Ctrl IQ, Inc.

Rocky Linux 9 OpenSSL FIPS Provider

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: 1.1
Last update: 2026-01-05

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for version Rocky9.20250210 of the Rocky Linux 9 OpenSSL FIPS Provider. 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.

1.2 Security Levels

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

Table 1: Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The Rocky Linux 9 OpenSSL FIPS Provider (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) for use by other applications that require cryptographic functionality. The module consists of one software component, the “FIPS provider”, which implements the FIPS requirements and the cryptographic functionality provided to the operator.

Module Type: Software

Module Embodiment: MultiChipStand

Cryptographic Boundary:

The cryptographic boundary of the module is defined as the shared library implementing the FIPS provider (fips.so).

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 cryptographic boundary (orange) and TOEPP (purple) are schematically represented in Figure 1. Green lines indicate the flow of data between the cryptographic module and its operator application. Components in white are only included in the diagram for informational purposes and are not part of the module’s validation.

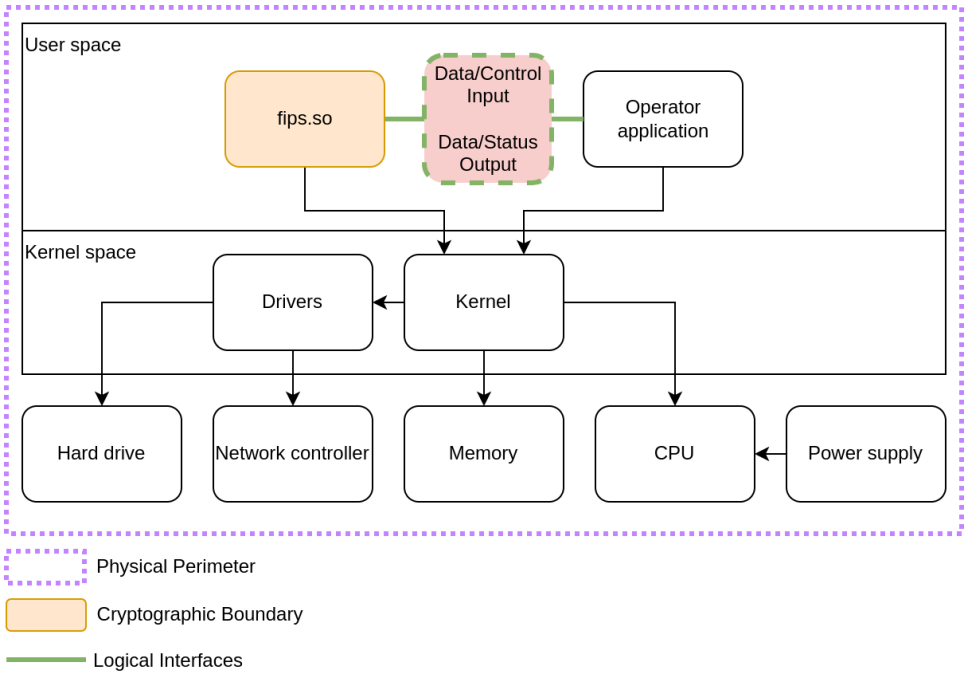

Figure 1: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
fips.so	Rocky9.20250210	N/A	HMAC SHA-256

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

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Rocky Linux 9	SuperMicro SuperServer 5039MS	Intel Kaby Lake Xeon E3-1270 v6	Yes	N/A	Rocky9.20250210
Rocky Linux 9	SuperMicro SuperServer 5039MS	Intel Kaby Lake Xeon E3-1270 v6	No	N/A	Rocky9.20250210

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

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

N/A for this module.

2.3 Excluded Components

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

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved	Automatically entered whenever an approved service is requested.	Approved	Equivalent to the indicator of the requested service: Message digest 'EVP_DigestFinal_ex returns 1'; XOF 'EVP_DigestFinalXOF returns 1'; Encryption 'EVP_EncryptFinal_ex returns 1'; Decryption 'EVP_DecryptFinal_ex returns 1'; Authenticated encryption 'AES-GCM: EVP_CIPHER_ROCKY_FIPS_INDICATOR_APPROVED; Others: EVP_EncryptFinal_ex returns 1'; Authenticated decryption 'EVP_DecryptFinal_ex returns 1'; MAC 'HMAC: OSSL_MAC_PARAM_ROCKY_FIPS_INDICATOR_APPROVED; Others: EVP_MAC_final returns 1'; KDF 'EVP_KDF_ROCKY_FIPS_INDICATOR_APPROVED'; Random number generation 'EVP_RAND_generate returns 1'; Shared secret computation 'EVP_PKEY_ROCKY_FIPS_INDICATOR_APPROVED'; Signature generation/verification

Mode Name	Description	Type	Status Indicator
			'OSSL_RL_FIPSINDICATOR_APPROVED and EVP_PKEY_ROCKY_FIPS_INDICATOR_APPROVED'; Asymmetric Encryption/Decryption 'EVP_PKEY_ROCKY_FIPS_INDICATOR_APPROVED'; Key pair generation 'EVP_PKEY_generate returns 1'; Key pair verification 'EVP_PKEY_public_check or EVP_PKEY_private_check or EVP_PKEY_check returns 1'
Non-Approved	Automatically entered whenever a non-approved service is requested.	Non-Approved	The status indicator depends on the invoked service and is equivalent to its indicator. As there is no non-approved service indicator, if the return code or flag differs from the approved ones it automatically implies the non-approved mode of operation

Table 4: Modes List and Description

Once the module is installed and initilized as per Section 11.1, after passing all pre-operational self-tests and conditional self-tests executed on startup, the module automatically transitions to the approved mode. On startup, 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	A6603, A6607, A6608	-	SP 800-38A
AES-CBC-CS1	A6603, A6607, A6608	-	SP 800-38A
AES-CBC-CS2	A6603, A6607, A6608	-	SP 800-38A
AES-CBC-CS3	A6603, A6607, A6608	-	SP 800-38A
AES-CCM	A6603, A6607, A6608	-	SP 800-38C
AES-CFB1	A6603, A6607, A6608	-	SP 800-38A
AES-CFB128	A6603, A6607, A6608	-	SP 800-38A
AES-CFB8	A6603, A6607, A6608	-	SP 800-38A
AES-CMAC	A6603, A6607, A6608	-	SP 800-38B
AES-CTR	A6603, A6607, A6608	-	SP 800-38A
AES-ECB	A6603, A6607, A6608	-	SP 800-38A
AES-GCM	A6604, A6609, A6610, A6611, A6612, A6613, A6614, A6615, A6616	-	SP 800-38D
AES-GMAC	A6604, A6609, A6610, A6611, A6612, A6613, A6614, A6615, A6616	-	SP 800-38D
AES-KW	A6603, A6607, A6608	-	SP 800-38F
AES-KWP	A6603, A6607, A6608	-	SP 800-38F
AES-OFB	A6603, A6607, A6608	-	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
AES-XTS Testing Revision 2.0	A6603, A6607, A6608	-	SP 800-38E
Counter DRBG	A5957	-	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-5)	A6605, A6617, A6618, A6619	-	FIPS 186-5
ECDSA KeyVer (FIPS186-5)	A6605, A6617, A6618, A6619	-	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A6605, A6606, A6617, A6618, A6619	-	FIPS 186-5
ECDSA SigVer (FIPS186-5)	A6605, A6606, A6617, A6618, A6619	-	FIPS 186-5
EDDSA KeyGen	A6328	-	FIPS 186-5
EDDSA SigGen	A6328	-	FIPS 186-5
EDDSA SigVer	A6328	-	FIPS 186-5
Hash DRBG	A5957	-	SP 800-90A Rev. 1
HMAC DRBG	A5957	-	SP 800-90A Rev. 1
HMAC-SHA-1	A6605, A6617, A6618, A6619	-	FIPS 198-1
HMAC-SHA2-224	A6605, A6617, A6618, A6619	-	FIPS 198-1
HMAC-SHA2-256	A6605, A6617, A6618, A6619	-	FIPS 198-1
HMAC-SHA2-384	A6605, A6617, A6618, A6619	-	FIPS 198-1
HMAC-SHA2-512	A6605, A6617, A6618, A6619	-	FIPS 198-1
HMAC-SHA2-512/224	A6605, A6617, A6618, A6619	-	FIPS 198-1
HMAC-SHA2-512/256	A6605, A6617, A6618, A6619	-	FIPS 198-1
HMAC-SHA3-224	A6606	-	FIPS 198-1
HMAC-SHA3-256	A6606	-	FIPS 198-1
HMAC-SHA3-384	A6606	-	FIPS 198-1
HMAC-SHA3-512	A6606	-	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A6605, A6617, A6618, A6619	-	SP 800-56A Rev. 3
KAS-FFC-SSC Sp800-56Ar3	A6602	-	SP 800-56A Rev. 3
KAS-IFC-SSC	A6605, A6617, A6618, A6619	-	SP 800-56A Rev. 3
KDA HKDF SP800-56Cr2	A6601	-	SP 800-56C Rev. 2
KDA OneStep SP800-56Cr2	A6600	-	SP 800-56C Rev. 2
KDA TwoStep SP800-56Cr2	A6600	-	SP 800-56C Rev. 2
KDF ANS 9.42 (CVL)	A6605, A6606, A6617, A6618, A6619	-	SP 800-135 Rev. 1
KDF ANS 9.63 (CVL)	A6605, A6606, A6617, A6618, A6619	-	SP 800-135 Rev. 1
KDF SP800-108	A6599	-	SP 800-108 Rev. 1
KDF SSH (CVL)	A6605, A6617, A6618, A6619	-	SP 800-135 Rev. 1
KTS-IFC	A6605, A6617, A6618, A6619	-	SP 800-56B Rev. 2

Algorithm	CAVP Cert	Properties	Reference
PBKDF	A6605, A6606, A6617, A6618, A6619	-	SP 800-132
RSA KeyGen (FIPS186-5)	A6605, A6617, A6618, A6619	-	FIPS 186-5
RSA SigGen (FIPS186-5)	A6605, A6606, A6617, A6618, A6619	-	FIPS 186-5
RSA SigVer (FIPS186-2)	A6605, A6617, A6618, A6619	-	FIPS 186-4
RSA SigVer (FIPS186-4)	A6605, A6617, A6618, A6619	-	FIPS 186-4
RSA SigVer (FIPS186-5)	A6605, A6606, A6617, A6618, A6619	-	FIPS 186-5
Safe Primes Key Generation	A6602	-	SP 800-56A Rev. 3
Safe Primes Key Verification	A6602	-	SP 800-56A Rev. 3
SHA-1	A6605, A6617, A6618, A6619	-	FIPS 180-4
SHA2-224	A6605, A6617, A6618, A6619	-	FIPS 180-4
SHA2-256	A6605, A6617, A6618, A6619	-	FIPS 180-4
SHA2-384	A6605, A6617, A6618, A6619	-	FIPS 180-4
SHA2-512	A6605, A6617, A6618, A6619	-	FIPS 180-4
SHA2-512/224	A6605, A6617, A6618, A6619	-	FIPS 180-4
SHA2-512/256	A6605, A6617, A6618, A6619	-	FIPS 180-4
SHA3-224	A6606	-	FIPS 202
SHA3-256	A6606	-	FIPS 202
SHA3-384	A6606	-	FIPS 202
SHA3-512	A6606	-	FIPS 202
SHAKE-128	A6606	-	FIPS 202
SHAKE-256	A6606	-	FIPS 202
TLS v1.2 KDF RFC7627 (CVL)	A6605, A6617, A6618, A6619	-	SP 800-135 Rev. 1
TLS v1.3 KDF (CVL)	A6601	-	SP 800-135 Rev. 1

Table 5: Approved Algorithms

Vendor-Affirmed Algorithms:

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

Table 6: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
AES-GCM with external IV	Authenticated encryption
HMAC with < 112-bit keys	Message authentication
KBKDF with < 112-bit keys	Key derivation
KDA OneStep, HKDF with < 112-bit keys	Key derivation
KDA OneStep with SHAKE128, SHAKE256	Key derivation
ANS X9.42 KDF, ANS X9.63 KDF with < 112-bit keys	Key derivation
ANS X9.42 KDF with SHAKE128, SHAKE256	Key derivation
ANS X9.63 KDF with SHA-1, SHAKE128, SHAKE256	Key derivation
SSH KDF with SHA-512/224, SHA-512/256, SHA-3, SHAKE128, SHAKE256	Key derivation
TLS 1.2 KDF with SHA-1, SHA-224, SHA-512/224, SHA-512/256, SHA-3	Key derivation
TLS 1.3 KDF with SHA-1, SHA-224, SHA-512, SHA-512/224, SHA-512/256, SHA-3	Key derivation
PBKDF2 with short password, short salt, insufficient iterations, < 112-bit output keys	Password-based key derivation
RSA-PSS with invalid salt length	Signature generation / verification
RSA with no padding	Signature generation / verification
RSA and ECDSA with no hashing	Signature generation / verification

Table 7: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Message digest	SHA	Compute a message digest		SHA-1: (A6605, A6617, A6618, A6619) SHA2-224: (A6605, A6617, A6618, A6619) SHA2-256: (A6605, A6617, A6618, A6619) SHA2-384: (A6605, A6617, A6618, A6619) SHA2-512: (A6605, A6617, A6618, A6619) SHA2-512/224: (A6605, A6617, A6618, A6619) SHA2-512/256: (A6605, A6617, A6618, A6619) SHA3-224: (A6606) SHA3-256: (A6606)

Name	Type	Description	Properties	Algorithms
				SHA3-384: (A6606) SHA3-512: (A6606)
XOF	XOF	Compute an extendable output message digest		SHAKE-128: (A6606) SHAKE-256: (A6606)
Encryption	BC-UnAuth	Encrypt a plaintext		AES-CBC: (A6603, A6607, A6608) AES-CBC-CS1: (A6603, A6607, A6608) AES-CBC-CS2: (A6603, A6607, A6608) AES-CBC-CS3: (A6603, A6607, A6608) AES-CFB1: (A6603, A6607, A6608) AES-CFB128: (A6603, A6607, A6608) AES-CFB8: (A6603, A6607, A6608) AES-CTR: (A6603, A6607, A6608) AES-ECB: (A6603, A6607, A6608) AES-OFB: (A6603, A6607, A6608) AES-XTS Testing Revision 2.0: (A6603, A6607, A6608)
Decryption	BC-UnAuth	Decrypt a ciphertext		AES-CBC: (A6603, A6607, A6608) AES-CBC-CS1: (A6603, A6607, A6608) AES-CBC-CS2: (A6603, A6607, A6608) AES-CBC-CS3: (A6603, A6607, A6608)

Name	Type	Description	Properties	Algorithms
				AES-CFB1: (A6603, A6607, A6608) AES-CFB128: (A6603, A6607, A6608) AES-CFB8: (A6603, A6607, A6608) AES-CTR: (A6603, A6607, A6608) AES-ECB: (A6603, A6607, A6608) AES-OFB: (A6603, A6607, A6608) AES-XTS Testing Revision 2.0: (A6603, A6607, A6608)
Authenticated encryption	BC-Auth	Encrypt and authenticate a plaintext		AES-CCM: (A6603, A6607, A6608) AES-GCM: (A6604, A6609, A6610, A6611, A6612, A6613, A6614, A6615, A6616) AES-KW: (A6603, A6607, A6608) AES-KWP: (A6603, A6607, A6608)
Authenticated decryption	BC-Auth	Decrypt and authenticate a ciphertext		AES-CCM: (A6603, A6607, A6608) AES-GCM: (A6604, A6609, A6610, A6611, A6612, A6613, A6614, A6615, A6616) AES-KW: (A6603, A6607, A6608) AES-KWP: (A6603, A6607, A6608)
Message authentication	MAC	Compute a MAC tag		AES-CMAC: (A6603, A6607, A6608) AES-GMAC:

Name	Type	Description	Properties	Algorithms
				(A6604, A6609, A6610, A6611, A6612, A6613, A6614, A6615, A6616) HMAC-SHA-1: (A6605, A6617, A6618, A6619) HMAC-SHA2-224: (A6605, A6617, A6618, A6619) HMAC-SHA2-256: (A6605, A6617, A6618, A6619) HMAC-SHA2-384: (A6605, A6617, A6618, A6619) HMAC-SHA2-512: (A6605, A6617, A6618, A6619) HMAC-SHA2-512/224: (A6605, A6617, A6618, A6619) HMAC-SHA2-512/256: (A6605, A6617, A6618, A6619) HMAC-SHA3-224: (A6606) HMAC-SHA3-256: (A6606) HMAC-SHA3-384: (A6606) HMAC-SHA3-512: (A6606)
Key-based key derivation	KBKDF	Derive keying material from a key-derivation key		KDF SP800-108: (A6599)
Key-establishment key derivation	KAS-56CKDF	Derive keying material from a shared secret		KDA OneStep SP800-56Cr2: (A6600) KDA TwoStep SP800-56Cr2: (A6600) KDA HKDF SP800-56Cr2: (A6601)
Protocol key derivation	KAS-135KDF	Derive keying material from a shared secret		TLS v1.3 KDF: (A6601) KDF ANS 9.42: (A6605, A6606, A6617, A6618, A6619)

Name	Type	Description	Properties	Algorithms
				KDF ANS 9.63: (A6605, A6606, A6617, A6618, A6619) KDF SSH: (A6605, A6617, A6618, A6619) TLS v1.2 KDF RFC7627: (A6605, A6617, A6618, A6619)
Password-based key derivation	PBKDF	Derive keying material from a password		PBKDF: (A6605, A6606, A6617, A6618, A6619)
Random number generation	DRBG	Generate random bytes		Counter DRBG: (A5957) Hash DRBG: (A5957) HMAC DRBG: (A5957)
Shared secret computation	KAS-SSC	Compute a shared secret		KAS-FFC-SSC Sp800-56Ar3: (A6602) KAS-ECC-SSC Sp800-56Ar3: (A6605, A6617, A6618, A6619) KAS-IFC-SSC: (A6605, A6617, A6618, A6619)
Signature generation	DigSig-SigGen	Generate a digital signature		ECDSA SigGen (FIPS186-5): (A6605, A6606, A6617, A6618, A6619) RSA SigGen (FIPS186-5): (A6605, A6606, A6617, A6618, A6619) EDDSA SigGen: (A6328)
Signature verification	DigSig-SigVer	Verify a digital signature		ECDSA SigVer (FIPS186-5): (A6605, A6606, A6617, A6618, A6619) RSA SigVer (FIPS186-2): (A6605, A6617, A6618, A6619) RSA SigVer (FIPS186-4):

Name	Type	Description	Properties	Algorithms
				(A6605, A6617, A6618, A6619) RSA SigVer (FIPS186-5): (A6605, A6606, A6617, A6618, A6619) EDDSA SigVer: (A6328)
Asymmetric Encryption	AsymKeyPair-Encap	Asymmetric encryption using RSA		KTS-IFC: (A6605, A6617, A6618, A6619)
Asymmetric Decryption	AsymKeyPair-Decap	Asymmetric decryption using RSA		KTS-IFC: (A6605, A6617, A6618, A6619)
Key pair generation	AsymKeyPair-KeyGen CKG	Generate a key pair		Safe Primes Key Generation: (A6602) ECDSA KeyGen (FIPS186-5): (A6605, A6617, A6618, A6619) RSA KeyGen (FIPS186-5): (A6605, A6617, A6618, A6619) EDDSA KeyGen: (A6328) Asymmetric Cryptographic Key Generation (CKG): () Key type: Asymmetric
Key pair verification	AsymKeyPair-KeyVer	Verify a key pair		Safe Primes Key Verification: (A6602) ECDSA KeyVer (FIPS186-5): (A6605, A6617, A6618, A6619)

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 AES-GCM IV

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. The module is compliant with SP 800-52 Rev. 2 Section 3.3.1 and the mechanism for IV generation is compliant with RFC 5288 and 8446.

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

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

Alternatively, the Crypto Officer can use the module's API to perform AES GCM encryption using internal IV generation. These IVs are always 96 bits and generated using the approved DRBG internal to the module's boundary, compliant to Scenario 2 of FIPS 140-3 IG C.H.

The module also provides a non-approved AES GCM encryption service which accepts arbitrary external IVs from the operator. This service can be requested by invoking the EVP_EncryptInit_ex2 API function with a non-NULL IV value. When this is the case, the API will set a non-approved service indicator.

Finally, for TLS 1.3, the AES GCM implementation uses the context of Scenario 5 of FIPS 140-3 IG C.H. The protocol that provides this compliance is TLS 1.3, defined in RFC8446 of August 2018, using the cipher-suites that explicitly select AES GCM as the encryption/decryption cipher (Appendix B.4 of RFC8446). The module supports acceptable AES GCM cipher suites from Section 3.3.1 of SP 800-52 Rev. 2. 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.

2.7.2 AES-XTS

The length of a single data unit encrypted or decrypted with AES XTS shall not exceed 2^{20} AES blocks, that is 16MB, of data per XTS instance. An XTS instance is defined in Section 4 of SP 800-38E.

To meet the requirement stated in IG C.I, the module implements a check that ensures, before performing any cryptographic operation, that the two AES keys used in AES XTS mode are not identical. Key_1 and Key_2 shall be generated and/or established independently according to the rules for component symmetric keys from NIST SP 800-133r2, Section 6.3.

The XTS mode shall only be used for the cryptographic protection of data on storage devices. It shall not be used for other purposes, such as the encryption of data in transit.

2.7.3 PBKDF2

The module provides password-based key derivation (PBKDF2), compliant with SP 800-132. The module supports option 1a from Section 5.4 of SP 800-132, in which the Master Key (MK) or a segment of it is used directly as the Data Protection Key (DPK).

In accordance with SP 800-132 and FIPS 140-3 IG D.N, the following requirements must be met:

- Derived keys shall be used only for storage applications and shall not be used for any other purposes. The length of the MK or DPK must be 112 bits or more.
- 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 Crypto Officer is responsible for choosing a sufficiently complex password. Assuming upper case, lower case, and numerical characters are used, the probability of guessing the password is approximately 1 / (62^8). Combined with the minimum iteration count as described below, this provides an acceptable trade-off between user experience and security against brute-force attacks.

- A portion of the salt shall be generated randomly using the SP 800-90Ar1 DRBG implemented by the module. The minimum length accepted by the module is 128 bits.
- The iteration count shall be selected as large as possible, as long as the time required to generate the key using the password is acceptable to the user. The minimum value accepted by the module is 1000.

If any of these requirements are not met, the requested service is non-approved.

2.7.4 SP 800-56Ar3 Assurances

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

The module's shared secret computation service will internally perform the full public key validation of the peer public key, complying with Sections 5.6.2.2.1 and 5.6.2.2.2 of SP 800-56Ar3.

2.7.5 SP 800-56Br2 Assurances

To comply with the assurances found in Section 6.4 of SP 800-56Br2, the operator must use the module in the context of the TLS or SSH protocols. Additionally, the module's approved key pair generation service (see Section 4.3) must be used to generate RSA 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 key pair validation of the generated public key.

The operator must use the EVP_PKEY_public_check() API to perform partial public key validation of the peer public key, complying with Section 6.4.2.2 of SP 800-56Br2. The operator must also confirm the peer's possession of private key by using any method specified in Section 6.4.2.3 of SP 800-56Br2.

2.7.6 RSA

For RSA key generation, signature generation, and signature verification, the approved modulus sizes of 2048, 3072, and 4096 bits are CAVP tested in compliance with FIPS 186-5. For KAS-IFC-SSC and KTS-IFC, the approved modulus sizes of 2048, 3072, 4096, 6144, 8192 are CAVP tested in compliance with SP 800-56Br2.

All other RSA modulus sizes listed in this document, and not mentioned above, cannot be tested by CAVP but are approved for RSA key generation, signature generation, and signature verification per FIPS 140-3 IG C.F, and KAS-IFC-SSC and KTS-IFC per SP 800-56Br2.

2.7.7 Legacy Use

RSA signature verification using modulus sizes between 1024 and 2048 bits is allowed for legacy use only. These legacy algorithms can only be used on data that was generated prior to the Legacy Date specified in FIPS 140-3 IG C.M.

2.7.8 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.

2.7.9 Key Transport

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

2.8 RBG and Entropy

Cert Number	Vendor Name
E208	Ctrl IQ, Inc.

Table 9: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Rocky Linux OpenSSL 3 CPU Time Jitter RNG Entropy Source	Non-Physical	Rocky Linux 9 on Intel Kaby Lake Xeon E3-1270 v6	256 bits	full entropy	SHA3-256 (A5837); SHA2-512-HMAC-DRBG (A5837); AES-256-CTR-DRBG (A5957)

Table 10: Entropy Sources

The module implements primary DRBG (AES-256-CTR-DRBG (A5957)) which acts as the conditioning component for the entropy source mentioned in the above table. It is only used internally by the module to seed the secondary DRBGs which can be of type (CTR, Hash, HMAC). The module complies with the Public Use Document for ESV certificate E208 by reading entropy data from the EVP_RAND_generate() function of the primary DRBG, which corresponds to the GetEntropy() conceptual interface. The operational environment on the ESV certificate is identical to the operating system described in this document. There are no maintenance requirements for the entropy source.

As per the Public document of entropy certificate E208, the entropy source provides full entropy of 256 bits.

When the module needs random data for internal purposes it uses two separate instances of AES-256 CTR_DRBG DRBG based on use case. i.e., it uses the “private DRBG” accessed via RAND_priv_bytes () for asymmetric key generation, signature generation, or other SSP use cases and it uses the “public DRBG” accessed via RAND_bytes() when it needs to generate IV or other non-SSP use cases.

When an external caller needs the random data, it can access it via “Random Number Generation” service of the module and it has a choice to choose between Hash, HMAC or CTR DRBG listed in the algorithms table.

2.9 Key Generation

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

- Safe primes key pair generation: compliant with SP 800-133r2, Section 5.2, which maps to SP 800-56Ar3. The method described in Section 5.6.1.1.4 of SP 800-56Ar3 (“Testing Candidates”) is used.
- EC (ECDH and ECDSA) key pair generation: compliant with SP 800-133r2, Section 5.1, which maps to FIPS 186-5. The method described in Appendix A.2.2 of FIPS 186-5 (“Rejection Sampling”) is used.

- EdDSA key pair generation: compliant with SP 800-133r2, Section 5.1, which maps to FIPS 186-5. The method described in Appendix A.2.3 of FIPS 186-5 is used.
- RSA key pair generation: compliant with SP 800-133r2, Section 5.1, which maps to FIPS 186-5. The method described in Appendix A.1.6 of FIPS 186-5 (“Probable Primes with Conditions Based on Auxiliary Probable Primes”) is used.

Additionally, the module implements the following key derivation methods:

- KBKDF: compliant with SP 800-108r1. This implementation can be used to generate secret keys from a pre-existing key-derivation-key.
- KDA OneStep, HKDF: compliant with SP 800-56Cr2.
- ANS X9.42 KDF, ANS X9.63 KDF: compliant with SP 800-135r1. These implementations shall only be used to generate secret keys in the context of an ANS X9.42-2001 resp. ANS X9.63-2001 key agreement scheme.
- SSH KDF, TLS 1.2 KDF, TLS 1.3 KDF: compliant with SP 800-135r1. These implementations shall only be used to generate secret keys in the context of the SSH, TLS 1.2, or TLS 1.3 protocols, respectively.
- PBKDF2: compliant with option 1a of SP 800-132. This implementation shall only be used to derive keys for use in storage applications.

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

2.10 Key Establishment

The module implements shared secret computation methods, asymmetric encryption and decryption services using RSA with OAEP padding, as listed in the Security Function Implementations table in Section 2.6.

2.11 Industry Protocols

The module implements key derivation functions for usage in the SSH (RFC 4253), TLS 1.2 (RFC 5288), and TLS 1.3 (RFC 8446) protocols. AES-GCM with internal IV generation is offered in the approved mode compliant with TLS 1.2 and TLS 1.3. Finally, the module supports the use of the safe primes defined in RFC 3526 (IKE) and RFC 7919 (TLS) for Diffie-Hellman.

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

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

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

Table 11: Ports and Interfaces

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

4 Roles, Services, and Authentication

4.1 Authentication Methods

The module does not implement any authentication methods.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 12: Roles

No support is provided for multiple concurrent operators.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Message digest	Compute a message digest	EVP_DigestFinal_ex returns 1	Message	Digest value	Message digest	Crypto Officer
XOF	Compute an extendable output message digest	EVP_DigestFinalXOF returns 1	Message, output length	Digest value	XOF	Crypto Officer
Encryption	Encrypt a plaintext	EVP_EncryptFinal_ex returns 1	AES key, plaintext, IV (if required)	Ciphertext	Encryption	Crypto Officer - AES key: W,E
Decryption	Decrypt a ciphertext	EVP_DecryptFinal_ex returns 1	AES key, ciphertext, IV (if required)	Plaintext	Decryption	Crypto Officer - AES key: W,E
Authenticated encryption	Encrypt and authenticate a	AES-GCM: EVP_CIPHER_ROCKY_FIPS_INDICATOR_APPROVED; Others: EVP_EncryptFinal_ex returns 1	AES key, plaintext, IV	Ciphertext, MAC tag	Authenticated encryption	Crypto Officer - AES key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	plaintext					- DRBG internal state (V, Key): W,E
Authenticated decryption	Decrypt and authenticate a ciphertext	EVP_DecryptFinal_ex returns 1	AES key, ciphertext, IV, MAC tag	Plaintext or failure	Authenticated decryption	Crypto Officer - AES key: W,E
Message authentication	Compute a MAC tag	HMAC: OSSL_MAC_PARAM_ROCKY_FIPS_INDICATOR_APPROVED; Others: EVP_MAC_final returns 1	AES key or HMAC key, message	MAC tag	Message authentication	Crypto Officer - AES key: W,E - HMAC key: W,E
Key-based key derivation	Derive keying material from a key-derivation key	EVP_KDF_ROCKY_FIPS_INDICATOR_APPROVED	Key-derivation key, output length	Derived key	Key-based key derivation	Crypto Officer - Key-derivation key: W,E - Derived key: G,R
Key-establishment key derivation	Derive keying material from a shared secret	EVP_KDF_ROCKY_FIPS_INDICATOR_APPROVED	Shared secret, output length	Derived key	Key-establishment key derivation	Crypto Officer - Shared secret: W,E - Derived key: G,R
Protocol key derivation	Derive keying material from a shared secret	EVP_KDF_ROCKY_FIPS_INDICATOR_APPROVED	Shared secret, output length	Derived key	Protocol key derivation	Crypto Officer - Shared secret: W,E - Derived

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						key: G,R
Password-based key derivation	Derive keying material from a password	EVP_KDF_ROCKY_FIPS_INDICATOR_APPROVED	Password, salt, iteration count, output length	Derived key	Protocol key derivation	Crypto Officer - Password: W,E - Derived key: G,R
Random number generation	Generate random bytes	EVP_RAND_generate returns 1	Output length	Random bytes	Random number generation	Crypto Officer - Entropy input: G,E,Z - DRBG seed: G,E,Z - DRBG internal state (V, Key): G,W,E - DRBG internal state (V, C): G,W,E
Shared secret computation	Compute a shared secret	EVP_PKEY_ROCKY_FIPS_INDICATOR_APPROVED	Owner private key, peer public key	Shared secret	Shared secret computation	Crypto Officer - DRBG internal state (V, Key): W,E - DH private key: W,E - DH public key: W,E - EC private key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- EC public key: W,E - RSA private key: W,E - RSA public key: W,E - Shared secret: G,R
Signature generation	Generate a digital signature	OSSL_RL_FIPSINDICATOR_APPROVED and EVP_PKEY_ROCKY_FIPS_INDICATOR_APPROVED	Private key, hash algorithm, message	Signature	Signature generation	Crypto Officer - DRBG internal state (V, Key): W,E - EC private key: W,E - EdDSA private key: W,E - RSA private key: W,E
Signature verification	Verify a digital signature	OSSL_RL_FIPSINDICATOR_APPROVED and EVP_PKEY_ROCKY_FIPS_INDICATOR_APPROVED	Public key, hash algorithm, message, signature	Pass/fail	Signature verification	Crypto Officer - EC public key: W,E - EdDSA public key: W,E - RSA public

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						key: W,E
Asymmetric Encryption	Perform RSA-based encryption (compliant with SP 800-56B Rev. 2))	EVP_PKEY_ROCKY_FIPS_INDICATOR_APPROVED	RSA public key, plaintext	Ciphertext	Asymmetric Encryption	Crypto Officer - RSA public key: W,E
Asymmetric Decryption	Perform RSA-based decryption (compliant with SP 800-56B Rev. 2))	EVP_PKEY_ROCKY_FIPS_INDICATOR_APPROVED	RSA private key, ciphertext	Plaintext	Asymmetric Decryption	Crypto Officer - RSA private key: W,E
Key pair generation	Generate a key pair	EVP_PKEY_generate returns 1	Group or curve or modulus bits	Module-generated key pair	Key pair generation	Crypto Officer - DRBG internal state (V, Key): W,E - Module - generated DH private key: G,R - Module - generated DH public key:

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						G,R - Module - generated EC private key: G,R - Module - generated EC public key: G,R - Module - generated EdDSA private key: G,R - Module - generated EdDSA public key: G,R - Module - generated RSA private key: G,R - Module - generated RSA public key: G,R

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- Intermediate key generation value: G,E,Z
Key pair verification	Verify a key pair	EVP_PKEY_public_check or EVP_PKEY_private_check or EVP_PKEY_check returns 1	Key pair	Pass/fail	Key pair verification	Crypto Officer - DH private key: W,E - DH public key: W,E - EC private key: W,E - EC public key: W,E
Show version	Return the module name and version information	None	N/A	Module name and version	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 Decryption Authenticated encryption Authenticated decryption	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					Key-based key derivation Key-establishment key derivation Protocol key derivation Password-based key derivation Random number generation Shared secret computation Signature generation Signature verification	
Zeroization	Zeroize SSPs	None	Any SSP	N/A	None	Crypto Officer - AES key: Z - HMAC key: Z - Key-derivation key: Z - Shared secret: Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- Password: Z - Derived key: Z - Entropy input: Z - DRBG seed: Z - DRBG internal state (V, Key): Z - DRBG internal state (V, C): Z - DH private key: Z - DH public key: Z - EC private key: Z - EC public key: Z - EdDSA private key: Z - EdDSA public key: Z - RSA private key: Z - RSA public key: Z - Module - generat

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						ed DH private key: Z - Module - generated DH public key: Z - Module - generated EC private key: Z - Module - generated EC public key: Z - Module - generated EdDSA private key: Z - Module - generated EdDSA public key: Z - Module - generated RSA private key: Z - Module - generat

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						ed RSA public key: Z - Intermediate key generation value: Z

Table 13: Approved Services

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

To interact with the module, a calling application must use the EVP API layer provided by OpenSSL. This layer will delegate the request to the FIPS provider, which will in turn perform the requested service. Additionally, this EVP API layer can be used to retrieve the approved service indicator for the module. The rocky_ossl_query_fipsindicator() function indicates whether an EVP API function is approved. After a cryptographic service was performed by the module, the API context (listed in the left column of the table below) associated with this request can contain a parameter (listed in the right column of the table below) which represents the approved service indicator.

Context	Service Indicator
EVP_CIPHER_CTX	OSSL_CIPHER_PARAM_ROCKY_FIPS_INDICATOR
EVP_MAC_CTX	OSSL_MAC_PARAM_ROCKY_FIPS_INDICATOR
EVP_KDF_CTX	OSSL_KDF_PARAM_ROCKY_FIPS_INDICATOR
EVP_PKEY_CTX	OSSL_SIGNATURE_PARAM_ROCKY_FIPS_INDICATOR
EVP_PKEY_CTX	OSSL_ASYM_CIPHER_PARAM_ROCKY_FIPS_INDICATOR
EVP_PKEY_CTX	OSSL_KEM_PARAM_ROCKY_FIPS_INDICATOR

The details to use these functions and parameters are described in the module's manual pages. They can be accessed using the "man openssl" command.

4.4 Non-Approved Services

Name	Description	Algorithms	Role
AES-GCM with external IV encryption	Encrypt and authenticate a plaintext using AES-GCM with an external IV	AES-GCM with external IV	Crypto Officer

Name	Description	Algorithms	Role
Message authentication	Compute a MAC tag	HMAC with < 112-bit keys	Crypto Officer
Key-based key derivation	Derive keying material from a key-derivation key	KBKDF with < 112-bit keys	Crypto Officer
Key-establishment key derivation	Derive keying material from a shared secret	KDA OneStep, HKDF with < 112-bit keys KDA OneStep with SHAKE128, SHAKE256	Crypto Officer
Protocol key derivation	Derive keying material from a shared secret	ANS X9.42 KDF, ANS X9.63 KDF with < 112-bit keys ANS X9.42 KDF with SHAKE128, SHAKE256 ANS X9.63 KDF with SHA-1, SHAKE128, SHAKE256 SSH KDF with SHA-512/224, SHA-512/256, SHA-3, SHAKE128, SHAKE256 TLS 1.2 KDF with SHA-1, SHA-224, SHA-512/224, SHA-512/256, SHA-3 TLS 1.3 KDF with SHA-1, SHA-224, SHA-512, SHA-512/224, SHA-512/256, SHA-3	Crypto Officer
Password-based key derivation	Derive keying material from a password	PBKDF2 with short password, short salt, insufficient iterations, < 112-bit output keys	Crypto Officer
Signature generation	Generate a digital signature	RSA-PSS with invalid salt length RSA with no padding RSA and ECDSA with no hashing	Crypto Officer
Signature verification	Verify a digital signature	RSA-PSS with invalid salt length RSA with no padding RSA and ECDSA with no hashing	Crypto Officer

Table 14: Non-Approved Services

4.5 External Software/Firmware Loaded

The module does not load external software or firmware.

5 Software/Firmware Security

5.1 Integrity Techniques

The integrity of the module is verified by comparing a HMAC-SHA2-256 value calculated at run time with the HMAC-SHA2-256 value embedded in the fips.so file that was computed at build time. The module performs a KAT for the HMAC SHA-256 algorithm in order to test its proper operation before performing the checksum of the fips.so file.

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 can be invoked on demand by unloading and subsequently re-initializing the module (i.e., rebooting the system), which will perform (among others) the software integrity tests.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

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

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

6.2 Configuration Settings and Restrictions

The module shall be installed as stated in Section 11.1.

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

7 Physical Security

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

8 Non-Invasive Security

The module does not implement any non-invasive security mechanisms.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
Module RAM	Temporary storage for SSPs used by the module as part of service execution	Dynamic

Table 15: Storage Areas

The module does not perform persistent storage of SSPs; SSPs in use by the module exist in volatile memory only. SSPs are provided to the module by the calling process and are destroyed when released by the appropriate zeroization function calls.

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
API input parameters	Operator calling application (TOEPP)	Module RAM	Plaintext	Manual	Electronic	
API output parameters	Module RAM	Operator calling application (TOEPP)	Plaintext	Manual	Electronic	

Table 16: SSP Input-Output Methods

The module only supports SSP entry and output to and from the calling application running on the same operational environment. This corresponds to manual distribution, electronic entry/output (“CM Software to/from App via TOEPP Path”) per FIPS 140-3 IG 9.5.A Table 1.

There is no entry or output of cryptographically protected SSPs.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Free cipher handle	Zeroizes the SSPs contained within the cipher handle	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 appropriate zeroization functions: EVP_CIPHER_CTX_free, EVP_MAC_CTX_free, EVP_KDF_CTX_free, EVP_RAND_CTX_free, EVP_PKEY_free
Module reset	De-allocates the volatile memory used to store SSPs	Volatile memory used by the module is overwritten within nanoseconds when power is removed. The successful completion of the module reset indicates that zeroization has completed.	By unloading and reloading the module

Zeroization Method	Description	Rationale	Operator Initiation
Automatic	Automatically zeroized by the module when no longer needed	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable. The successful completion of the running service indicates that zeroization was completed.	N/A

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
AES key	Symmetric key used for AES operations	128, 256 bits (AES-XTS); 128, 192, 256 bits (others) - 128, 256 bits (AES-XTS); 128, 192, 256 bits (others)	Symmetric key - CSP			Encryption Decryption Authenticated encryption Authenticated decryption Message authentication
HMAC key	Symmetric key used for HMAC operations	112-524288 bits - 112-256 bits	Authentication key - CSP			Message authentication
Key-derivation key	Symmetric key used to derive symmetric keys	112-4096 bits - 112-256 bits	Symmetric key - CSP			Key-based key derivation
Shared secret	Shared secret established using KAS-SSC	112-8192 bits - 112-256 bits	Shared secret - CSP		Shared secret computation	Key-establishment key derivation Protocol key derivation
Password	Password used to derive symmetric keys	8-128 characters - N/A	Password - CSP			Password-based key derivation
Derived key	Symmetric key derived from a key-derivation key, shared	112-4096 bits - 112-256 bits	Symmetric key - CSP	Key-based key derivation Key-establishme		

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	secret, or password			nt key derivation Protocol key derivation Password-based key derivation		
Entropy input	Entropy input used to seed DRBGs	128-384 bits - 128-384 bits	Entropy input - CSP	Random number generation		Random number generation
DRBG seed	DRBG seed derived from entropy input and additional data	CTR_DRBG: 256, 320, 384 bits; Hash_DRBG: 440, 888 bits; HMAC_DRBG: 440, 888 bits - CTR_DRBG: 128, 192, 256 bits; Hash_DRBG: 128, 256 bits; HMAC_DRBG: 128, 256 bits	Seed - CSP	Random number generation		Random number generation
DRBG internal state (V, Key)	Internal state of CTR_DRBG and HMAC_DRBG	CTR_DRBG: 256, 320, 384 bits; HMAC_DRBG: 320, 512, 1024 bits - CTR_DRBG: 128, 192, 256 bits; HMAC_DRBG: 128, 256 bits	Internal state - CSP	Random number generation		Random number generation
DRBG internal state (V, C)	Internal state of Hash_DRBG	Hash_DRBG: 880, 1776 bits - Hash_DRBG: 128, 256 bits	Internal state - CSP	Random number generation		Random number generation
DH private key	Private key used for Diffie-Hellman	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096,	Private key - CSP			Shared secret computation Key pair verification

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		MODP-6144, MODP-8192 - 112-200 bits				
DH public key	Public key used for Diffie-Hellman	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 - 112-200 bits	Public key - PSP			Shared secret computation Key pair verification
EC private key	Private key used for EC Diffie-Hellman and ECDSA	P-224, P-256, P-384, P-521 - 112, 128, 192, 256 bits	Private key - CSP			Shared secret computation Signature generation Key pair verification
EC public key	Public key used for EC Diffie-Hellman and ECDSA	P-224, P-256, P-384, P-521 - 112, 128, 192, 256 bits	Public key - PSP			Shared secret computation Signature verification Key pair verification
EdDSA private key	Private key used for EdDSA	Ed25519, Ed448 - 128, 224 bits	Private key - CSP			Signature generation
EdDSA public key	Public key used for EdDSA	Ed25519, Ed448 - 128, 224 bits	Public key - PSP			Signature verification
RSA private key	Private key used for RSA	2048-16384 bits - 112-256 bits	Private key - CSP			Shared secret computation Signature generation Asymmetric Decryption
RSA public key	Public key used for RSA	1024, 1536, 2048-16384 bits - 80, 96, 112-256 bits	Public key - PSP			Shared secret computation Signature verification Asymmetric Encryption
Module-generated	DH private key	ffdhe2048, ffdhe3072,	Private key - CSP	Key pair generation		

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DH private key	generated by the module	ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 - 112-200 bits				
Module-generated DH public key	DH public key generated by the module	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 - 112-200 bits	Public key - PSP	Key pair generation		
Module-generated EC private key	EC private key generated by the module	P-224, P-256, P-384, P-521 - 112, 128, 192, 256 bits	Private key - CSP	Key pair generation		
Module-generated EC public key	EC public key generated by the module	P-224, P-256, P-384, P-521 - 112, 128, 192, 256 bits	Public key - PSP	Key pair generation		
Module-generated EdDSA private key	EdDSA private key generated by the module	Ed25519, Ed448 - 128, 224 bits	Private key - CSP	Key pair generation		
Module-generated EdDSA public key	EdDSA public key generated by the module	Ed25519, Ed448 - 128, 224 bits	Public key - PSP	Key pair generation		
Module-generated RSA private key	RSA private key generated by the module	2048-15360 bits - 112-256 bits	Private key - CSP	Key pair generation		
Module-generated RSA public key	RSA public key generated by the module	2048-15360 bits - 112-256 bits	Public key - PSP	Key pair generation		

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Intermediate key generation value	Temporary value generated during key pair generation services	224-15360 bits - 112-256 bits	Intermediate value - CSP	Key pair generation		Key pair generation

Table 18: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES key	API input parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	
HMAC key	API input parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	
Key-derivation key	API input parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	Derived key:Derives
Shared secret	API input parameters API output parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	DH private key:Established By DH public key:Established By EC private key:Established By EC public key:Established By RSA private key:Established By RSA public key:Established By Derived key:Derives
Password	API input parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	Derived key:Derives
Derived key	API output parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	Key-derivation key:Derived From Shared secret:Derived From Password:Derived From

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Entropy input		Module RAM:Plaintext	From generation until DRBG seed is created	Module reset Automatic	DRBG seed:Derives
DRBG seed		Module RAM:Plaintext	While the DRBG is being instantiated	Module reset Automatic	Entropy input:Derived From DRBG internal state (V, Key):Generates DRBG internal state (V, C):Generates
DRBG internal state (V, Key)		Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	DRBG seed:Generated From
DRBG internal state (V, C)		Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	DRBG seed:Generated From
DH private key	API input parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	DH public key:Paired With Shared secret:Establishes
DH public key	API input parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	DH private key:Paired With Shared secret:Establishes
EC private key	API input parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	EC public key:Paired With Shared secret:Establishes
EC public key	API input parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	EC private key:Paired With Shared secret:Establishes
EdDSA private key	API input parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	EdDSA public key:Paired With
EdDSA public key	API input parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	EdDSA private key:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
RSA private key	API input parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	RSA public key:Paired With Shared secret:Establishes
RSA public key	API input parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	RSA private key:Paired With Shared secret:Establishes
Module-generated DH private key	API output parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	Module-generated DH public key:Paired With Intermediate key generation value:Generated From
Module-generated DH public key	API output parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	Module-generated DH private key:Paired With Intermediate key generation value:Generated From
Module-generated EC private key	API output parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	Module-generated EC public key:Paired With Intermediate key generation value:Generated From
Module-generated EC public key	API output parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	Module-generated EC private key:Paired With Intermediate key generation value:Generated From
Module-generated EdDSA private key	API output parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	Module-generated EdDSA public key:Paired With Intermediate key generation value:Generated From
Module-generated EdDSA public key	API output parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	Module-generated EdDSA private key:Paired With Intermediate key generation value:Generated From

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Module-generated RSA private key	API output parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	Module-generated RSA public key:Paired With Intermediate key generation value:Generated From
Module-generated RSA public key	API output parameters	Module RAM:Plaintext	Until cipher handle is freed or module is reset	Free cipher handle Module reset	Module-generated RSA private key:Paired With Intermediate key generation value:Generated From
Intermediate key generation value		Module RAM:Plaintext	For the duration of the service	Module reset Automatic	Module-generated DH private key:Generates Module-generated DH public key:Generates Module-generated EC private key:Generates Module-generated EC public key:Generates Module-generated EdDSA private key:Generates Module-generated EdDSA public key:Generates Module-generated RSA private key:Generates Module-generated RSA public key:Generates

Table 19: SSP Table 2

9.5 Transitions

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

10 Self-Tests

While the module is executing the self-tests, services are not available, and data output (via the data output interface) is inhibited until the tests are successfully completed. The module does not return control to the calling application until the tests are completed. If any of the self-tests fails, the module immediately transitions to the error state.

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256	Key size: 256 bits	Message authentication	SW/FW Integrity	Module becomes available and services are available for use	Integrity test of the fips.so shared library

Table 20: Pre-Operational Self-Tests

The pre-operational software integrity tests are performed automatically when the module is powered on, before the module transitions into the operational state.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-GCM - Encryption	Key size: 256 bits	KAT	CAST	Module becomes operational	Encryption	At power-on, before the integrity test
AES-GCM - Decryption	Key size: 256 bits	KAT	CAST	Module becomes operational	Decryption	At power-on, before the integrity test
AES-ECB	Key size: 128 bits	KAT	CAST	Module becomes operational	Decryption	At power-on, before the integrity test
SHA-1	24-bit message	KAT	CAST	Module becomes operational	Message digest	At power-on, before the integrity test
SHA2-512	24-bit message	KAT	CAST	Module becomes operational	Message digest	At power-on, before the integrity test
SHA3-256	32-bit message	KAT	CAST	Module becomes operational	Message digest	At power-on, before the integrity test
KBKDF	HMAC-SHA2-256 in counter mode	KAT	CAST	Module becomes operational	Key derivation	At power-on, before the integrity test
KDA OneStep	SHA2-224	KAT	CAST	Module becomes operational	Key derivation	At power-on, before the integrity test
KDA HKDF	SHA2-256	KAT	CAST	Module becomes operational	Key derivation	At power-on, before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
X9.42 KDF	SHA-1	KAT	CAST	Module becomes operational	Key derivation	At power-on, before the integrity test
X9.63 KDF	SHA2-256	KAT	CAST	Module becomes operational	Key derivation	At power-on, before the integrity test
SSH KDF	SHA-1	KAT	CAST	Module becomes operational	Key derivation	At power-on, before the integrity test
TLS 1.2 KDF	SHA2-256	KAT	CAST	Module becomes operational	Key derivation	At power-on, before the integrity test
TLS 1.3 KDF	SHA2-256	KAT	CAST	Module becomes operational	Key derivation	At power-on, before the integrity test
PBKDF2	SHA-256 with 4096 iterations, 24-byte password, and 288-bit salt	KAT	CAST	Module becomes operational	Key derivation	At power-on, before the integrity test
Counter DRBG	AES-128 with prediction resistance, with derivation function	KAT	CAST	Module becomes operational	Instantiate, seed, generate, reseed, generate (compliant to SP 800-90Ar1 Section 11.3)	At power-on, before the integrity test
Hash DRBG	SHA2-256 with prediction resistance	KAT	CAST	Module becomes operational	Instantiate, seed, generate, reseed, generate (compliant to SP 800-90Ar1 Section 11.3)	At power-on, before the integrity test
HMAC DRBG	HMAC-SHA-1 with prediction resistance	KAT	CAST	Module becomes operational	Instantiate, seed, generate, reseed, generate (compliant to SP 800-90Ar1 Section 11.3)	At power-on, before the integrity test
KAS-FFC-SSC	ffdhe2048	KAT	CAST	Module becomes operational	Shared secret computation	At power-on, before the integrity test
KAS-ECC-SSC	P-256	KAT	CAST	Module becomes operational	Shared secret computation	At power-on, before the integrity test
ECDSA SigGen (FIPS186-5)	SHA-256 and P-224, P-256, P-384, P-521	KAT	CAST	Module becomes operational	Signature generation	At power-on, before the integrity test
ECDSA SigVer (FIPS186-5)	SHA-256 and P-224, P-256, P-384, P-521	KAT	CAST	Module becomes operational	Signature verification	At power-on, before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
EdDSA SigGen	Ed25519, Ed448	KAT	CAST	Module becomes operational	Signature generation	At power-on, before the integrity test
EdDSA SigVer	Ed25519, Ed448	KAT	CAST	Module becomes operational	Signature verification	At power-on, before the integrity test
RSA PKCS#1 v1.5 SigGen	SHA-256 and 2048-bit key	KAT	CAST	Module becomes operational	Signature generation	At power-on, before the integrity test
RSA PKCS#1 v1.5 SigVer	SHA-256 and 2048-bit key	KAT	CAST	Module becomes operational	Signature verification	At power-on, before the integrity test
DH	N/A	PCT	PCT	Key pair generation is successful	SP 800-56Ar3 Section 5.6.2.1.4	Key pair generation
ECDSA KeyGen (FIPS186-5)	SHA2-256	PCT	PCT	Key pair generation is successful	Signature generation and verification	Key pair generation
EdDSA KeyGen	N/A	PCT	PCT	Key pair generation is successful	Signature generation and verification	Key pair generation
RSA PKCS#1 v1.5 KeyGen	SHA2-256	PCT	PCT	Key pair generation is successful	Signature generation and verification	Key pair generation

Table 21: Conditional Self-Tests

10.3 Periodic Self-Test Information

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

Table 22: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-GCM - Encryption	KAT	CAST	On demand	Manually
AES-GCM - Decryption	KAT	CAST	On demand	Manually
AES-ECB	KAT	CAST	On demand	Manually
SHA-1	KAT	CAST	On demand	Manually
SHA2-512	KAT	CAST	On demand	Manually
SHA3-256	KAT	CAST	On demand	Manually
KBKDF	KAT	CAST	On demand	Manually
KDA OneStep	KAT	CAST	On demand	Manually
KDA HKDF	KAT	CAST	On demand	Manually
X9.42 KDF	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
X9.63 KDF	KAT	CAST	On demand	Manually
SSH KDF	KAT	CAST	On demand	Manually
TLS 1.2 KDF	KAT	CAST	On demand	Manually
TLS 1.3 KDF	KAT	CAST	On demand	Manually
PBKDF2	KAT	CAST	On demand	Manually
Counter DRBG	KAT	CAST	On demand	Manually
Hash DRBG	KAT	CAST	On demand	Manually
HMAC DRBG	KAT	CAST	On demand	Manually
KAS-FFC-SSC	KAT	CAST	On demand	Manually
KAS-ECC-SSC	KAT	CAST	On demand	Manually
ECDSA SigGen (FIPS186-5)	KAT	CAST	On demand	Manually
ECDSA SigVer (FIPS186-5)	KAT	CAST	On demand	Manually
EdDSA SigGen	KAT	CAST	On demand	Manually
EdDSA SigVer	KAT	CAST	On demand	Manually
RSA PKCS#1 v1.5 SigGen	KAT	CAST	On demand	Manually
RSA PKCS#1 v1.5 SigVer	KAT	CAST	On demand	Manually
DH	PCT	PCT	On demand	Manually
ECDSA KeyGen (FIPS186-5)	PCT	PCT	On demand	Manually
EdDSA KeyGen	PCT	PCT	On demand	Manually
RSA PKCS#1 v1.5 KeyGen	PCT	PCT	On demand	Manually

Table 23: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Power-up error	An error occurred during the self-tests executed on power-up	Software integrity test failure CAST failure	Module reinitialization	Module will not load (OSSL_PROV_PARAM_STATUS is set to 0)
PCT error	An error occurred during a PCT	PCT failure	Module reinitialization	Module stops functioning (aborts)

Table 24: Error States

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

10.5 Operator Initiation of Self-Tests

The pre-operational self-tests and CASTs can be invoked on demand by unloading and subsequently re-initializing the module. The PCTs can be invoked on demand by requesting the key pair generation service.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

Before the openssl-fips-provider-so-3.0.7-27.el9_2.ciqfips.0.2.7.x86_64 RPM package is 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 validated configuration after installation. Execute the “fips- mode-setup --enable” command. Restart the system.

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

11.2 Administrator Guidance

After installation of the openssl-fips-provider-so-3.0.7-27.el9_2.ciqfips.0.2.7.x86_64 RPM package, the Crypto Officer must execute the “openssl list -providers” command. The Crypto Officer must ensure that the FIPS provider is listed in the output as follows:

fips
name: Rocky Linux 9 - OpenSSL FIPS Provider
version: Rocky9.20250210
status: active

The cryptographic boundary consists only of the FIPS provider as listed. If any other OpenSSL or third-party provider is invoked, the user is not interacting with the module specified in this Security Policy.

11.3 Non-Administrator Guidance

There is no non-administrator guidance.

11.4 Design and Rules

Not applicable.

11.5 Maintenance Requirements

Not applicable.

11.6 End of Life

As the module does not persistently store SSPs, secure sanitization of the module consists of unloading the module. This will zeroize all SSPs in volatile memory. Then, if desired, the openssl-fips-provider-so-3.0.7-27.el9_2.ciqfips.0.2.7.x86_64 RPM package can be uninstalled from the Rocky Linux 9 system.

12 Mitigation of Other Attacks

Certain cryptographic subroutines and algorithms are vulnerable to timing analysis. The module mitigates this vulnerability by using constant-time implementations. This includes, but is not limited to:

- Big number operations: computing GCDs, modular inversion, multiplication, division, and modular exponentiation (using Montgomery multiplication)
- Elliptic curve point arithmetic: addition and multiplication (using the Montgomery ladder)
- EdDSA implementations
- Vector-based AES implementations

In addition, RSA, ECDSA, ECDH, and DH employ blinding techniques to further impede timing and power analysis. No configuration is needed to enable the aforementioned countermeasures.

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
CBC-CS	Cipher Block Chaining with Ciphertext Stealing
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CKG	Cryptographic Key Generation
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter
CVL	Component Validation List
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
EC	Elliptic Curve
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EdDSA	Edwards Curve Digital Signature Algorithm
ESV	Entropy Source Validation
EVP	Envelope
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
GMAC	Galois Counter Mode Message Authentication Code
HKDF	HMAC-based Key Derivation Function
HMAC	Keyed-Hash Message Authentication Code
IFC	Integer Factorization Cryptography
IG	Implementation Guidance
IKE	Internet Key Exchange
IV	Initialization Vector
KAS	Key Agreement Scheme
KAT	Known Answer Test
KBKDF	Key-based Key Derivation Function
KDA	Key Derivation Algorithm
KDF	Key Derivation Function
KTS	Key Transport Scheme
KW	Key Wrap
KWP	Key Wrap with Padding
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
OAEP	Optimal Asymmetric Encryption Padding
OFB	Output Feedback
PAA	Processor Algorithm Acceleration
PBKDF	Password-Based Key Derivation Function
PCT	Pair-wise Consistency Test
PKCS	Public-Key Cryptography Standards
PSP	Public Security Parameter
PSS	Probabilistic Signature Scheme
RSA	Rivest Shamir Adleman

SHA	Secure Hash Algorithm
SSC	Shared Secret Computation
SSH	Secure Shell
SSP	Sensitive Security Parameter
TOEPP	Tested Operational Environment's Physical Perimeter
XOF	Extendable Output Function
XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

B References

ANSI X9.42-2001	**Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography** 2001 https://webstore.ansi.org/standards/ascx9/ansix9422001
ANSI X9.63-2001	**Public Key Cryptography for the Financial Services Industry, Key Agreement and Key Transport Using Elliptic Curve Cryptography** 2001 https://webstore.ansi.org/standards/ascx9/ansix9632001
FIPS 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** https://csrc.nist.gov/CSRC/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/FIPS%20140-3%20IG.pdf
FIPS 180-4	**Secure Hash Standard (SHS)** August 2015 https://doi.org/10.6028/NIST.FIPS.180-4
FIPS 186-2	**Digital Signature Standard (DSS)** January 2000 https://doi.org/10.6028/NIST.FIPS.186-2
FIPS 186-4	**Digital Signature Standard (DSS)** July 2013 https://doi.org/10.6028/NIST.FIPS.186-4
FIPS 186-5	**Digital Signature Standard (DSS)** February 2023 https://doi.org/10.6028/NIST.FIPS.186-5
FIPS 197	**Advanced Encryption Standard (AES)** November 2001; Updated 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 202	**SHA-3 Standard: Permutation-Based Hash and Extendable- Output Functions** August 2015 https://doi.org/10.6028/NIST.FIPS.202
PKCS#1	**PKCS #1: RSA Cryptography Specifications Version 2.2** November 2016 https://doi.org/10.17487/RFC8017
RFC 3526	**More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)** May 2003 https://doi.org/10.17487/RFC3526
RFC 4253	**The Secure Shell (SSH) Transport Layer Protocol** January 2006 https://doi.org/10.17487/RFC4253
RFC 5288	**AES Galois Counter Mode (GCM) Cipher Suites for TLS** August 2008 https://doi.org/10.17487/RFC5288
RFC 7627	**Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension** September 2015

https://doi.org/10.17487/RFC7627

RFC 7919 **Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)**
August 2016
https://doi.org/10.17487/RFC7919

RFC 8446 **The Transport Layer Security (TLS) Protocol Version 1.3**
August 2018
https://doi.org/10.17487/RFC8446

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-Add **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; Updated October 2016
https://doi.org/10.6028/NIST.SP.800-38B

SP 800-38C **Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality**
May 2004; Updated July 2007
https://doi.org/10.6028/NIST.SP.800-38C

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

SP 800-38E **Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices**
January 2010
https://doi.org/10.6028/NIST.SP.800-38E

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 800-52r2 **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 800-56Ar3 **Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography**
April 2018
https://doi.org/10.6028/NIST.SP.800-56Ar3

SP 800-56Br2 **Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography**
March 2019
https://doi.org/10.6028/NIST.SP.800-56Br2

SP 800-56Cr2 **Recommendation for Key-Derivation Methods in Key-Establishment Schemes**
August 2020
https://doi.org/10.6028/NIST.SP.800-56Cr2

SP 800-90Ar1 **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 800-108r1	**Recommendation for Key Derivation Using Pseudorandom Functions** August 2022 https://doi.org/10.6028/NIST.SP.800-108r1-upd1
SP 800-131Ar2	**Transitioning the Use of Cryptographic Algorithms and Key Lengths** Marcy 2019 https://doi.org/10.6028/NIST.SP.800-131Ar2
SP 800-132	**Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** December 2010 https://doi.org/10.6028/NIST.SP.800-132
SP 800-133r2	**Recommendation for Cryptographic Key Generation** June 2020 https://doi.org/10.6028/NIST.SP.800-133r2
SP 800-135r1	**Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 https://doi.org/10.6028/NIST.SP.800-135r1
SP 800-140Br1	**Cryptographic Module Validation Program (CMVP) Security Policy Requirements: CMVP Validation Authority Updates to ISO/IEC 24759 and ISO/IEC 19790 Annex B** November 2023 https://doi.org/10.6028/NIST.SP.800-140Br1