v1.5 with SHA2-256, SHA2-384, SHA2-512, and 2048-bit key	KAT	CAST	Module becomes operational and services are available for use	Signature Generation	Module initialization
RSA SigVer (FIPS186-5)	PKCS#1 v1.5 with SHA2-256, SHA2-384, SHA2-512, and 2048-bit key	KAT	CAST	Module becomes operational and services are available for use	Signature Verification	Module initialization
ECDSA SigGen (FIPS186-5)	SHA2-256 and P-256	KAT	CAST	Module becomes operational and services are available for use	Signature Generation	Module initialization
ECDSA SigVer (FIPS186-5)	SHA2-256 and P-256	KAT	CAST	Module becomes operational and services	Signature Verification	Module initialization

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				are available for use		
ML-DSA SigGen	ML-DSA-44, ML-DSA-65, ML-DSA-87	KAT	CAST	Module becomes operational and services are available for use	Signature Generation	Module initialization
ML-DSA SigVer	ML-DSA-44, ML-DSA-65, ML-DSA-87	KAT	CAST	Module becomes operational and services are available for use	Signature Verification	Module initialization
ML-DSA KeyGen	ML-DSA-44, ML-DSA-65, ML-DSA-87	KAT	CAST	Module becomes operational and services are available for use	Key pair generation from known seed	Module initialization
ML-KEM EncapDecap - Encapsulation	ML-KEM-768, ML-KEM-1024	KAT	CAST	Module becomes operational and services are available for use	Encapsulation	Module initialization
ML-KEM EncapDecap - Decapsulation	ML-KEM-768, ML-KEM-1024	KAT	CAST	Module becomes operational and services are available for use	Decapsulation	Module initialization
ML-KEM EncapDecap - Implicit rejection path	ML-KEM-768, ML-KEM-1024	KAT	CAST	Module becomes operational and services are available for use	Decapsulation covering implicit rejection path	Module initialization

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
ML-KEM KeyGen	ML-KEM-768, ML-KEM-1024	KAT	CAST	Module becomes operational and services are available for use	Key pair generation from known seed	Module initialization
RSA KeyGen (FIPS186-5) - Sign/Verify	PKCS#1 v1.5 with SHA-256	PCT	PCT	Successful key pair generation	Signature Generation and Signature Verification	Key Pair Generation
RSA KeyGen (FIPS186-5) - Enc/Dec	OAEP with SHA-256	PCT	PCT	Successful key pair generation	Encryption and decryption	Key Pair Generation
ECDSA KeyGen (FIPS186-5) - SSC	N/A	PCT	PCT	Successful key pair generation	PCT according to section 5.6.2.1.4 of SP 800-56A Rev. 3	Key Pair Generation
ECDSA KeyGen (FIPS186-5) - SigGen/SigVer	SHA-256	PCT	PCT	Successful key pair generation	Signature Generation and Signature Verification	Key Pair Generation
Safe Primes Key Generation	N/A	PCT	PCT	Successful key pair generation	PCT according to section 5.6.2.1.4 of SP 800-56A Rev. 3	Key Pair Generation
ML-DSA KeyGen - PCT	N/A	PCT	PCT	Successful key pair generation	Signature Generation and Signature Verification	Key Pair Generation
ML-KEM KeyGen - PCT	N/A	PCT	PCT	Successful key pair generation	Encapsulation and Decapsulation	Key Pair Generation

Table 21: Conditional Self-Tests

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

10.3 Periodic Self-Test Information

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

Table 22: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
SHA2-224	KAT	CAST	On demand	Manually
SHA2-256	KAT	CAST	On demand	Manually
SHA2-384	KAT	CAST	On demand	Manually
SHA2-512	KAT	CAST	On demand	Manually
HMAC-SHA2-224	KAT	CAST	On demand	Manually
HMAC-SHA2-256	KAT	CAST	On demand	Manually
HMAC-SHA2-384	KAT	CAST	On demand	Manually
HMAC-SHA2-512	KAT	CAST	On demand	Manually
AES-ECB - Encrypt	KAT	CAST	On demand	Manually
AES-ECB - Decrypt	KAT	CAST	On demand	Manually
AES-CBC - Encrypt	KAT	CAST	On demand	Manually
AES-CBC - Decrypt	KAT	CAST	On demand	Manually
AES-CMAC	KAT	CAST	On demand	Manually
AES-GCM - Encrypt	KAT	CAST	On demand	Manually
AES-GCM - Decrypt	KAT	CAST	On demand	Manually
KDF SP800-108	KAT	CAST	On demand	Manually
KDA HKDF SP800-56Cr2	KAT	CAST	On demand	Manually
TLS v1.2 KDF RFC7627	KAT	CAST	On demand	Manually
KDF IKEv2	KAT	CAST	On demand	Manually

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
PBKDF	KAT	CAST	On demand	Manually
Hash DRBG	KAT	CAST	On demand	Manually
KAS-FFC-SSC Sp800-56Ar3	KAT	CAST	On demand	Manually
KAS-ECC-SSC Sp800-56Ar3	KAT	CAST	On demand	Manually
RSA SigGen (FIPS186-5)	KAT	CAST	On demand	Manually
RSA SigVer (FIPS186-5)	KAT	CAST	On demand	Manually
ECDSA SigGen (FIPS186-5)	KAT	CAST	On demand	Manually
ECDSA SigVer (FIPS186-5)	KAT	CAST	On demand	Manually
ML-DSA SigGen	KAT	CAST	On demand	Manually
ML-DSA SigVer	KAT	CAST	On demand	Manually
ML-DSA KeyGen	KAT	CAST	On demand	Manually
ML-KEM EncapDecap - Encapsulation	KAT	CAST	On demand	Manually
ML-KEM EncapDecap - Decapsulation	KAT	CAST	On demand	Manually
ML-KEM EncapDecap - Implicit rejection path	KAT	CAST	On demand	Manually
ML-KEM KeyGen	KAT	CAST	On demand	Manually
RSA KeyGen (FIPS186-5) - Sign/Verify	PCT	PCT	On demand	Manually
RSA KeyGen (FIPS186-5) - Enc/Dec	PCT	PCT	On demand	Manually
ECDSA KeyGen (FIPS186-5) - SSC	PCT	PCT	On demand	Manually
ECDSA KeyGen (FIPS186-5) - SigGen/SigVer	PCT	PCT	On demand	Manually
Safe Primes Key Generation	PCT	PCT	On demand	Manually
ML-DSA KeyGen - PCT	PCT	PCT	On demand	Manually
ML-KEM KeyGen - PCT	PCT	PCT	On demand	Manually

Table 23: Conditional Periodic Information

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

10.4 Error States

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

Table 24: Error States

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

10.5 Operator Initiation of Self-Tests

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

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

- Rocky Linux 9 system:
  - nss-softokn-3.112.0-4.el9_6ciqfips.2.1.x86_64
  - nss-softokn-freebl-3.112.0-4.el9_6ciqfips.2.1.x86_64

Before these packages are installed, the Rocky Linux 9 system must operate in the FIPS validated configuration. This can be achieved by:

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

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

After the completion of the above-mentioned installation steps, the RPM packages can be installed on the Rocky Linux 9 system using “dnf install” command.

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

Rocky Linux 9 NSS Cryptographic Module rocky9.20251215 3.112.0

Alternatively, the /usr/lib64/nss/unsupported-tools/validation tool is provided as a convenience by the nss-tools-3.112.0-4.el9_6ciqfips.2.1.x86_64 package. This tool performs the same steps, and also outputs the FIPS module identifier as above.

11.2 Administrator Guidance

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

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-3.112.0-

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

4.el9_6ciqfips.2.1.x86_64 and nss-softokn-freebl-3.112.0-4.el9_6ciqfips.2.1.x86_64 RPM packages can be uninstalled from the Rocky Linux 9 system.

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

12 Mitigation of Other Attacks

12.1 Attack List

Timing attacks on RSA

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

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

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

Arithmetic errors in RSA signatures

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

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Appendix A Glossary and Abbreviations

AES	Advanced Encryption Standard
API	Application Programming Interface
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CKG	Cryptographic Key Generation
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter
CTS	Ciphertext Stealing
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
HKDF	HMAC-based Key Derivation Function
HMAC	Keyed-Hash Message Authentication Code
IKE	Internet Key Exchange
KAS	Key Agreement Scheme
KAT	Known Answer Test
KBKDF	Key-based Key Derivation Function

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

KTS	Key Transport Scheme
KW	Key Wrap
KWP	Key Wrap with Padding
MAC	Message Authentication Code
ML-DSA	Module-Lattice-Based Digital Signature Algorithm
ML-KEM	Module-Lattice-Based Key-Encapsulation Mechanism
NIST	National Institute of Science and Technology
OAEP	Optimal Asymmetric Encryption Padding
PAA	Processor Algorithm Acceleration
PAI	Processor Algorithm Implementation
PCT	Pair-wise Consistency Test
PBKDF2	Password-based Key Derivation Function v2
PKCS	Public-Key Cryptography Standards
PSS	Probabilistic Signature Scheme
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SSC	Shared Secret Computation
SSP	Sensitive Security Parameter
TLS	Transport Layer Security
XOF	Extendable Output Function

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Appendix B References

FIPS 140-3 **FIPS PUB 140-3 - Security Requirements For Cryptographic Modules**
March 2019
https://doi.org/10.6028/NIST.FIPS.140-3

FIPS 140-3 IG **Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program**
2 September 2025
https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements

FIPS 180-4 **Secure Hash Standard (SHS)**
August 2015
https://doi.org/10.6028/NIST.FIPS.180-4

FIPS 186-5 **Digital Signature Standard (DSS)**
February 2023
https://doi.org/10.6028/NIST.FIPS.186-5

FIPS 197 **Advanced Encryption Standard**
May 2023
https://doi.org/10.6028/NIST.FIPS.197-upd1

FIPS 198-1 **The Keyed Hash Message Authentication Code (HMAC)**
July 2008
https://doi.org/10.6028/NIST.FIPS.198-1

FIPS 203 **Module-Lattice-Based Key-Encapsulation Mechanism Standard**
August 2024
https://doi.org/10.6028/NIST.FIPS.203

FIPS 204 **Module-Lattice-Based Digital Signature Standard**
August 2024
https://doi.org/10.6028/NIST.FIPS.204

PKCS#1 **Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.2**
November 2016
https://www.rfc-editor.org/rfc/rfc8017.txt

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

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 7627	**Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension** September 2015 https://www.rfc-editor.org/rfc/rfc7627.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://doi.org/10.6028/NIST.SP.800-38A
SP 800-38A Addendum	**Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode** October 2010 https://doi.org/10.6028/NIST.SP.800-38A-Add
SP 800-38B	**Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 https://doi.org/10.6028/NIST.SP.800-38B
SP 800-38D	**Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 https://doi.org/10.6028/NIST.SP.800-38D

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

SP 800-38F	**Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 https://doi.org/10.6028/NIST.SP.800-38F
SP 800-52 Rev. 2	**Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations** August 2019 https://doi.org/10.6028/NIST.SP.800-52r2
SP 800-56A Rev. 3	**Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography** April 2018 https://doi.org/10.6028/NIST.SP.800-56Ar3
SP 800-56B Rev. 2	**Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography** March 2019 https://doi.org/10.6028/NIST.SP.800-56Br2
SP 800-56C Rev. 2	**Recommendation for Key-Derivation Methods in Key-Establishment Schemes** August 2020 https://doi.org/10.6028/NIST.SP.800-56Cr2
SP 800-90A Rev. 1	**Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 https://doi.org/10.6028/NIST.SP.800-90Ar1
SP 800-90B	**Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://doi.org/10.6028/NIST.SP.800-90B
SP 800-108 Rev. 1	**NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions** August 2022 https://doi.org/10.6028/NIST.SP.800-108r1-upd1
SP 800-132	**Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** December 2010 https://doi.org/10.6028/NIST.SP.800-132

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

SP 800-133 Rev. 2	**Recommendation for Cryptographic Key Generation** June 2020 https://doi.org/10.6028/NIST.SP.800-133r2
SP 800-135 Rev. 1	**Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 https://doi.org/10.6028/NIST.SP.800-135r1

© 2026 Ctrl IQ, Inc. / atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.