Ctrl IQ, Inc.

Rocky Linux 8 and 9 GnuTLS 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: 1.2

Last update: 2026-04-07

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 versions rocky8.20241217 and rocky9.20241217 of the Rocky Linux 8 and 9 GnuTLS 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	N/A
	Overall Level	1

Table 1: Security Levels

1.3 Additional Information

This Security Policy describes the features and design of the module named Rocky Linux 8 and 9 GnuTLS Cryptographic Module using the terminology contained in the FIPS 140-3 specification. The FIPS 140-3 Security Requirements for Cryptographic Module specifies the security requirements that will be satisfied by a cryptographic module utilized within a security system protecting sensitive but unclassified information. The NIST/CCCS Cryptographic Module Validation Program (CMVP) validates cryptographic module to FIPS 140-3. Validated products are accepted by the Federal agencies of both the USA and Canada for the protection of sensitive or designated information.

In preparing the Security Policy document, the laboratory formatted the vendor-supplied documentation for consolidation without altering the technical statements therein contained. The further refining of the Security Policy document was conducted iteratively throughout the conformance testing, wherein the Security Policy

was submitted to the vendor, who would then edit, modify, and add technical contents. The vendor would also supply additional documentation, which the laboratory formatted into the existing Security Policy, and resubmitted to the vendor for their final editing.

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The Rocky Linux 8 and 9 GnuTLS Cryptographic Module (hereafter referred to as "the module") is a software library implementing general purpose cryptographic algorithms. The module provides cryptographic services to applications running in the user space of the underlying operating system through an application program interface (API).

Module Type: Software

Module Embodiment: MultiChipStand

Cryptographic Boundary:
The block diagram in Figure 1 shows the cryptographic boundary of the module, its interfaces with the operational environment and the flow of information between the module and operator (depicted through the arrows).

Tested Operational Environment's Physical Perimeter (TOEPP):
The TOEPP is the general-purpose computer on which the module is installed. The module makes use of an SP800-90B-compliant Entropy Source (described in Section 2.8) located within the TOEPP.
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.

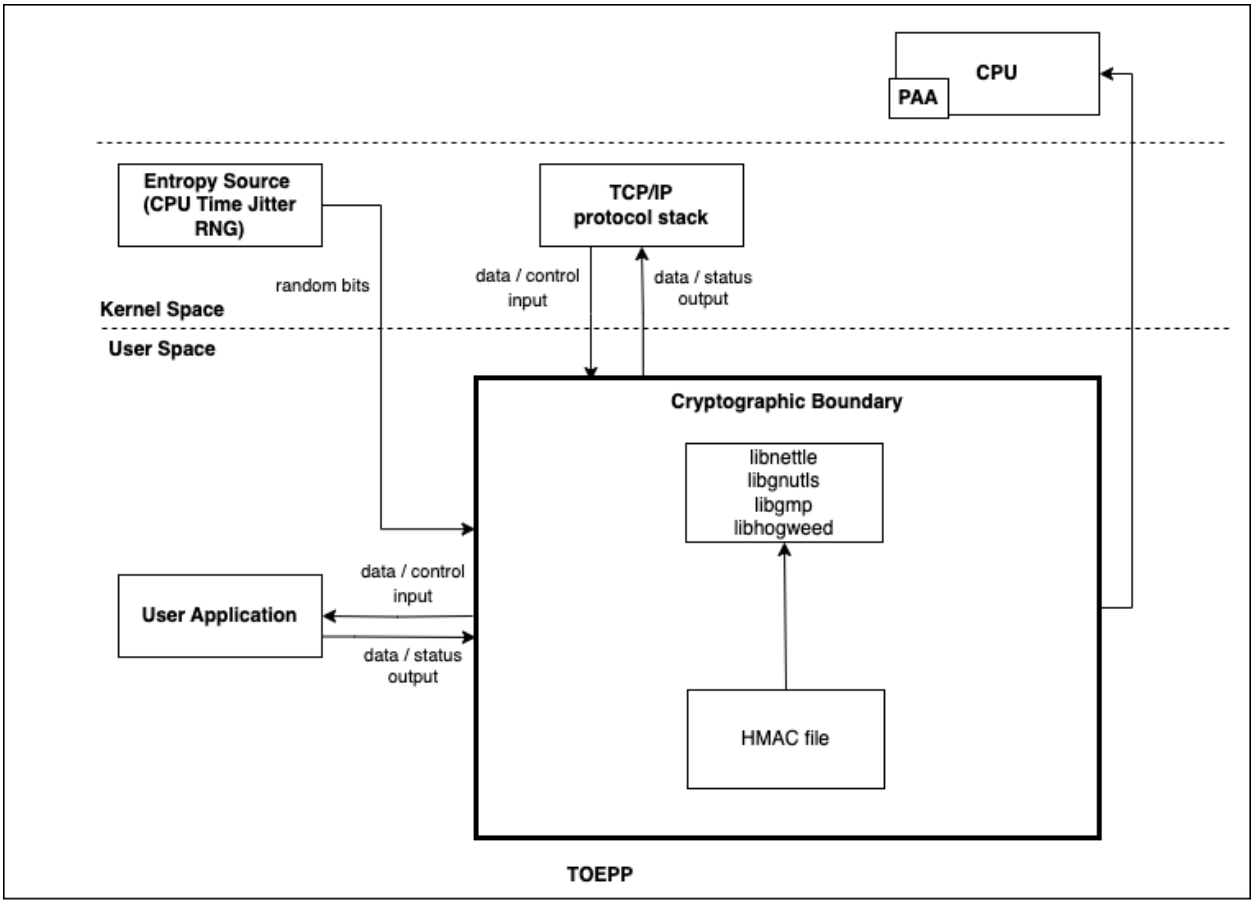

Figure 1: Block Diagram

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
libgnutls.so.30 (libgmp is statically linked to libgnutls), libnettle.so.8, libhogweed.so.6	rocky9.20241217	N/A	HMAC-SHA2-256
libgnutls.so.30, libnettle.so.6,	rocky8.20241217	N/A	HMAC-SHA2-256

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libhogweed.so.4, libgmp.so.10			

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	rocky9.20241217
Rocky Linux 8	SuperMicro SuperServer 5039MS	Intel Kaby Lake Xeon E3-1270 v6	Yes	N/A	rocky8.20241217
Rocky Linux 9	SuperMicro SuperServer 5039MS	Intel Kaby Lake Xeon E3-1270 v6	No	N/A	rocky9.20241217
Rocky Linux 8	SuperMicro SuperServer 5039MS	Intel Kaby Lake Xeon E3-1270 v6	No	N/A	rocky8.20241217

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

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

N/A for this module.

2.3 Excluded Components

The module does not claim any excluded components.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved mode	Automatically entered whenever an approved service is requested	Approved	Equivalent to the indicator of the requested service (GNUTLS_FIPS140_OP_APPROVED)
Non-approved mode	Automatically entered whenever a non-approved service is requested	Non-Approved	Equivalent to the indicator of the requested service (GNUTLS_FIPS140_OP_NOT_APPROVED)

Table 4: Modes List and Description

When the module starts up successfully, after passing all the pre-operational self-test and the cryptographic algorithms self-tests (CASTs), the module is operating in the approved mode of operation by default.

Mode Change Instructions and Status:

If the module is in the approved mode, it can only be transitioned into the non-approved mode by calling one of the non-approved services listed in the Non-Approved Services table. The module will transition back to approved mode when approved service is called. Section 4 provides details on the service indicator implemented by the module. The service indicator identifies when an approved service is called.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A6464, A6465, A6466, A6467, A6472, A6476, A6477, A6478, A6479, A6484	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A6464, A6476	Key Length - 128, 256	SP 800-38C
AES-CFB8	A6469, A6470, A6475, A6481, A6482, A6487	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A6464, A6467, A6472, A6476, A6479, A6484	Direction - Generation Key Length - 128, 256	SP 800-38B
AES-ECB	A6520, A6533	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A6464, A6465, A6466, A6467, A6472, A6476, A6477, A6478, A6479, A6484	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 256	SP 800-38D
AES-GMAC	A6472, A6484	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 256	SP 800-38D
AES-XTS Testing Revision 2.0	A6473, A6485	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E
Counter DRBG	A6472, A6484	Prediction Resistance - No Mode - AES-256 Derivation Function Enabled - No	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-5)	A6472, A6484	Curve - P-256, P-384, P-521 Secret Generation Mode - testing candidates	FIPS 186-5
ECDSA KeyVer (FIPS186-5)	A6472, A6484	Curve - P-256, P-384, P-521	FIPS 186-5

Algorithm	CAVP Cert	Properties	Reference
ECDSA SigGen (FIPS186-5)	A6472, A6484	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)	A6472, A6484	Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-5
HMAC-SHA-1	A6467, A6472, A6479, A6484	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A6467, A6472, A6479, A6484	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A6467, A6472, A6479, A6484	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A6467, A6472, A6479, A6484	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A6467, A6472, A6479, A6484	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A6472, A6484	Domain Parameter Generation Methods - P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-FFC-SSC Sp800-56Ar3	A6472, A6484	Domain Parameter Generation Methods - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Scheme - dhEphem - KAS Role - initiator, responder	SP 800-56A Rev. 3
KDA HKDF Sp800-56Cr1	A6471, A6483	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-65336 Increment 8 HMAC Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	SP 800-56C Rev. 2
PBKDF	A6472, A6484	Iteration Count - Iteration Count: 1000-10000 Increment 1 Password Length - Password Length: 8-128 Increment 1	SP 800-132
RSA KeyGen (FIPS186-5)	A6472, A6484	Key Generation Mode - provable Hash Algorithm - SHA2-384 Modulo - 2048, 3072, 4096, 6144, 8192 Private Key Format - standard	FIPS 186-5

Algorithm	CAVP Cert	Properties	Reference
RSA SigGen (FIPS186-5)	A6472, A6484	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
RSA SigVer (FIPS186-5)	A6472, A6484	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
Safe Primes Key Generation	A6472, A6484	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3
SHA-1	A6467, A6472, A6479, A6484	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-224	A6467, A6472, A6479, A6484	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A6467, A6472, A6479, A6484	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-384	A6467, A6472, A6479, A6484	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A6467, A6472, A6479, A6484	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA3-224	A6468, A6474, A6480, A6486	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-256	A6468, A6474, A6480, A6486	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-384	A6468, A6474, A6480, A6486	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-512	A6468, A6474, A6480, A6486	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
TLS v1.2 KDF RFC7627 (CVL)	A6472, A6484	Hash Algorithm - SHA2-256, SHA2-384	SP 800-135 Rev. 1

Table 5: Approved Algorithms

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

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG (asymmetric)	Key Type:Asymmetric	N/A	SP 800-133r2 section 4 example 1 without the use of V (refer to additional comment 2 of IG D.H)

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
Camellia	Symmetric encryption; Symmetric decryption
ChaCha20	Symmetric encryption; Symmetric decryption
Chacha20 and Poly1305	Authenticated encryption; Authenticated decryption
CMAC with Triple-DES	Message authentication code (MAC)
DES	Symmetric encryption; Symmetric decryption
ECDSA (with curves other than P-256, P-384, P-512)	Key generation; Public key verification
ECDSA (with curves other than P-256, P-384, P-512 or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512)	Digital signature generation; Digital signature verification
EC Diffie-Hellman (with curves other than P-256, P-384, P-512)	Key agreement; Shared secret computation
GMAC	Message authentication code (MAC)
GOST	Symmetric encryption; Symmetric decryption; Message digest
HMAC (with keys smaller than 112-bits)	Message authentication code (MAC)
HMAC (with GOST)	Message authentication code (MAC)
MD2, MD5	Message digest; Message authentication code (MAC)
PBKDF (with non-approved message digest algorithms or using input parameters not meeting requirements stated in section 2.7 of the security policy)	Key derivation
RC2, RC4	Symmetric encryption; Symmetric decryption
RMD160	Message digest; Message authentication code (MAC)

Name	Use and Function
RSA (with keys smaller than 2048 bits and/or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512)	Digital signature generation
RSA (with any key sizes)	Key encapsulation; Key un-encapsulation
Salsa20	Symmetric encryption; Symmetric decryption
SRP	Key agreement; Shared secret computation
STREEBOG	Message digest; Message authentication code (MAC)
Triple-DES	Symmetric encryption; Symmetric decryption
UMAC	Message authentication code (MAC)
Yarrow	Random number generation
AES-GCM (when not used in the context of the TLS protocol)	Authenticated encryption; Authenticated decryption
DSA	Key generation; Domain parameter generation; Digital signature generation; Digital signature verification
Diffie-Hellman (with domain parameters other than safe primes)	Key agreement; Shared secret computation

Table 7: Non-Approved, Not Allowed Algorithms

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

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Symmetric Encryption with AES	BC-UnAuth	Encryption using AES		AES-CBC: (A6464, A6465, A6466, A6467, A6472, A6476, A6477, A6478, A6479, A6484) AES-CFB8: (A6469, A6470, A6475, A6481, A6482, A6487) AES-XTS Testing Revision 2.0: (A6473, A6485)
Symmetric Decryption with AES	BC-UnAuth	Decryption using AES		AES-CBC: (A6464, A6465, A6466, A6467, A6472,

Name	Type	Description	Properties	Algorithms
				A6476, A6477, A6478, A6479, A6484) AES-CFB8: (A6469, A6470, A6475, A6481, A6482, A6487) AES-XTS Testing Revision 2.0: (A6473, A6485)
Authenticated Encryption with AES	BC-Auth	Authenticated encryption using AES		AES-CCM: (A6464, A6476)
Authenticated Decryption with AES	BC-Auth	Authenticated decryption using AES		AES-CCM: (A6464, A6476)
Authenticated Encryption (in the context of the TLS 1.2/1.3 protocol) with AES-GCM	BC-Auth	Authenticated encryption using AES-GCM (as part of TLS protocol)		AES-GCM: (A6464, A6465, A6466, A6467, A6472, A6476, A6477, A6478, A6479, A6484)
Authenticated Decryption (in the context of the TLS 1.2/1.3 protocol) with AES-GCM	BC-Auth	Authenticated decryption using AES-GCM (as part of TLS protocol)		AES-GCM: (A6464, A6465, A6466, A6467, A6472, A6476, A6477, A6478, A6479, A6484)
Message Authentication Code (MAC) with HMAC	MAC	Message authentication code using HMAC		HMAC-SHA-1: (A6467, A6472, A6479, A6484) HMAC-SHA2-224: (A6467, A6472, A6479, A6484) HMAC-SHA2-256: (A6467, A6472, A6479, A6484) HMAC-SHA2-384: (A6467, A6472, A6479, A6484) HMAC-SHA2-512: (A6467, A6472, A6479, A6484)

Name	Type	Description	Properties	Algorithms
Message Authentication Code (MAC) with AES	MAC	Message authentication code using AES		AES-CMAC: (A6464, A6467, A6472, A6476, A6479, A6484) AES-GMAC: (A6472, A6484)
Message Digest with SHA	SHA	Message digest using SHA		SHA-1: (A6467, A6472, A6479, A6484) SHA2-224: (A6467, A6472, A6479, A6484) SHA2-256: (A6467, A6472, A6479, A6484) SHA2-384: (A6467, A6472, A6479, A6484) SHA2-512: (A6467, A6472, A6479, A6484) SHA3-224: (A6468, A6474, A6480, A6486) SHA3-256: (A6468, A6474, A6480, A6486) SHA3-384: (A6468, A6474, A6480, A6486) SHA3-512: (A6468, A6474, A6480, A6486)
Random Number Generation with CTR_DRBG	DRBG	Random number generation using CTR_DRBG		Counter DRBG: (A6472, A6484) AES-ECB: (A6520, A6533)
Digital Signature Generation with RSA	DigSig-SigGen	Digital signature generation using RSA		RSA SigGen (FIPS186-5): (A6472, A6484)
Digital Signature Verification with RSA	DigSig-SigVer	Digital signature verification using RSA		RSA SigVer (FIPS186-5): (A6472, A6484)

Name	Type	Description	Properties	Algorithms
Digital Signature Generation with ECDSA	DigSig-SigGen	Digital signature generation using ECDSA		ECDSA SigGen (FIPS186-5): (A6472, A6484)
Digital Signature Verification with ECDSA	DigSig-SigVer	Digital signature verification using ECDSA		ECDSA SigVer (FIPS186-5): (A6472, A6484)
Key Pair Generation with RSA	AsymKeyPair-KeyGen CKG	Key Generation using RSA		RSA KeyGen (FIPS186-5): (A6472, A6484) CKG (asymmetric): ()
Key Pair Generation with ECDSA	AsymKeyPair-KeyGen CKG	Generate ECDSA key pairs		ECDSA KeyGen (FIPS186-5): (A6472, A6484) CKG (asymmetric): ()
Key Pair Generation with Safe Primes	AsymKeyPair-KeyGen CKG	Key Pair Generation with Safe Primes	Compliance:SP800-56Arev3	Safe Primes Key Generation: (A6472, A6484) CKG (asymmetric): ()
Public Key Verification with ECDSA	AsymKeyPair-KeyVer	Public key verification using ECDSA"		ECDSA KeyVer (FIPS186-5): (A6472, A6484)
Shared Secret Computation with EC Diffie-Hellman	KAS-SSC	Shared secret computation per SP800-56ARev3	Compliance:IG D.F scenario 2 path (1)	KAS-ECC-SSC Sp800-56Ar3: (A6472, A6484)
Shared Secret Computation with Diffie-Hellman	KAS-SSC	Shared secret computation per SP800-56ARev3	Compliance:IG D.F scenario 2 path (1)	KAS-FFC-SSC Sp800-56Ar3: (A6472, A6484)
Key Derivation with PBKDF	PBKDF	Key derivation using PBKDF		PBKDF: (A6472, A6484)
Key Derivation with TLS 1.2 KDF	KAS-135KDF	Key derivation using TLS KDF (CVL) / TLS v1.2 KDF RFC7627 (CVL)		TLS v1.2 KDF RFC7627: (A6472, A6484)
Key Derivation (as part of TLSv1.3) with KDA HKDF	KAS-56CKDF	Key derivation using KDA HKDF		KDA HKDF Sp800-56Cr1: (A6471, A6483)
TLS Handshake	KAS-Full	Key Agreement	IG:IG D.F scenario 2 path (2)	KAS-ECC-SSC Sp800-56Ar3:

Name	Type	Description	Properties	Algorithms
			Key derivation:IG 2.4.B SP 800-135rev1 CVL and KDA (separately tested) Key confirmation:No Caveat:Key establishment methodology provides between 112 and 256 bits of security strength	(A6472, A6484) KAS-FFC-SSC Sp800-56Ar3: (A6472, A6484) KDA HKDF Sp800-56Cr1: (A6471, A6483) TLS v1.2 KDF RFC7627: (A6472, A6484)

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

Hash Algorithms

In compliance with IG C.B, every approved hash algorithm implementation was CAVP tested and validated on all the module's operational environments. Section 2.5 of this security policy contains a table of the CAVP certificates of the approved hash functions.

For the higher-level algorithms that use the approved hash functions - Counter DRBG, ECDSA SigGen, ECDSA SigVer, HMAC, KDA HKDF Sp800-56Cr1, TLS v1.2 KDF RFC7627 (CVL), PBKDF2, RSA SigGen, RSA SigVer – every implemented combination for which CAVP testing exists was CAVP tested and validated on all the module's operational environments. Section 2.5 of this security policy contains a table of the CAVP certificates of these higher-level algorithms.

SHA-1 Use

SHA-1 is only approved when used in approved mode for message digest, message authentication and KDF (PBKDF). The use of SHA-1 for digital signature generation (e.g., ECDSA, RSA) or verification is non-approved.

SHA-3

The module provides SHA-3 hash functions compliant with IG C.C. Every implementation of each SHA-3 function was tested and validated on all the module's operating environments. SHAKE functions are not implemented. SHA-3 hash functions are not used as part of a higher-level algorithm.

RSA Key Generation

In compliance with IG C.E, the module generates RSA signature keys using an approved method of FIPS 186-5: generation of random primes that are provably prime. The CAVP certificate in table 2.5 indicates that the RSA key generating algorithm has been tested and validated for conformance to the methods in FIPS 186-5.

RSA Signature Generation and Signature Verification

In compliance with IG C.F, the module supports RSA modulus lengths greater than or equal to 2048 bits for both signature generation and signature verification. The RSA signature generation and signature verification

implementations have been tested for all module lengths available in CAVP testing: 2048, 3072, and 4096 bits. The number of Miller-Rabin tests is consistent with the bit sizes of p and q from Table B.1 of FIPS 186-5.

AES-GCM

The module implements AES GCM for being used in the TLS v1.2 and v1.3 protocols. AES GCM IV generation is compliant with FIPS 140-3 IG C.H for both protocols as follows:

- For TLS v1.2, IV generation is compliant with scenario 1.a of IG C.H and RFC5288. The module supports acceptable AES-GCM cipher suites from section 3.3.1 of SP 800-52 Rev. 2.
- For TLS v1.3, IV generation is compliant with scenario 5 of IG C.H and RFC8446. The module supports acceptable AES-GCM cipher suites from section 3.3.1 of SP 800-52 Rev. 2.
- The IV generated in both scenarios is only used within the context of the TLS protocol implementation. The nonce explicit part of the IV does not exhaust the maximum number of possible values for a given session key. The design of the TLS protocol in this module implicitly ensures that the nonce explicit, or counter portion of the IV will not exhaust all its possible values.

In case the module's power is lost and then restored, the key used for the AES GCM encryption or decryption shall be redistributed.

AES-XTS

The AES algorithm in XTS mode can only be used for the cryptographic protection of data on storage devices, as specified in SP 800-38E. The length of a single data unit encrypted with the AES-XTS shall not exceed 2^{20} AES blocks, that is 16MB of data.

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. As the module does not generate symmetric keys, the check is performed when keys are input to the service APIs.

The two XTS keys shall be generated and/or established independently according to the rules for component symmetric keys from NIST SP 800-133Rev2, Sec. 6.3.

Key Agreement Methods

To comply with the assurances listed in section 5.6.2 of SP 800-56ARev3, the operator must use the module in the context of the TLS protocol and the following steps shall be performed:

1. The entity using the module, must use the module's "Key pair generation" service for generating DH/ECDH ephemeral keys. This meets the assurances required by key pair owner defined in the section 5.6.2.1 of SP 800-56ARev3.
2. As part of the module's shared secret computation (SSC) service, the module internally performs the public key validation on the peer's public key passed in as input to the SSC function. This meets the public key validity assurance required by the sections 5.6.2.2.1/5.6.2.2.2 of SP 800-56ARev3.

The module does not support static keys therefore the "assurance of peer's possession of private key" is not applicable.

Cryptographic Key Generation

In compliance with IG D.H, the module generates seeds for asymmetric keys using the method described in section 4 example 1 of SP 800-133 Rev. 2 without the use of V (direct DRBG output as described in additional comment 2 of IG D.H).

Please refer to section 2.9 for more information on the key generation methods employed by the module.

DRBGs

In compliance with IG D.L, the entropy input, DRBG seed, DRBG internal state (values of V and Key) are considered CSPs.

The DRBG internal state is contained within the DRBG mechanism boundary and is not accessible by other mechanisms.

PBKDF2

The module provides password-based key derivation (PBKDF), compliant with SP 800-132 and IG D.N. 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, the following requirements shall be met.

- Derived keys shall only be used in storage applications. The Master Key (MK) shall not be used for other purposes. The length of the MK or DPK shall be 112 bits or more (this is verified by the module to determine the service is approved).
- A portion of the salt, with a length of at least 128 bits (this is verified by the module to determine the service is approved), shall be generated randomly using the SP 800-90A Rev. 1 DRBG.
- The iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users. The minimum value shall be 1000 (this is verified by the module to determine the service is approved).
- Passwords or passphrases, used as an input for the PBKDF, shall not be used as cryptographic keys.
- The length of the password or passphrase shall be of at least 20 characters (this is verified by the module to determine the service is approved), and shall consist of lower-case, upper-case, and numeric characters. The probability of guessing the value is estimated to be $1/62^{20}$, which is less than 2^{-112}. If the password consists of only digits (worst case), the probability of guessing the value is estimated to be 10^{-20} which is less than 2^{-112}.

The requirements of input parameters (derived key length, salt length, iteration count and password length) are verified by the module when providing the service indicator.

The calling application shall also observe the rest of the requirements and recommendations specified in SP 800-132.

TLS v1.2 KDF

In compliance with IG D.Q, the module supports the TLS 1.2 KDF with the extended master secret: TLS v1.2 KDF RFC7627 (CVL).

Authenticated Encryption / Decryption

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
E210	Ctrl IQ, Inc.

Table 9: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Rocky Linux Userspace CPU Time Jitter RNG Entropy Source	Non-Physical	Rocky Linux 8 on Intel Kaby Lake Xeon E3-1270 v6; Rocky Linux 9 on Intel Kaby Lake Xeon E3-1270 v6	256 bits	Full entropy	SHA3-256 cert. A5807, A5837; SHA2-512-HMAC-DRBG cert. A5807, A5837

Table 10: Entropy Sources

The module implements CTR_DRBG with AES-256, according to SP 800-90A Rev. 1, without a derivation function and without prediction resistance. The module uses an SP 800-90B-compliant entropy source as specified above. This entropy source is located within the physical perimeter, but outside of the cryptographic boundary of the module.

The module obtains 384 bits from the entropy source to seed the DRBG, and this is sufficient to provide a DRBG with 256 bits of security strength. The largest key strength generated by the module is 256 bits.

2.9 Key Generation

In accordance with FIPS 140-3 IG D.H, the cryptographic module performs Cryptographic Key Generation (CKG) and key derivation methods as listed in Section 2.6.

When random values are required, they are obtained from the SP 800-90Ar1 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 agreement methods as listed in Section 2.6.

2.11 Industry Protocols

The TLS protocol implementation provides both server and client sides. To operate in the approved mode, digital certificates used for server and client authentication shall comply with the restrictions of key size and message digest algorithms imposed by SP 800-131A Rev. 2.

No parts of the TLS protocol, other than the approved cryptographic algorithms and the KDFs, have been tested by the CAVP and CMVP.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	API input parameters, kernel I/O network or files on filesystem, TLS protocol input messages.
N/A	Data Output	API output parameters, kernel I/O network or files on filesystem, TLS protocol output messages.
N/A	Control Input	API function calls.
N/A	Status Output	API return codes, error message.

Table 11: Ports and Interfaces

All data output via data output interface is inhibited when the module is performing pre-operational test or zeroization or when the module enters error state.

The module does not implement a control output interface.

4 Roles, Services, and Authentication

4.1 Authentication Methods

N/A for this module.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 12: Roles

No support is provided for multiple concurrent operators.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Symmetric Encryption	Perform AES encryption	GNUTLS_FIPS140_OP_APPROVED	AES key, Plaintext	Ciphertext	Symmetric Encryption with AES	Crypto Officer - AES key: W,E
Symmetric Decryption	Perform AES decryption	GNUTLS_FIPS140_OP_APPROVED	AES key, Ciphertext	Plaintext	Symmetric Decryption with AES	Crypto Officer - AES key: W,E
Authenticated Encryption	Encrypt a plaintext	GNUTLS_FIPS140_OP_APPROVED	AES key, Plaintext, IV	Ciphertext, MAC tag	Authenticated Encryption with AES	Crypto Officer - AES key: W,E
Authenticated Decryption	Decrypt a ciphertext	GNUTLS_FIPS140_OP_APPROVED	AES key, Ciphertext, IV, MAC tag	Plaintext or fail	Authenticated Decryption with AES	Crypto Officer - AES key: W,E
Message Authentication Code (MAC) with HMAC	Compute HMAC	GNUTLS_FIPS140_OP_APPROVED	HMAC key, message	Message authentication code	Message Authentication Code (MAC) with HMAC	Crypto Officer - HMAC key: W,E
Message Authentication Code	Compute AES-base CMAC	GNUTLS_FIPS140_OP_APPROVED	AES key, message	Message authentication code	Message Authentication Code	Crypto Officer - AES key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
(MAC) with AES	or GMAC				(MAC) with AES	
Message Digest	Generating message digest	GNUTLS_FIPS140_OP_APPROVED	Message	Message digest	Message Digest with SHA	Crypto Officer
Random Number Generation	Generating random bitstrings	GNUTLS_FIPS140_OP_APPROVED	Output length	Random bytes	Random Number Generation with CTR_DRBG	Crypto Officer - Entropy Input: W,E,Z - DRBG seed: G,E - DRBG internal state (V value, key): G,W,E
Key Pair Generation with RSA	Generate RSA key pairs	GNUTLS_FIPS140_OP_APPROVED	Key size	Module-generated RSA private key, Module-generated RSA public key	Random Number Generation with CTR_DRBG Key Pair Generation with RSA	Crypto Officer - Module-generated RSA private key: G,R - Module-generated RSA public key: G,R - Intermediate key generation value: G,E,Z
Key Pair Generation with ECDSA	Generate ECDSA key pairs	GNUTLS_FIPS140_OP_APPROVED	Key size, enabled-curve	Module-generated ECDSA private key, Module-	Random Number Generation with CTR_DRBG	Crypto Officer - Module-generated ECDSA private

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
				generated ECDSA public key; Module-generated EC Diffie-Hellman private key, Module-generated EC Diffie-Hellman public key	Key Pair Generation with ECDSA	key: G,R - Module-generated ECDSA public key: G,R - Module-generated EC Diffie-Hellman private key: G,R - Module-generated EC Diffie-Hellman public key: G,R - Intermediate key generation value: G,E,Z
Key Pair Generation with Safe Primes	Generate DH key pairs	GNUTLS_FIPS140_OP_APPROVED	Key size	Module-generated Diffie-Hellman private key, Module-generated Diffie-Hellman public key	Key Pair Generation with Safe Primes	Crypto Officer - Module-generated Diffie-Hellman private key: G,R - Module-generated Diffie-Hellman public key: G,R - Intermediate key generation value: G,E,Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Public Key Verification with ECDSA	Verify ECDSA public key	GNUTLS_FIPS140_OP_APPROVED	ECDSA public key	Return codes/log messages	Public Key Verification with ECDSA	Crypto Officer - ECDSA public key: W,E
Digital Signature Generation with RSA	Generate RSA signature	GNUTLS_FIPS140_OP_APPROVED	Message, hash algorithm, RSA private key	Digital signature	Digital Signature Generation with RSA	Crypto Officer - RSA private key: W,E
Digital Signature Generation with ECDSA	Generate ECDSA signature	GNUTLS_FIPS140_OP_APPROVED	Message, hash algorithm, ECDSA private key	Digital signature	Digital Signature Generation with ECDSA	Crypto Officer - ECDSA private key: W,E
Digital Signature Verification with RSA	Verify RSA signature	GNUTLS_FIPS140_OP_APPROVED	Message, signature, hash algorithm, RSA public key	Verification result	Digital Signature Verification with RSA	Crypto Officer - RSA public key: W,E
Digital Signature Verification with ECDSA	Verify ECDSA signature	GNUTLS_FIPS140_OP_APPROVED	Message, signature, hash algorithm, ECDSA public key	Verification result	Digital Signature Verification with ECDSA	Crypto Officer - ECDSA public key: W,E
Shared Secret Computation with Diffie-Hellman	Compute a shared secret	GNUTLS_FIPS140_OP_APPROVED	Diffie-Hellman private key, Diffie-Hellman public key from peer	Diffie-Hellman shared secret	Shared Secret Computation with Diffie-Hellman	Crypto Officer - Diffie-Hellman shared secret: G,R - Diffie-Hellman

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						private key: W,E - Diffie-Hellman public key: W,E
Shared Secret Computation with EC Diffie-Hellman	Compute a shared secret	GNUTLS_FIPS140_OP_APPROVED	EC Diffie-Hellman private key, EC Diffie-Hellman public key from peer	EC Diffie-Hellman shared secret	Shared Secret Computation with Diffie-Hellman	Crypto Officer - EC Diffie-Hellman shared secret: G,R - EC Diffie-Hellman private key: W,E - EC Diffie-Hellman public key: W,E
TLS Key Derivation	Perform key derivation using TLS KDF	GNUTLS_FIPS140_OP_APPROVED	TLS pre-master secret	TLS Derived Secret	Key Derivation with TLS 1.2 KDF	Crypto Officer - TLS pre-master secret: W,E - TLS master secret: G,E,Z - TLS Derived Secret: G,R
Key Derivation with PBKDF	Perform password-based key	GNUTLS_FIPS140_OP_APPROVED	PBKDF password or passphrase	PBKDF Derived key	Key Derivation with PBKDF	Crypto Officer - PBKDF password or

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	derivation					passphrase: W,E - PBKDF Derived key: G,R
HKDF Key Derivation (derivation of HKDF Derived key)	Perform key derivation using HKDF	GNUTLS_FIPS140_OP_APPROVED	Diffie-Hellman shared secret or EC Diffie-Hellman shared secret	HKDF Derived key	Key Derivation (as part of TLSv1.3) with KDA HKDF	Crypto Officer - Diffie-Hellman shared secret: W,E - EC Diffie-Hellman shared secret: W,E - HKDF Derived key: G,R
Transport Layer Security (TLS) Network Protocol	Provide supported cipher suites (listed in Appendix A) in approved mode	GNUTLS_FIPS140_OP_APPROVED	Cipher-suites listed in Appendix A, Digital Certificate, Public and Private Keys, Application Data	Return codes and/or log messages, Application data	Symmetric Encryption with AES Symmetric Decryption with AES Authenticated Encryption with AES Authenticated Decryption with AES Authenticated Encryption (in the context of the TLS 1.2/1.3 protocol)	Crypto Officer - RSA public key: W,E - RSA private key: W,E - ECDSA public key: W,E - ECDSA private key: W,E - AES key: W,E - GCM IV: W,E - TLS pre-master secret: E,G

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					with AES-GCM Authenticated Decryption (in the context of the TLS 1.2/1.3 protocol) with AES-GCM Message Digest with SHA Message Authentication Code (MAC) with HMAC Message Authentication Code (MAC) with AES Digital Signature Generation with RSA Digital Signature Verification with RSA Digital Signature Generation with ECDSA Digital Signature Verification with ECDSA	- TLS master secret: E,G,Z - Module-generated Diffie-Hellman private key: E,G - Module-generated Diffie-Hellman public key: W,E,G,R - Module-generated EC Diffie-Hellman private key: E,G - Module-generated EC Diffie-Hellman public key: W,E,G,R - TLS Derived Secret: E,G - HKDF Derived key: E,G

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					Public Key Verification with ECDSA TLS Handshake Key Pair Generation with RSA Key Pair Generation with ECDSA Key Pair Generation with Safe Primes	
Self-tests	Perform self-tests	N/A	N/A	Result of self-test (pass/fail)	Symmetric Encryption with AES Symmetric Decryption with AES Authenticated Encryption with AES Authenticated Decryption with AES Authenticated Encryption (in the context of the TLS 1.2/1.3 protocol) with AES-GCM Authenticated Decryption	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					(in the context of the TLS 1.2/1.3 protocol) with AES-GCM Message Digest with SHA Random Number Generation with CTR_DRBG Message Authentication Code (MAC) with HMAC Message Authentication Code (MAC) with AES Digital Signature Generation with RSA Digital Signature Verification with RSA Digital Signature Generation with ECDSA Digital Signature Verification with	

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					ECDSA Public Key Verification with ECDSA Shared Secret Computation with EC Diffie-Hellman Shared Secret Computation with Diffie-Hellman Key Derivation with PBKDF Key Derivation with TLS 1.2 KDF Key Derivation (as part of TLSv1.3) with KDA HKDF TLS Handshake Key Pair Generation with RSA Key Pair Generation with ECDSA Key Pair Generation with Safe Primes	

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show module name and version	Show module name and version	N/A	N/A	Name and version information	None	Crypto Officer
Show Status	Show module status	N/A	N/A	Return codes and/or log messages	None	Crypto Officer
Zeroization	Zeroize SSPs	N/A	Context containing SSPs	N/A	None	Crypto Officer - AES key: Z - GCM IV: Z - HMAC key: Z - Module-generated RSA private key: Z - Module-generated RSA public key: Z - RSA private key: Z - RSA public key: Z - PBKDF password or passphrase: Z - PBKDF Derived key: Z - Intermedi

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						ate key generation value: Z - Module-generated ECDSA private key: Z - Module-generated ECDSA public key: Z - ECDSA private key: Z - ECDSA public key: Z - Module-generated EC Diffie-Hellman private key: Z - Module-generated EC Diffie-Hellman public key: Z - EC Diffie-Hellman private key: Z - EC Diffie-Hellman public key: Z - Module-generated Diffie-

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Hellman private key: Z - Module-generated Diffie-Hellman public key: Z - Diffie-Hellman private key: Z - Diffie-Hellman public key: Z - Diffie-Hellman shared secret: Z - EC Diffie-Hellman shared secret: Z - Entropy Input: Z - DRBG seed: Z - DRBG internal state (V value, key): Z - TLS pre-master secret: Z - HKDF Derived key: Z - TLS master secret: Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- TLS Derived Secret: Z

Table 13: Approved Services

The table above lists the approved services. For each service, the table lists the associated cryptographic algorithm(s), the role to perform the service, the cryptographic keys or CSPs involved, and their access type(s). The following convention is used to specify access rights to a CSP:

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

The details of the approved cryptographic algorithms including the CAVP certificate numbers can be found in the Approved Algorithm table.

The "Indicator" column shows the service indicator API functions that must be used to verify the service indicator for each of the services. The function gnutls_fips140_get_operation_state() indicates GNUTLS_FIPS140_OP_APPROVED or GNUTLS_FIPS140_OP_NOT_APPROVED depending on whether the API invoked corresponds to an approved or non-approved algorithm.

4.4 Non-Approved Services

Name	Description	Algorithms	Role
Symmetric encryption; Symmetric decryption	Symmetric encryption; Symmetric decryption	Camellia ChaCha20 DES GOST RC2, RC4 Salsa20 Triple-DES	CO
Authenticated encryption; Authenticated decryption	Authenticated encryption; Authenticated decryption	Chacha20 and Poly1305 AES-GCM (when not used in the context of the TLS protocol)	CO
Message authentication code	Message authentication code	CMAC with Triple-DES GMAC HMAC (with keys smaller than 112-bits)	CO

Name	Description	Algorithms	Role
		HMAC (with GOST) UMAC	
Message digest	Message digest	GOST MD2, MD5 RMD160 STREEBOG	CO
Key derivation	Key derivation	PBKDF (with non-approved message digest algorithms or using input parameters not meeting requirements stated in section 2.7 of the security policy)	CO
Domain parameter generation	Domain parameter generation	DSA	CO
Asymmetric key generation	Asymmetric key generation	ECDSA (with curves other than P-256, P-384, P-512) DSA	CO
Public key verification	Public key verification	ECDSA (with curves other than P-256, P-384, P-512)	CO
Digital signature verification	Digital signature verification	ECDSA (with curves other than P-256, P-384, P-512 or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512) RSA (with keys smaller than 2048 bits and/or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512) DSA	CO
Digital signature generation	Digital signature generation	ECDSA (with curves other than P-256, P-384, P-512 or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512) RSA (with keys smaller than 2048 bits and/or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512) DSA	CO
Key agreement; Shared secret computation	Key agreement; Shared secret computation	EC Diffie-Hellman (with curves other than P-256, P-384, P-512) SRP Diffie-Hellman (with domain parameters other than safe primes)	CO
Key encapsulation; Key un-encapsulation	Key encapsulation; Key un-encapsulation	RSA (with any key sizes)	CO
Random number generation	Yarrow	Yarrow	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 have the capability of loading software or firmware from an external source.

5 Software/Firmware Security

5.1 Integrity Techniques

Each software component of the module has an associated HMAC-SHA2-256 integrity check value. The integrity of the module is verified by comparing the HMAC-SHA2-256 value calculated at run time for each software component of the module listed in section 2, with the HMAC value stored in the .hmac file that was computed at build time for each software component of the module listed in section 2. The .hmac file has HMAC value for libgnutls, libnettle and libhogweed listed in section 2. For the rocky8.20241217 version of the module, the .hmac file also includes the HMAC value for the libgmp. While, for version rocky9.20241217 of the module, the libgmp is statically linked to the libgnutls, thus the HMAC value for is unique for both libgnutls and libgmp. The HMAC key used for integrity check is stored within the module. If the integrity test fails the module enters the error state.

Integrity tests are performed as part of the Pre-Operational Self-Tests.

5.2 Initiate on Demand

The pre-operational integrity self-test can be initiated on demand by calling the Self-Test service (via the gnutls_fips140_run_self_tests() function) or by powering-off and reloading the module. During the execution of the on-demand integrity self-test, services are not available, and no data output is possible.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

The module operates in a modifiable operational environment per FIPS 140-3 level 1 specification: the module executes on a general-purpose operating system, which allows modification, loading, and execution of software that is not part of the validated module.

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.

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

7 Physical Security

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

8 Non-Invasive Security

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

9 Sensitive Security Parameters Management

9.1 Storage Areas

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

Table 15: Storage Areas

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

9.2 SSP Input-Output Methods

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

Table 16: SSP Input-Output Methods

The module does not support manual SSP entry or intermediate SSP generation output.

The SSPs are provided to the module via API input parameters in plaintext form and output via API output parameters in plaintext form within the physical perimeter of the operational environment. This is allowed by FIPS 140-3 IG 9.5.A, according to the "CM Software to/from App via TOEPP Path" entry on the Key Establishment Table.

The module does not support entry or output of cryptographically protected SSPs.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Zeroize Context	The memory occupied by SSPs is allocated by regular memory allocation	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable. The completion of the	By calling the appropriate zeroization functions: AES key: gnutls_cipher_deinit() AES key: gnutls_aead_cipher_deinit() HMAC key: gnutls_hmac_deinit() RSA Public Key, RSA Private Key: gnutls_privkey_deinit() gnutls_x509_privkey_deinit()

Zeroization Method	Description	Rationale	Operator Initiation
	operating system calls.	zeroization routine indicates that the zeroization procedure succeeded.	gnutls_rsa_params_deinit() ECDSA Public Key, ECDSA Private Key: gnutls_privkey_deinit() gnutls_x509_privkey_deinit() gnutls_rsa_params_deinit() Diffie-Hellman Public Key, Diffie-Hellman private key: gnutls_dh_params_deinit() TLS pre-master secret: gnutls_deinit() TLS master secret: gnutls_deinit() HKDF Derived key: gnutls_deinit() Diffie-Hellman Public Key, Diffie-Hellman private key: gnutls_pk_params_clear() EC Diffie-Hellman public key, EC Diffie-Hellman private key: gnutls_pk_params_clear() Diffie-Hellman Shared Secret: zeroize key() EC Diffie-Hellman Shared Secret: zeroize_key() All SSPs: gnutls_global_deinit()
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
Reset	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.	Unloading and reloading the module

Table 17: SSP Zeroization Methods

The memory occupied by SSPs is allocated by regular memory allocation operating system calls. The application that is acting as the CO is responsible for calling the appropriate zeroization functions provided in the module's API and listed in the above table. Calling gcry_free(), which will zeroize the SSPs and also invoke the corresponding API functions listed in the above table to zeroize SSPs. The zeroization functions overwrite the memory occupied by SSPs with “zeros” and deallocate the memory with the regular memory deallocation operating system call.

All data output is inhibited during zeroization.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES key	AES key used for encryption, decryption, and computing MAC tags	AES-XTS, AES-CCM, AES-GCM, AES-CMAC: 128, 256 bits; Other modes: 128, 192, 256 bits - AES-XTS, AES-CCM, AES-GCM, AES-CMAC: 128, 256 bits; Other modes: 128, 192, 256 bits	Symmetric key - CSP			Symmetric Encryption with AES Symmetric Decryption with AES Authenticated Encryption with AES Authenticated Decryption with AES Message Authentication Code (MAC) with AES
GCM IV	Initialization vector used by AES-GCM in the context of TLS protocol	96 bits - N/A	Initialization vector - PSP			Authenticated Encryption (in the context of the TLS 1.2/1.3 protocol) with AES-GCM Authenticated Decryption (in the context of the TLS 1.2/1.3 protocol) with AES-GCM

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
HMAC key	HMAC key used for computing MAC tags	112 to 524288 bits - 112 to 256 bits	Symmetric key - CSP			Message Authentication Code (MAC) with HMAC
Module-generated RSA private key	RSA private key generated through asymmetric key generation	2048, 3072, 4096, 6144, 7680, 8192, 15360-bits - 112, 128, 149, 178, 192, 201, and 256 bits	Private key - CSP	Key Pair Generation with RSA		
Module-generated RSA public key	RSA public key generated through asymmetric key generation	2048, 3072, 4096, 6144, 7680, 8192, 15360-bits - 112, 128, 149, 178, 192, 201, and 256 bits	Public key - PSP	Key Pair Generation with RSA		
RSA private key	RSA private key used for digital signature generation	2048-16384 bits - 112-256 bits	Private key - CSP			Digital Signature Generation with RSA
RSA public key	RSA public key used for digital signature verification	2048-16384 bits - 112-256 bits	Public key - PSP			Digital Signature Verification with RSA

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Module-generated ECDSA private key	ECDSA private key generated through the asymmetric key generation	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP	Key Pair Generation with ECDSA		
Module-generated ECDSA public key	ECDSA private key generated through the asymmetric key generation	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP	Key Pair Generation with ECDSA		
ECDSA private key	ECDSA private key used for digital signature generation	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP			Digital Signature Generation with ECDSA
ECDSA public key	ECDSA public key used for digital signature verification	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP			Public Key Verification with ECDSA Digital Signature Verification with ECDSA
Module-generated EC Diffie-Hellman public key	EC Diffie-Hellman public key generated during asymmetric key generation	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP	Key Pair Generation with ECDSA		TLS Handshake
Module-generated EC Diffie-Hellman private key	EC Diffie-Hellman private key generated during asymmetric key generation	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP	Key Pair Generation with ECDSA		TLS Handshake
EC Diffie-Hellman public key	Public key used for Shared Secret Computation	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP			Shared Secret Computation with EC Diffie-Hellman
EC Diffie-Hellman private key	Private key used for Shared Secret Computation	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP			Shared Secret Computation with EC Diffie-Hellman

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Module-generated Diffie-Hellman public key	Diffie-Hellman public key generated during Safe Primes Key Generation	2048, 3072, 4096, 6144, 8192 bits - 112, 128, 152, 176, 200 bits	Public key - PSP	Key Pair Generation with Safe Primes		TLS Handshake
Module-generated Diffie-Hellman private key	Diffie-Hellman private key generated during Safe Primes Key Generation	2048, 3072, 4096, 6144, 8192 bits - 112, 128, 152, 176, 200 bits	Private key - CSP	Key Pair Generation with Safe Primes		TLS Handshake
Diffie-Hellman public key	Public key used for Shared Secret Computation	2048, 3072, 4096, 6144, 8192 bits - 112, 128, 152, 176, 200 bits	Public key - PSP			Shared Secret Computation with Diffie-Hellman
Diffie-Hellman private key	Private key used for Shared Secret Computation	2048, 3072, 4096, 6144, 8192 bits - 112, 128, 152, 176, 200 bits	Private key - CSP			Shared Secret Computation with Diffie-Hellman
Diffie-Hellman shared secret	Shared secret generated by Diffie-Hellman	2048, 3072, 4096, 6144, 8192 bits - 112, 128, 152,	Shared Secret - CSP		Shared Secret Computation with Diffie-Hellman	

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		176, 200 bits				
EC Diffie-Hellman shared secret	Shared secret generated by EC Diffie-Hellman	P-256, P-384, P-521 - 128, 192, 256 bits	Shared Secret - CSP		Shared Secret Computation with EC Diffie-Hellman	
PBKDF password or passphrase	Password used to derive symmetric keys	20 characters minimum - N/A	Password - CSP			Key Derivation with PBKDF
PBKDF Derived key	Key derived from PBKDF password/passphrase during key derivation	128-256 bits - 128-256 bits	Derived key - CSP	Key Derivation with PBKDF		
Entropy Input	Entropy input used to seed the DRBG	256-384 bits - 256-384 bits	Entropy Input - CSP			Random Number Generation with CTR_DRBG
DRBG seed	DRBG seed derived from entropy input	256-384 bits - 256-384 bits	Seed - CSP	Random Number Generation with CTR_DRBG		Random Number Generation with CTR_DRBG
DRBG internal state (V value, key)	Used for Random number generation. This SSP is a CSP in compliance with IG D.L	V: 128 bits; Key: 256 bits - 256 bits	Internal state - CSP	Random Number Generation with CTR_DRBG		Random Number Generation with CTR_DRBG
TLS pre-master secret	Used for Transport Layer Security (TLS) network protocol	256-8192 bits - 112 to 256 bits	Shared secret - CSP		Shared Secret Computation with EC Diffie-Hellman Shared	TLS Handshake

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
					Secret Computation with Diffie-Hellman	
TLS master secret	TLS master secret used for deriving the TLS Derived Secret	384 bits - 112-256 bits	Intermediate secret value - CSP	Key Derivation with TLS 1.2 KDF		TLS Handshake
TLS Derived Secret	Used as encryption key or MAC key	112-256 bits - 112-256 bits	Derived secret - CSP	Key Derivation with TLS 1.2 KDF		TLS Handshake
Intermediate key generation value	Temporary value generated during key generation services	256 to 15360 bits - 112 to 256 bits	Intermediate key generation value - CSP	Key Pair Generation with RSA Key Pair Generation with ECDSA Key Pair Generation with Safe Primes		Key Pair Generation with RSA Key Pair Generation with ECDSA Key Pair Generation with Safe Primes
HKDF Derived key	HKDF (used as part of TLS 1.3 protocol) derived key	128-256 bits - 128-256 bits	Derived secret - CSP	Key Derivation (as part of TLSv1.3) with KDA HKDF		TLS Handshake

Table 18: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES key	API input parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	
GCM IV	API input parameters API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
HMAC key	API input parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	
Module-generated RSA private key	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Module-generated RSA public key:Paired With Intermediate key generation value:Generated From
Module-generated RSA public key	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Module-generated RSA private key:Paired With Intermediate key generation value:Generated From
RSA private key	API input parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	RSA public key:Paired With
RSA public key	API input parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	RSA private key:Paired With
Module-generated ECDSA private key	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Module-generated ECDSA public key:Paired With Intermediate key generation value:Generated From
Module-generated ECDSA public key	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Module-generated ECDSA private key:Paired With Intermediate key generation value:Generated From
ECDSA private key	API input parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	ECDSA public key:Paired With
ECDSA public key	API input parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	ECDSA private key:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Module-generated EC Diffie-Hellman public key	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Module-generated EC Diffie-Hellman private key:Paired With Intermediate key generation value:Generated From
Module-generated EC Diffie-Hellman private key	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Module-generated EC Diffie-Hellman public key:Paired With Intermediate key generation value:Generated From
EC Diffie-Hellman public key	API input parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	EC Diffie-Hellman private key:Paired With EC Diffie-Hellman shared secret:Derives
EC Diffie-Hellman private key	API input parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	EC Diffie-Hellman public key:Paired With EC Diffie-Hellman shared secret:Derives
Module-generated Diffie-Hellman public key	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Module-generated Diffie-Hellman private key:Paired With Intermediate key generation value:Generated From
Module-generated Diffie-Hellman private key	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Module-generated Diffie-Hellman public key:Paired With Intermediate key generation value:Generated From
Diffie-Hellman public key	API input parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Diffie-Hellman private key:Paired With Diffie-Hellman shared secret:Derives
Diffie-Hellman private key	API input parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Diffie-Hellman public key:Paired With Diffie-Hellman shared secret:Derives

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Diffie-Hellman shared secret	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Diffie-Hellman public key:Derived from Diffie-Hellman private key:Derived from
EC Diffie-Hellman shared secret	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	EC Diffie-Hellman public key:Derived from EC Diffie-Hellman private key:Derived from
PBKDF password or passphrase	API input parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	PBKDF derived key:Derived From
PBKDF Derived key	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	PBKDF password or passphrase:Derived From
Entropy Input		RAM:Plaintext	From generation until DRBG Seed is created	Zeroize Context Automatic	DRBG seed:Derives
DRBG seed		RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Automatic	Entropy input:Derived From DRBG internal state:Derives
DRBG internal state (V value, key)		RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	DRBG seed:Derived from
TLS pre-master secret	API input parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	TLS master secret:Derives Module-generated Diffie-Hellman private key:Established by Module-generated Diffie-Hellman public key:Established by Module-generated EC Diffie-Hellman private key:Established by Module-generated EC

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
					Diffie-Hellman public key:Established by
TLS master secret		RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	TLS pre-master secret:Derived From TLS Derived Secret:Derives
TLS Derived Secret	API output parameters	RAM:Plaintext	For the duration of the service	Zeroize Context Reset	TLS master secret:Derived From
Intermediate key generation value		RAM:Plaintext	Until explicitly zeroized by operator	Automatic Zeroize Context Reset	Module-generated Diffie-Hellman private key:Generation Of Module-generated Diffie-Hellman public key:Generation Of Module-generated EC Diffie-Hellman private key:Generation Of Module-generated EC Diffie-Hellman public key:Generation Of Module-generated RSA private key:Generation Of Module-generated RSA public key:Generation Of
HKDF Derived key	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Diffie-Hellman Shared secret:Derived From EC Diffie-Hellman Shared secret:Derived From

Table 19: SSP Table 2

The tables above summarize the Sensitive Security Parameters (SSPs) that are used by the cryptographic services implemented in the module.

9.5 Transitions

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

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A6472)	256-bit key	Integrity	SW/FW Integrity	Module becomes operational and services are available for use	MAC tag computation
HMAC-SHA2-256 (A6484)	256-bit key	Integrity	SW/FW Integrity	Module becomes operational and services are available for use	MAC tag computation

Table 20: Pre-Operational Self-Tests

The module performs pre-operational self-tests automatically when the module is becoming available for the consuming application. Pre-operational self-tests ensure that the module is not corrupted. While the module is executing the pre-operational self-tests, services are not available, input and output are inhibited. The module is not available for use by the calling application until the pre-operational self-tests are completed successfully.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC - Encrypt (A6464)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC - Encrypt (A6465)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC - Encrypt (A6466)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC - Encrypt (A6467)	256-bit key	KAT	CAST	Module becomes operational and	Encryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
AES-CBC - Encrypt (A6472)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC - Encrypt (A6476)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC - Encrypt (A6477)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC - Encrypt (A6478)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC - Encrypt (A6479)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC - Encrypt (A6484)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC - Decrypt (A6464)	256-bit key	KAT	CAST	Module becomes operational and	Decryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
AES-CBC - Decrypt (A6465)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CBC - Decrypt (A6466)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CBC - Decrypt (A6467)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CBC - Decrypt (A6472)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CBC - Decrypt (A6476)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CBC - Decrypt (A6477)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CBC - Decrypt (A6478)	256-bit key	KAT	CAST	Module becomes operational and	Decryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
AES-CBC - Decrypt (A6479)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CBC - Decrypt (A6484)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM - Encrypt (A6464)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Encrypt (A6465)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Encrypt (A6466)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Encrypt (A6467)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Encrypt (A6472)	256-bit key	KAT	CAST	Module becomes operational and	Encryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
AES-GCM - Encrypt (A6476)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Encrypt (A6477)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Encrypt (A6478)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Encrypt (A6479)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Encrypt (A6484)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Decrypt (A6464)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM - Decrypt (A6465)	256-bit key	KAT	CAST	Module becomes operational and	Decryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
AES-GCM - Decrypt (A6466)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM - Decrypt (A6467)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM - Decrypt (A6472)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM - Decrypt (A6476)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM - Decrypt (A6477)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM - Decrypt (A6478)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM - Decrypt (A6479)	256-bit key	KAT	CAST	Module becomes operational and	Decryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
AES-GCM - Decrypt (A6484)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CFB8 - Encrypt (A6469)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CFB8 - Encrypt (A6470)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CFB8 - Encrypt (A6475)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CFB8 - Encrypt (A6481)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CFB8 - Encrypt (A6482)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CFB8 - Encrypt (A6487)	256-bit key	KAT	CAST	Module becomes operational and	Encryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
AES-CFB8 - Decrypt (A6469)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CFB8 - Decrypt (A6470)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CFB8 - Decrypt (A6475)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CFB8 - Decrypt (A6481)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CFB8 - Decrypt (A6482)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CFB8 - Decrypt (A6487)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-XTS Testing Revision 2.0 -	256-bit key	KAT	CAST	Module becomes operational and	Encryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
Encrypt (A6473)				services are available for use		
AES-XTS Testing Revision 2.0 - Encrypt (A6485)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-XTS Testing Revision 2.0 - Decrypt (A6473)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-XTS Testing Revision 2.0 - Decrypt (A6485)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
SHA3-224 (A6468)	32-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-224 (A6474)	32-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-224 (A6480)	32-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-224 (A6486)	32-bit message	KAT	CAST	Module becomes operational and	Message digest	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
SHA3-256 (A6468)	32-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-256 (A6474)	32-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-256 (A6480)	32-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-256 (A6486)	32-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-384 (A6468)	64-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-384 (A6474)	64-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-384 (A6480)	64-bit message	KAT	CAST	Module becomes operational and	Message digest	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
SHA3-384 (A6486)	64-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-512 (A6468)	136-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-512 (A6474)	136-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-512 (A6480)	136-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-512 (A6486)	136-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
HMAC-SHA-1 (A6467)	SHA-1	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA-1 (A6472)	SHA-1	KAT	CAST	Module becomes operational and	Message authentication	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
HMAC-SHA-1 (A6479)	SHA-1	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA-1 (A6484)	SHA-1	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-224 (A6467)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-224 (A6472)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-224 (A6479)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-224 (A6484)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-256 (A6467)	160-bit key	KAT	CAST	Module becomes operational and	Message authentication	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
HMAC-SHA2-256 (A6472)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-256 (A6479)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-256 (A6484)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-384 (A6467)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-384 (A6472)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-384 (A6479)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-384 (A6484)	160-bit key	KAT	CAST	Module becomes operational and	Message authentication	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
HMAC-SHA2-512 (A6467)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-512 (A6472)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-512 (A6479)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-512 (A6484)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
AES-CMAC (A6464)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	MAC generation	Module initialization
AES-CMAC (A6467)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	MAC generation	Module initialization
AES-CMAC (A6472)	256-bit keys	KAT	CAST	Module becomes operational and	MAC generation	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
AES-CMAC (A6476)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	MAC generation	Module initialization
AES-CMAC (A6479)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	MAC generation	Module initialization
AES-CMAC (A6484)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	MAC generation	Module initialization
RSA SigGen (FIPS186-5) (A6472)	2048-bit key using SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Digital signature generation	Module initialization
RSA SigGen (FIPS186-5) (A6484)	2048-bit key using SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Digital signature generation	Module initialization
RSA SigVer (FIPS186-5) (A6472)	2048-bit key using SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Digital signature verification	Module initialization
RSA SigVer (FIPS186-5) (A6484)	2048-bit key using SHA2-256	KAT	CAST	Module becomes operational and	Digital signature verification	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
ECDSA SigGen (FIPS186-5) (A6472)	P-256 using SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Digital signature generation	Module initialization
ECDSA SigGen (FIPS186-5) (A6484)	P-256 using SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Digital signature generation	Module initialization
ECDSA SigVer (FIPS186-5) (A6472)	P-256 using SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Digital signature verification	Module initialization
ECDSA SigVer (FIPS186-5) (A6484)	P-256 using SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Digital signature verification	Module initialization
Counter DRBG (A6472)	256-bit keys without DF, without PR	KAT	CAST	Module becomes operational and services are available for use	Instantiate, Generate, and Reseed (compliant to SP 800-90A Rev. 1, Section 11.3)	Test runs at power-on before the integrity test
Counter DRBG (A6484)	256-bit keys without DF, without PR	KAT	CAST	Module becomes operational and services are available for use	Instantiate, Generate, and Reseed (compliant to SP 800-90A Rev. 1, Section 11.3)	Test runs at power-on before the integrity test
KAS-FFC-SSC Sp800-56Ar3 (A6472)	ffdhe3072	KAT	CAST	Module becomes operational and	Primitive "Z" Computation	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
KAS-FFC-SSC Sp800-56Ar3 (A6484)	ffdhe3072	KAT	CAST	Module becomes operational and services are available for use	Primitive "Z" Computation	Module initialization
KAS-ECC-SSC Sp800-56Ar3 (A6472)	P-256	KAT	CAST	Module becomes operational and services are available for use	Primitive "Z" Computation	Module initialization
KAS-ECC-SSC Sp800-56Ar3 (A6484)	P-256	KAT	CAST	Module becomes operational and services are available for use	Primitive "Z" Computation	Module initialization
TLS v1.2 KDF RFC7627 (A6472)	SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Industry-based TLS v1.2 KDF key derivation	Module initialization
TLS v1.2 KDF RFC7627 (A6484)	SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Industry-based TLS v1.2 KDF key derivation	Module initialization
PBKDF (A6472)	SHA-256 with 4096 iterations and 288-bit salt	KAT	CAST	Module becomes operational and services are available for use	Key Derivation with PBKDF	Module initialization
PBKDF (A6484)	SHA-256 with 4096 iterations	KAT	CAST	Module becomes operational and	Key Derivation with PBKDF	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
	and 288-bit salt			services are available for use		
KDA HKDF Sp800-56Cr1 (A6471)	SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Key Derivation (as part of TLSv1.3) with KDA HKDF	Module initialization
KDA HKDF Sp800-56Cr1 (A6483)	SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Key Derivation (as part of TLSv1.3) with KDA HKDF	Module initialization
RSA KeyGen (FIPS186-5) (A6472)	SHA2-256	PCT	PCT	Successful key pair generation	Signature generation and verification	Key Pair Generation
RSA KeyGen (FIPS186-5) (A6484)	SHA2-256	PCT	PCT	Successful key pair generation	Signature generation and verification	Key Pair Generation
ECDSA KeyGen (FIPS186-5) (A6472)	SHA-256 with the respective curve	PCT	PCT	Successful key pair generation	Signature generation and verification	Key Pair Generation
ECDSA KeyGen (FIPS186-5) (A6484)	SHA-256 with the respective curve	PCT	PCT	Successful key pair generation	Signature generation and verification	Key Pair Generation
Safe Primes Key Generation (A6472)	N/A	PCT	PCT	Successful key pair generation	PCT according to section 5.6.2.1.4 of [SP800-56Arev3]	Key Pair Generation
Safe Primes Key Generation (A6484)	N/A	PCT	PCT	Successful key pair generation	PCT according to section 5.6.2.1.4 of [SP800-56Arev3]	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 (A6472)	Integrity	SW/FW Integrity	On demand	Manual
HMAC-SHA2-256 (A6484)	Integrity	SW/FW Integrity	On demand	Manual

Table 22: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC - Encrypt (A6464)	KAT	CAST	On demand	Manually
AES-CBC - Encrypt (A6465)	KAT	CAST	On demand	Manually
AES-CBC - Encrypt (A6466)	KAT	CAST	On demand	Manually
AES-CBC - Encrypt (A6467)	KAT	CAST	On demand	Manually
AES-CBC - Encrypt (A6472)	KAT	CAST	On demand	Manually
AES-CBC - Encrypt (A6476)	KAT	CAST	On demand	Manually
AES-CBC - Encrypt (A6477)	KAT	CAST	On demand	Manually
AES-CBC - Encrypt (A6478)	KAT	CAST	On demand	Manually
AES-CBC - Encrypt (A6479)	KAT	CAST	On demand	Manually
AES-CBC - Encrypt (A6484)	KAT	CAST	On demand	Manually
AES-CBC - Decrypt (A6464)	KAT	CAST	On demand	Manually
AES-CBC - Decrypt (A6465)	KAT	CAST	On demand	Manually
AES-CBC - Decrypt (A6466)	KAT	CAST	On demand	Manually
AES-CBC - Decrypt (A6467)	KAT	CAST	On demand	Manually
AES-CBC - Decrypt (A6472)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC - Decrypt (A6476)	KAT	CAST	On demand	Manually
AES-CBC - Decrypt (A6477)	KAT	CAST	On demand	Manually
AES-CBC - Decrypt (A6478)	KAT	CAST	On demand	Manually
AES-CBC - Decrypt (A6479)	KAT	CAST	On demand	Manually
AES-CBC - Decrypt (A6484)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A6464)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A6465)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A6466)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A6467)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A6472)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A6476)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A6477)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A6478)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A6479)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A6484)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A6464)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A6465)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A6466)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A6467)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A6472)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-GCM - Decrypt (A6476)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A6477)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A6478)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A6479)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A6484)	KAT	CAST	On demand	Manually
AES-CFB8 - Encrypt (A6469)	KAT	CAST	On demand	Manually
AES-CFB8 - Encrypt (A6470)	KAT	CAST	On demand	Manually
AES-CFB8 - Encrypt (A6475)	KAT	CAST	On demand	Manually
AES-CFB8 - Encrypt (A6481)	KAT	CAST	On demand	Manually
AES-CFB8 - Encrypt (A6482)	KAT	CAST	On demand	Manually
AES-CFB8 - Encrypt (A6487)	KAT	CAST	On demand	Manually
AES-CFB8 - Decrypt (A6469)	KAT	CAST	On demand	Manually
AES-CFB8 - Decrypt (A6470)	KAT	CAST	On demand	Manually
AES-CFB8 - Decrypt (A6475)	KAT	CAST	On demand	Manually
AES-CFB8 - Decrypt (A6481)	KAT	CAST	On demand	Manually
AES-CFB8 - Decrypt (A6482)	KAT	CAST	On demand	Manually
AES-CFB8 - Decrypt (A6487)	KAT	CAST	On demand	Manually
AES-XTS Testing Revision 2.0 - Encrypt (A6473)	KAT	CAST	On demand	Manually
AES-XTS Testing Revision 2.0 - Encrypt (A6485)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-XTS Testing Revision 2.0 - Decrypt (A6473)	KAT	CAST	On demand	Manually
AES-XTS Testing Revision 2.0 - Decrypt (A6485)	KAT	CAST	On demand	Manually
SHA3-224 (A6468)	KAT	CAST	On demand	Manually
SHA3-224 (A6474)	KAT	CAST	On demand	Manually
SHA3-224 (A6480)	KAT	CAST	On demand	Manually
SHA3-224 (A6486)	KAT	CAST	On demand	Manually
SHA3-256 (A6468)	KAT	CAST	On demand	Manually
SHA3-256 (A6474)	KAT	CAST	On demand	Manually
SHA3-256 (A6480)	KAT	CAST	On demand	Manually
SHA3-256 (A6486)	KAT	CAST	On demand	Manually
SHA3-384 (A6468)	KAT	CAST	On demand	Manually
SHA3-384 (A6474)	KAT	CAST	On demand	Manually
SHA3-384 (A6480)	KAT	CAST	On demand	Manually
SHA3-384 (A6486)	KAT	CAST	On demand	Manually
SHA3-512 (A6468)	KAT	CAST	On demand	Manually
SHA3-512 (A6474)	KAT	CAST	On demand	Manually
SHA3-512 (A6480)	KAT	CAST	On demand	Manually
SHA3-512 (A6486)	KAT	CAST	On demand	Manually
HMAC-SHA-1 (A6467)	KAT	CAST	On demand	Manually
HMAC-SHA-1 (A6472)	KAT	CAST	On demand	Manually
HMAC-SHA-1 (A6479)	KAT	CAST	On demand	Manually
HMAC-SHA-1 (A6484)	KAT	CAST	On demand	Manually
HMAC-SHA2-224 (A6467)	KAT	CAST	On demand	Manually
HMAC-SHA2-224 (A6472)	KAT	CAST	On demand	Manually
HMAC-SHA2-224 (A6479)	KAT	CAST	On demand	Manually
HMAC-SHA2-224 (A6484)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A6467)	KAT	CAST	On demand	Manually
HMAC-SHA2-256 (A6472)	KAT	CAST	On demand	Manually
HMAC-SHA2-256 (A6479)	KAT	CAST	On demand	Manually
HMAC-SHA2-256 (A6484)	KAT	CAST	On demand	Manually
HMAC-SHA2-384 (A6467)	KAT	CAST	On demand	Manually
HMAC-SHA2-384 (A6472)	KAT	CAST	On demand	Manually
HMAC-SHA2-384 (A6479)	KAT	CAST	On demand	Manually
HMAC-SHA2-384 (A6484)	KAT	CAST	On demand	Manually
HMAC-SHA2-512 (A6467)	KAT	CAST	On demand	Manually
HMAC-SHA2-512 (A6472)	KAT	CAST	On demand	Manually
HMAC-SHA2-512 (A6479)	KAT	CAST	On demand	Manually
HMAC-SHA2-512 (A6484)	KAT	CAST	On demand	Manually
AES-CMAC (A6464)	KAT	CAST	On demand	Manually
AES-CMAC (A6467)	KAT	CAST	On demand	Manually
AES-CMAC (A6472)	KAT	CAST	On demand	Manually
AES-CMAC (A6476)	KAT	CAST	On demand	Manually
AES-CMAC (A6479)	KAT	CAST	On demand	Manually
AES-CMAC (A6484)	KAT	CAST	On demand	Manually
RSA SigGen (FIPS186-5) (A6472)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
RSA SigGen (FIPS186-5) (A6484)	KAT	CAST	On demand	Manually
RSA SigVer (FIPS186-5) (A6472)	KAT	CAST	On demand	Manually
RSA SigVer (FIPS186-5) (A6484)	KAT	CAST	On demand	Manually
ECDSA SigGen (FIPS186-5) (A6472)	KAT	CAST	On demand	Manually
ECDSA SigGen (FIPS186-5) (A6484)	KAT	CAST	On demand	Manually
ECDSA SigVer (FIPS186-5) (A6472)	KAT	CAST	On demand	Manually
ECDSA SigVer (FIPS186-5) (A6484)	KAT	CAST	On demand	Manually
Counter DRBG (A6472)	KAT	CAST	On demand	Manually
Counter DRBG (A6484)	KAT	CAST	On demand	Manually
KAS-FFC-SSC Sp800-56Ar3 (A6472)	KAT	CAST	On demand	Manually
KAS-FFC-SSC Sp800-56Ar3 (A6484)	KAT	CAST	On demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A6472)	KAT	CAST	On demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A6484)	KAT	CAST	On demand	Manually
TLS v1.2 KDF RFC7627 (A6472)	KAT	CAST	On demand	Manually
TLS v1.2 KDF RFC7627 (A6484)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
PBKDF (A6472)	KAT	CAST	On demand	Manually
PBKDF (A6484)	KAT	CAST	On demand	Manually
KDA HKDF Sp800-56Cr1 (A6471)	KAT	CAST	On demand	Manually
KDA HKDF Sp800-56Cr1 (A6483)	KAT	CAST	On demand	Manually
RSA KeyGen (FIPS186-5) (A6472)	PCT	PCT	On demand	Manually
RSA KeyGen (FIPS186-5) (A6484)	PCT	PCT	On demand	Manually
ECDSA KeyGen (FIPS186-5) (A6472)	PCT	PCT	On demand	Manually
ECDSA KeyGen (FIPS186-5) (A6484)	PCT	PCT	On demand	Manually
Safe Primes Key Generation (A6472)	PCT	PCT	On demand	Manually
Safe Primes Key Generation (A6484)	PCT	PCT	On demand	Manually

Table 23: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Error State	The module stops functioning and ends the application process	When the integrity test or KAT fail; When the KAT of DRBG fails during CASTs; When the newly generated RSA, ECDSA, Diffie-Hellman or EC Diffie-Hellman key pair fails the PCT; When the module is in error	The module must be restarted and perform the pre-operational self-test and the CASTs to recover from these errors.	GNUTLS_E_SELF_TEST_ERROR (-400); GNUTLS_E_RANDOM_FAILED (-206); GNUTLS_E_PK_GENERATION_ERROR (-403); GNUTLS_E_LIB_IN_ERROR_STATE (-402)

Name	Description	Conditions	Recovery Method	Indicator
		state and caller requests cryptographic operations		

Table 24: Error States

When the module fails any pre-operational self-test or conditional test, the module will return an error code to indicate the error and enters error state. Any further cryptographic operations and the data output via the data output interface are inhibited. The calling application can obtain the module state by calling the gnutls_fips140_get_operation_state() API function. The function returns GNUTLS_FIPS140_OP_ERROR if the module is in the Error state.

Self-test errors transition the module into an error state that keeps the module operational but prevents any cryptographic related operations. The module must be restarted and perform the pre-operational self-test and the CASTs to recover from these errors. If failures persist, the module must be re-installed.

10.5 Operator Initiation of Self-Tests

The operator can initiate the pre-operational integrity self-test and cryptographic algorithm self-tests by calling the Self-Test service (via the gnutls_fips140_run_self_tests() function) or by powering-off and reloading the module. The operator can initiate a pairwise consistency self-test by calling the “Asymmetric key generation” service. During the execution of the pre-operational integrity self-test and cryptographic algorithm self-tests, services are not available, and no data output is possible.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

The Crypto Officer can install the RPM packages of the Module:

- For Rocky Linux 8.6: gnutls-3.6.16-6.el8_6.ciqfips.0.11.x86_64.rpm and nettle-3.4.1-7.el8_6.ciqfips.x86_64.rpm
- For Rocky Linux 9.2: gnutls-3.7.6-23.el9_2.ciqfips.3.4.x86_64.rpm and nettle-3.8-3.el9_2.ciqfips.x86_64.rpm

The Crypto Officer can install these using standard tools recommended for the installation of RPM packages on a Rocky Linux system (for example, dnf, rpm). The integrity of the RPM package is automatically verified during the installation, and the Crypto Officer shall not install the RPM package if there is any integrity error.

Before the RPM package of the module is installed, the system must operate in the FIPS-validated configuration. This can be achieved by:

- Starting the installation in the FIPS-validated configuration. Add the fips=1 option to the kernel command line during the system installation. During the software selection stage, do not install any third-party software.
- Switching the system into the 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 command “gnutls-cli --fips140-mode” to check the installed version.

11.2 Administrator Guidance

After installation of the RPM packages of the module, the operator must check the installed version, by calling the "Show Module Name and Version" service, by issuing the commands specified in the Approved Services table and section 11.1. The service should output the following module name and version:

“Rocky Linux 8 GnuTLS Cryptographic Module” and *“rocky8.20241217”* (for Rocky Linux 8.6)

“Rocky Linux 9 GnuTLS Cryptographic Module” and *“rocky9.20241217”* (for Rocky Linux 9.2)

11.3 Non-Administrator Guidance

The module implements only the Crypto Officer. There are no requirements for non-administrator guidance.

11.4 End of Life

For secure sanitization of the cryptographic module, the module must first to be powered off, which will zeroize all keys and CSPs in volatile memory. Then, for actual deprecation, the module shall be upgraded to a newer version that is FIPS 140-3 validated.

The module does not possess persistent storage of SSPs, so further sanitization steps are not required.

12 Mitigation of Other Attacks

The module does not mitigate other attacks.

Appendix A. TLS Cipher Suites

The module supports the following cipher suites for the TLS protocol version 1.0, 1.1, 1.2 and 1.3, compliant with section 3.3.1 of [SP800-52rev2]. Each cipher suite defines the key exchange algorithm, the bulk encryption algorithm (including the symmetric key size) and the MAC algorithm.

Cipher Suite	ID	Reference
TLS_DH_RSA_WITH_AES_128_CBC_SHA	{ 0x00, 0x31 }	RFC3268
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	{ 0x00, 0x33 }	RFC3268
TLS_DH_RSA_WITH_AES_256_CBC_SHA	{ 0x00, 0x37 }	RFC3268
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	{ 0x00, 0x39 }	RFC3268
TLS_DH_RSA_WITH_AES_128_CBC_SHA256	{ 0x00,0x3F }	RFC5246
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	{ 0x00,0x67 }	RFC5246
TLS_DH_RSA_WITH_AES_256_CBC_SHA256	{ 0x00,0x69 }	RFC5246
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	{ 0x00,0x6B }	RFC5246
TLS_PSK_WITH_AES_128_CBC_SHA	{ 0x00, 0x8C }	RFC4279
TLS_PSK_WITH_AES_256_CBC_SHA	{ 0x00, 0x8D }	RFC4279
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	{ 0x00, 0x9E }	RFC5288
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	{ 0x00, 0x9F }	RFC5288
TLS_DH_RSA_WITH_AES_128_GCM_SHA256	{ 0x00, 0xA0 }	RFC5288
TLS_DH_RSA_WITH_AES_256_GCM_SHA384	{ 0x00, 0xA1 }	RFC5288
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x04 }	RFC4492
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x05 }	RFC4492
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x09 }	RFC4492
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x0A }	RFC4492
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x0E }	RFC4492
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x0F }	RFC4492
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x13 }	RFC4492
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x14 }	RFC4492
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x23 }	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x24 }	RFC5289
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x25 }	RFC5289
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x26 }	RFC5289
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x27 }	RFC5289

Cipher Suite	ID	Reference
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x28 }	RFC5289
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x29 }	RFC5289
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x2A }	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	{ 0xC0, 0x2B }	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	{ 0xC0, 0x2C }	RFC5289
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256	{ 0xC0, 0x2D }	RFC5289
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384	{ 0xC0, 0x2E }	RFC5289
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	{ 0xC0, 0x2F }	RFC5289
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	{ 0xC0, 0x30 }	RFC5289
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256	{ 0xC0, 0x31 }	RFC5289
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	{ 0xC0, 0x32 }	RFC5289
TLS_DHE_RSA_WITH_AES_128_CCM	{ 0xC0, 0x9E }	RFC6655
TLS_DHE_RSA_WITH_AES_256_CCM	{ 0xC0, 0x9F }	RFC6655
TLS_DHE_RSA_WITH_AES_128_CCM_8	{ 0xC0, 0xA2 }	RFC6655
TLS_DHE_RSA_WITH_AES_256_CCM_8	{ 0xC0, 0xA3 }	RFC6655
TLS_AES_128_GCM_SHA256	{ 0x13, 0x01 }	RFC8446
TLS_AES_256_GCM_SHA384	{ 0x13, 0x02 }	RFC8446
TLS_AES_128_CCM_SHA256	{ 0x13, 0x04 }	RFC8446
TLS_AES_128_CCM_8_SHA256	{ 0x13, 0x05 }	RFC8446

Appendix B. 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
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 Mode
DF	Derivation Function
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
KAT	Known Answer Test
KW	AES Key Wrap
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
OFB	Output Feedback
PAA	Processor Algorithm Acceleration
PAI	Processor Algorithm Implementation
PBKDF2	Password-based Key Derivation Function v2
PCT	Pair-wise Consistency Test
PKCS	Public-Key Cryptography Standards
PR	Prediction Resistance
PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SSP	Sensitive Security Parameter
XOF	Extendable Output Function
XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

Appendix C. References

FIPS140-3	**FIPS PUB 140-3 - Security Requirements For Cryptographic Modules** March 2019 https://doi.org/10.6028/NIST.FIPS.140-3
FIPS140-3_IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program** October 2024 https://csrc.nist.gov/csrc/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/FIPS%20140-3%20IG.pdf
FIPS140-3_MM	**FIPS 140-3 Cryptographic Module Validation Program - Management Manual (Draft)** December 2022 https://csrc.nist.gov/csrc/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/Draft%20FIPS-140-3-CMVP%20Management%20Manual%20v1.2%20%5BDec%2023%202022%5D.pdf
FIPS180-4	**Secure Hash Standard (SHS)** August 2015 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS186-5	**Digital Signature Standard (DSS)** February 2023 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf
FIPS197	**Advanced Encryption Standard** November 2001 https://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS198-1	**The Keyed Hash Message Authentication Code (HMAC)** July 2008 https://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
FIPS202	**SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions** August 2015 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
PKCS#1	**Public Key Cryptography Standards (PKCS) #1: RSA Cryptography** Specifications Version 2.1 February 2003 https://www.ietf.org/rfc/rfc3447.txt
SP800-38A	**NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf
SP800-38B	**NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38b.pdf
SP800-38C	**NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality** May 2004 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf
SP800-38E	**NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices** January 2010 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf

SP800-38F	**NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP800-90Arev1	**NIST Special Publication 800-90A Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP800-90B	**NIST Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf
SP800-132	**NIST Special Publication 800-132 - Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** December 2010 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf
SP800-133rev2	**NIST Special Publication 800-133 - Recommendation for Cryptographic Key Generation** June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf
SP800-140Br1	**NIST Special Publication 800-140B – Revision 1 - CMVP Security Policy Requirements** November 2023 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140Br1.pdf