Ctrl IQ, Inc.

Rocky Linux 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: 1.0
Last update: 2026-07-15

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for version rocky9.20250825 of the Rocky Linux 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

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 9 GnuTLS Cryptographic Module (hereafter referred to as "the module") is a software module in a multi-chip standalone embodiment. The module is an open-source, general-purpose set of libraries designed to support cross-platform development of security-enabled client and server applications and provides cryptographic services to applications running in the user space of the underlying operating system through a C language Application Program Interface (API).

Module Type: Software

Module Embodiment: Multi-Chip Standalone

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). The depiction in Figure 1 applies to all TOEs listed in the Tested Operational Environment – Software, Firmware, Hybrid table.

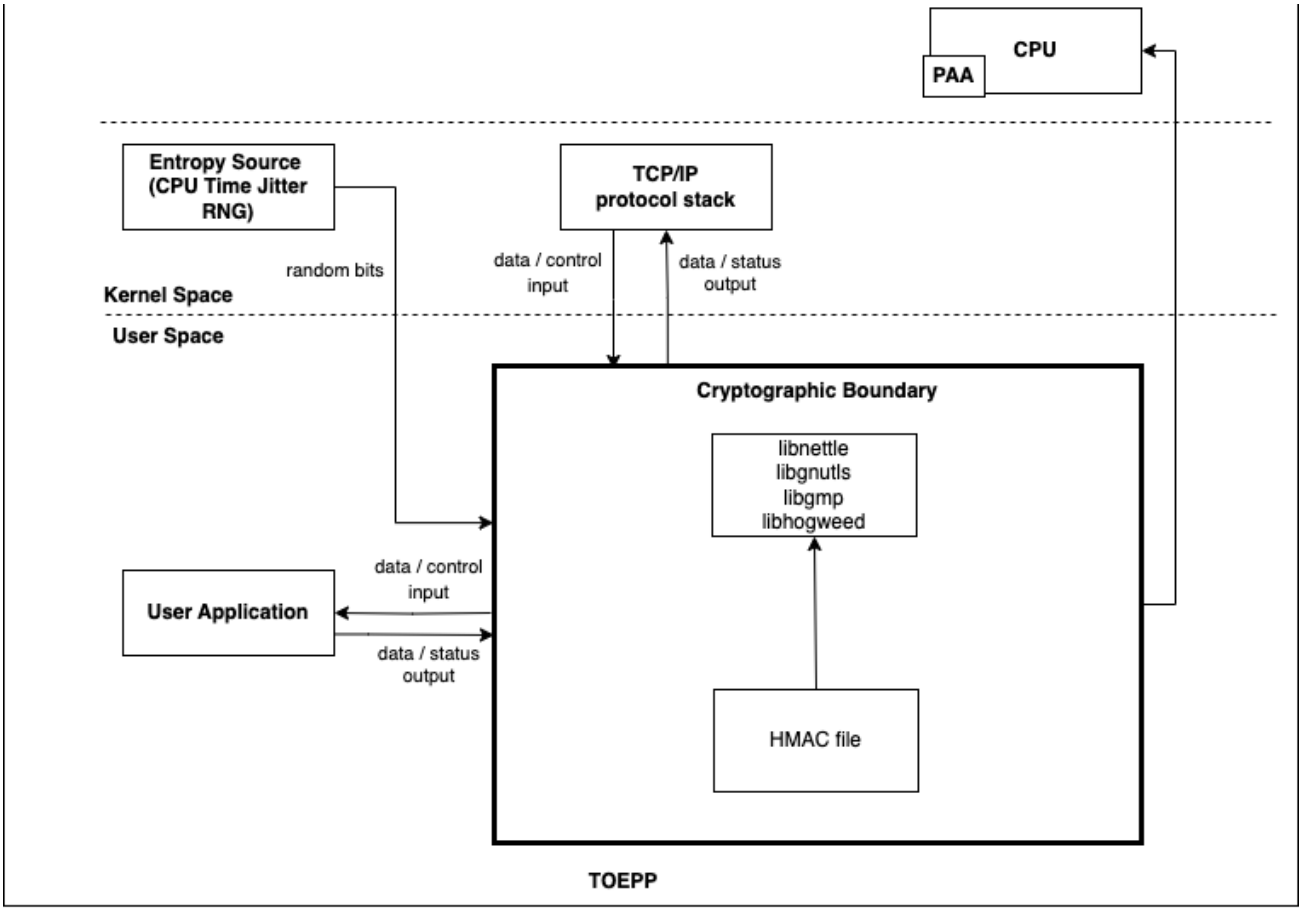

Figure 1: Block Diagram

Tested Operational Environment's Physical Perimeter (TOEPP):

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

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
/usr/lib64/libgnutls.so.30.37.1; /usr/lib64/libnettle.so.8.10; /usr/lib64/libhogweed.so.6.10; /usr/lib64/.libgnutls.so.30.37.1.hmac (Note: libgmp is statically linked to libgnutls)	rocky9.20250825	N/A	HMAC-SHA-256

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

Tested Module Identification – Hybrid Disjoint Hardware:

N/A for this module.

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Rocky Linux 9	SuperMicro SuperServer 5039MS	Intel Xeon E3-1270 v6	Yes	N/A	rocky9.20250825
Rocky Linux 9	SuperMicro SuperServer 5039MS	Intel Xeon E3-1270 v6	No	N/A	rocky9.20250825

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

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

N/A for this module.

2.3 Excluded Components

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

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved Mode	Automatically entered whenever an approved service is requested	Approved	The approved mode indicator maps to the approved service indicator which is GNUTLS_FIPS140_OP_APPROVED and is equivalent to the status of the service requested.
Non-approved Mode	Automatically entered whenever a non-approved service is requested	Non-Approved	The non-approved mode indicator maps to the non-approved service indicator which is GNUTLS_FIPS140_OP_NOT_APPROVED and is equivalent to the status of the service requested.

Table 4: Modes List and Description

When the module starts up successfully, after passing all the pre-operational and conditional 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, Not Allowed Algorithms Table. Please see section 4 for the details on service indicator provided by the module that identifies when an approved service is called.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A7423	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A7424	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A7425	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A7426	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A7431	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A7423	Key Length - 128, 256 Tag Length - 112, 128, 32, 48, 64, 80, 96 IV Length - IV Length: 56, 64, 72, 80, 88, 96, 104	SP 800-38C

Algorithm	CAVP Cert	Properties	Reference
		Payload Length - Payload Length: 0-256 Increment 8 AAD Length - AAD Length: 0, 256, 65536	
AES-CFB8	A7428	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A7429	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A7434	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A7423	Direction - Generation Key Length - 128, 256 MAC Length - MAC Length: 128 Message Length - Message Length: 8-524288 Increment 8	SP 800-38B
AES-CMAC	A7426	Direction - Generation Key Length - 128, 256 MAC Length - MAC Length: 128 Message Length - Message Length: 8-524288 Increment 8	SP 800-38B
AES-CMAC	A7431	Direction - Generation Key Length - 128, 256 MAC Length - MAC Length: 128 Message Length - Message Length: 8-524288 Increment 8	SP 800-38B
AES-ECB	A7435	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A7423	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 Payload Length - Payload Length: 128, 256, 120, 248 AAD Length - AAD Length: 128, 256, 120, 0	SP 800-38D
AES-GCM	A7424	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 Payload Length - Payload Length: 128, 256, 120, 248 AAD Length - AAD Length: 128, 256, 120, 0	SP 800-38D
AES-GCM	A7425	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96	SP 800-38D

Algorithm	CAVP Cert	Properties	Reference
		IV Length - IV Length: 96 Payload Length - Payload Length: 128, 256, 120, 248 AAD Length - AAD Length: 128, 256, 120, 0	
AES-GCM	A7426	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 Payload Length - Payload Length: 128, 256, 120, 248 AAD Length - AAD Length: 128, 256, 120, 0	SP 800-38D
AES-GCM	A7431	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 Payload Length - Payload Length: 128, 256, 120, 248 AAD Length - AAD Length: 128, 256, 120, 0	SP 800-38D
AES-GMAC	A7431	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 AAD Length - AAD Length: 128, 256, 120, 0	SP 800-38D
AES-XTS Testing Revision 2.0	A7432	Direction - Decrypt, Encrypt Key Length - 128, 256 Payload Length - Payload Length: 128-65536 Increment 8 Tweak Mode - Hex Data Unit Length Matches Payload Length - Yes	SP 800-38E
Counter DRBG	A7431	Prediction Resistance - No Supports Reseed - Yes Mode - AES-256 Derivation Function Enabled - No Additional Input - Additional Input: 0, 256 Entropy Input - Entropy Input: 384 Nonce - Nonce: 0 Personalization String Length - Personalization String Length: 0, 256 Returned Bits - 512	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-5)	A7431	Curve - P-256, P-384, P-521 Secret Generation Mode - testing candidates	FIPS 186-5

Algorithm	CAVP Cert	Properties	Reference
ECDSA KeyVer (FIPS186-5)	A7431	Curve - P-256, P-384, P-521	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A7431	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)	A7431	Component - No Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-5
HMAC-SHA2-224	A7426	MAC - MAC: 224 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A7431	MAC - MAC: 224 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A7426	MAC - MAC: 256 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A7431	MAC - MAC: 256 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A7426	MAC - MAC: 384 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A7431	MAC - MAC: 384 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A7426	MAC - MAC: 512 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A7431	MAC - MAC: 512 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A7431	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	A7431	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	A7430	Fixed Info Pattern - uPartyInfo||vPartyInfo Fixed Info Encoding - concatenation Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-65336 Increment 8	SP 800-56C Rev. 2

Algorithm	CAVP Cert	Properties	Reference
		HMAC Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	
PBKDF	A7431	Iteration Count - Iteration Count: 1000-10000 Increment 1 HMAC Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512 Password Length - Password Length: 8-128 Increment 1 Salt Length - Salt Length: 128-4096 Increment 8 Key Data Length - Key Data Length: 112-4096 Increment 8	SP 800-132
RSA KeyGen (FIPS186-5)	A7431	Key Generation Mode - provable Hash Algorithm - SHA2-384 Modulo - 2048, 3072, 4096, 6144, 8192 p mod 8 - 0 q mod 8 - 0 Info Generated By Server - Yes Private Key Format - standard Public Exponent Mode - random	FIPS 186-5
RSA SigGen (FIPS186-5)	A7431	Hash Pair - Hash Algorithm - SHA2-224 Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss Mask Function - mgf1	FIPS 186-5
RSA SigVer (FIPS186-5)	A7431	Hash Pair - Hash Algorithm - SHA2-224 Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss Mask Function - mgf1 Public Exponent Mode - random	FIPS 186-5
Safe Primes Key Generation	A7431	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3
SHA2-224	A7426	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-224	A7431	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A7426	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A7431	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-384	A7426	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4

Algorithm	CAVP Cert	Properties	Reference
SHA2-384	A7431	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A7426	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A7431	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA3-224	A7427	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-224	A7433	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-256	A7427	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-256	A7433	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-384	A7427	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-384	A7433	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-512	A7427	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-512	A7433	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
TLS v1.2 KDF RFC7627 (CVL)	A7431	Hash Algorithm - SHA2-256, SHA2-384 Key Block Length - Key Block Length: 1024	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
Asymmetric Key Pair Generation (CKG)	Type:Asymmetric	N/A	SP 800-133r2, section 4 example 1

Table 6: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
Blowfish	Symmetric encryption; Symmetric decryption
Camellia	Symmetric encryption; Symmetric decryption
CAST	Symmetric encryption; Symmetric decryption
ChaCha20	Symmetric encryption; Symmetric decryption
ChaCha20 and Poly1305	Authenticated encryption; Authenticated decryption
AES-GCM (when not used in the context of the TLS protocol)	Authenticated encryption; Authenticated decryption
DES	Symmetric encryption; Symmetric decryption
Diffie-Hellman with keys generated with domain parameters other than safe primes	Key agreement; Shared secret computation
DRBG when key length is less than 112 bits	Symmetric key generation
DSA	Key generation; Domain parameter generation; Digital signature generation; Digital signature verification
ECDSA with curves not listed in Table "Approved Algorithms"	Key generation; Public key verification
ECDSA with curves not listed in Table "Approved Algorithms" or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512	Digital signature generation; Digital signature verification
EC Diffie-Hellman with curves not listed in Table "Approved Algorithms"	Key agreement; Shared secret computation
GOST	Symmetric encryption; Symmetric decryption; Message digest
HMAC with keys smaller than 112-bit	Message authentication code (MAC)
HMAC with GOST	Message authentication code (MAC)
MD2	Message digest; Message authentication code (MAC)
MD4	Message digest; Message authentication code (MAC)

Name	Use and Function
MD5	Message digest; Message authentication code (MAC)
Non-supported cipher suites (not listed in Appendix A)	Transport Layer Security (TLS) Network Protocol
PBKDF with non-approved message digest algorithms, key lengths, salt lengths, minimum iteration counts, password lengths.	Key derivation
RC2	Symmetric encryption; Symmetric decryption
RC4	Symmetric encryption; Symmetric decryption
RMD160	Message digest; Message authentication code (MAC)
RSA (with keys smaller than 2048 bits and/or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512)	Digital signature generation; Digital signature verification
RSA (with keys smaller than 2048 bits)	Key generation
RSA (with any key sizes)	Key encapsulation; Key unencapsulation
Salsa20	Symmetric encryption; Symmetric decryption
SHA-1	Message digest; Message authentication code (MAC)
SM3	Message digest
Serpent	Symmetric encryption; Symmetric decryption
STREEBOG	Message digest; Message authentication code (MAC)
Triple-DES	Symmetric encryption; Symmetric decryption
Twofish	Symmetric encryption; Symmetric decryption
UMAC	Message authentication code (MAC)
Yarrow	Random number generation

Table 7: Non-Approved, Not Allowed Algorithms

The above table lists non-Approved security functions that are not allowed in the approved mode of operation.

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Symmetric Encryption with AES	BC-UnAuth	Encryption using AES		AES-CBC: (A7423, A7424, A7425, A7426, A7431) AES-CFB8: (A7428, A7429, A7434) AES-XTS Testing Revision 2.0: (A7432)
Symmetric Decryption with AES	BC-UnAuth	Decryption using AES		AES-CBC: (A7423, A7424, A7425, A7426, A7431) AES-CFB8: (A7428, A7429, A7434) AES-XTS Testing Revision 2.0: (A7432)
Authenticated Encryption with AES	BC-Auth	Authenticated encryption using AES		AES-CCM: (A7423)
Authenticated Decryption with AES	BC-Auth	Authenticated decryption using AES		AES-CCM: (A7423)
Authenticated Encryption (in 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: (A7423, A7424, A7425, A7426, A7431)
Authenticated Decryption (in 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: (A7423, A7424, A7425, A7426, A7431)
Message Digest with SHA	SHA	Message digest using SHA		SHA2-224: (A7426, A7431) SHA2-256: (A7426, A7431) SHA2-384: (A7426, A7431) SHA2-512: (A7426, A7431) SHA3-224: (A7427,

Name	Type	Description	Properties	Algorithms
				A7433) SHA3-256: (A7427, A7433) SHA3-384: (A7427, A7433) SHA3-512: (A7427, A7433)
Random Number Generation with CTR_DRBG	DRBG	Random number generation using CTR_DRBG		Counter DRBG: (A7431) AES-ECB: (A7435)
Message Authentication Code (MAC) with HMAC	MAC	Message authentication code using HMAC		HMAC-SHA2-224: (A7426, A7431) HMAC-SHA2-256: (A7426, A7431) HMAC-SHA2-384: (A7426, A7431) HMAC-SHA2-512: (A7426, A7431)
Message Authentication Code (MAC) with AES	MAC	Message authentication code using AES		AES-CMAC: (A7423, A7426, A7431) AES-GMAC: (A7431)
Digital Signature Generation with RSA	DigSig-SigGen	Digital signature generation using RSA		RSA SigGen (FIPS186-5): (A7431)
Digital Signature Verification with RSA	DigSig-SigVer	Digital signature verification using RSA		RSA SigVer (FIPS186-5): (A7431)
Digital Signature Generation with ECDSA	DigSig-SigGen	Digital signature generation using ECDSA		ECDSA SigGen (FIPS186-5): (A7431)
Digital Signature Verification with ECDSA	DigSig-SigVer	Digital signature verification using ECDSA		ECDSA SigVer (FIPS186-5): (A7431)
Public Key Verification with ECDSA	AsymKeyPair-KeyVer	Public key verification using ECDSA"		ECDSA KeyVer (FIPS186-5): (A7431)
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: (A7431)

Name	Type	Description	Properties	Algorithms
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: (A7431)
Key Derivation with PBKDF	PBKDF	Key derivation using PBKDF	PBKDF Derived key:112 to 256 bits of key strength	PBKDF: (A7431)
Key Derivation with TLS 1.2 KDF	KAS-135KDF	Key derivation using TLS KDF (CVL) / TLS v1.2 KDF RFC7627 (CVL)	TLS Derived Secret:112-256 with 112-256 bits of key strength	TLS v1.2 KDF RFC7627: (A7431)
Key Derivation (as part of TLSv1.3) with KDA HKDF	KAS-56CKDF	Key derivation using KDA HKDF	HKDF Derived key:128-256 bits key with 128-256 bits of key strength	KDA HKDF Sp800-56Cr1: (A7430)
TLS Handshake	KAS-Full	Key Agreement	IG:IG D.F scenario 2 path (2) Key confirmation:No Caveat:Key establishment methodology provides between 112 and 256 bits of security strength	KAS-ECC-SSC Sp800-56Ar3: (A7431) KAS-FFC-SSC Sp800-56Ar3: (A7431) KDA HKDF Sp800-56Cr1: (A7430) TLS v1.2 KDF RFC7627: (A7431)
Key Pair Generation with RSA	AsymKeyPair-KeyGen CKG	Key Generation using RSA		RSA KeyGen (FIPS186-5): (A7431) Asymmetric Key Pair Generation (CKG): () Key type: Asymmetric
Key Pair Generation with ECDSA	AsymKeyPair-KeyGen CKG	Generate ECDSA key pairs		ECDSA KeyGen (FIPS186-5): (A7431) Asymmetric Key Pair Generation (CKG): () Key type: Asymmetric

Name	Type	Description	Properties	Algorithms
Key Pair Generation with Safe Primes	AsymKeyPair-KeyGen CKG	Key Pair Generation with Safe Primes		Safe Primes Key Generation: (A7431) Asymmetric Key Pair Generation (CKG): () Key type: Asymmetric

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 AES-XTS

The AES algorithm in XTS mode can be only used for the cryptographic protection of data on storage devices, as specified in [SP800-38E]. The length of a single data unit encrypted with the XTS-AES 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. 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.

Note: AES-XTS shall be used with 128 and 256-bit keys only.

2.7.2 AES-GCM IV

The module implements AES GCM for being used in the TLS v1.2 and v1.3 protocols. AES GCM IV generation is compliant with [FIPS140-3_IG] 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 [SP800-52rev2].
- 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 [SP800-52rev2].

The IV generated in both scenarios is only used within the context of the TLS protocol implementation.

The module implements the TLS protocol within the boundary. The design of the TLS protocol in this module implicitly ensures that the 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.

2.7.3 Key Derivation using SP 800-132 PBKDF2

The module provides password-based key derivation (PBKDF), compliant with SP800-132 and IG D.N. The module supports option 1a from section 5.4 of [SP800-132], in which the Master Key (MK) or a segment of it is used directly as the Data Protection Key (DPK).

In accordance with [SP800-132], 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 module only allows use of MK or DPK with at least 112 bits.
- The module only allows a portion of the salt with a length of at least 128 bits, which shall be generated randomly using the SP800- 90Arev1 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 allowed by the module is 1000.
- Passwords or passphrases, used as an input for the PBKDF, shall not be used as cryptographic keys.
- The module only allows password or passphrase with at least 20 characters, consisting 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}.

2.7.4 SP 800-56ARev3 Assurances

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.

2.7.5 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 RSA key generation has been tested for all module lengths available in CAVP testing: 2048, 3072, 4096, 6144, and 8192-bits. 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. The number of Miller-Rabin tests is consistent with the bit sizes of p and q from Table B.1 of FIPS 186-5.

2.7.6 RSA Signature Generation and Signature Verification

The module provides RSA signature generation and signature verification compliant 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.

2.7.7 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.

2.7.8 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 - ECDSA SigGen, ECDSA SigVer, HMAC, KDA HKDF Sp800-56Cr1, KDF TLS (CVL), 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.

2.7.8 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.

2.7.9 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.

2.7.10 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).

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 9 on Intel Kaby Lake Xeon E3-1270 v6	256 bits	Full entropy	SHA3-256 cert. A5837; SHA2-512-HMAC-DRBG cert. A5837

Table 10: Entropy Sources

The module employs a Deterministic Random Bit Generator (DRBG) based on [SP800- 90ARev1] for the generation of random value used in asymmetric keys, and for providing a RNG service to calling applications. The approved DRBG provided by the module is the CTR_DRBG with AES-256. The DRBG does not employ

prediction resistance or a derivation function. The module uses an SP800-90B-compliant Entropy Source specified in the table above to seed the DRBG. The entropy source is located within the module's physical perimeter, but outside the module's cryptographic boundary.

The DRBG is instantiated with a 384-bits long entropy input (corresponding to 384 bits of entropy). Additionally, the DRBG is reseeded with a 256-bits long entropy input (corresponding to 256 bits of entropy).

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, API input parameters for control.
N/A	Status Output	API return codes, API output parameters for status output.

Table 11: Ports and Interfaces

All data output via data output interface is inhibited when the module is performing pre- operational self-test, conditional cryptographic algorithms self-tests, zeroization or when the module enters error state.

The module does not implement a control output interface.

The module does not output any control data to another cryptographic module.

4 Roles, Services, and Authentication

4.1 Authentication Methods

The module does not support authentication.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 12: Roles

The module supports the Crypto Officer role only. This sole role is implicitly and always assumed by the operator of the module.

No support is provided for multiple concurrent operators.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Symmetric Encryption	Perform AES encryption	GNUTLS_FIPS140_OP_APPROVED	Key, Plaintext	Ciphertext	Symmetric Encryption with AES	Crypto Officer - AES key: W,E
Symmetric Decryption	Perform AES decryption	GNUTLS_FIPS140_OP_APPROVED	Key, Ciphertext	Plaintext	Symmetric Decryption with AES	Crypto Officer - AES key: W,E
Authenticated Encryption	Encrypt a plaintext	GNUTLS_FIPS140_OP_APPROVED	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	Key, Ciphertext, IV, MAC tag	Plaintext or fail	Authenticated Decryption with AES	Crypto Officer - AES key: W,E
Message Digest	Compute message digest	GNUTLS_FIPS140_OP_APPROVED	Message	Digest of the message	Message Digest with SHA	Crypto Officer
Random Number Generation	Generate random bitstrings	GNUTLS_FIPS140_OP_APPROVED	Number of bits	Random number	Random Number Generation with	Crypto Officer - Entropy Input: W,E,Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					CTR_DRBG	- DRBG seed: G,E,Z - DRBG internal state (V value, key): G,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 (MAC) with AES	Compute AES-base CMAC or GMAC	GNUTLS_FIPS140_OP_APPROVED	AES key, message	Message authentication code	Message Authentication Code (MAC) with AES	Crypto Officer - AES key: W,E
Shared Secret Computation with Diffie-Hellman	Compute a shared secret	GNUTLS_FIPS140_OP_APPROVED	Private key, public key from peer	Shared secret	Shared Secret Computation with Diffie-Hellman	Crypto Officer - Diffie-Hellman shared secret: G,R - Diffie-Hellman 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	Private key, public key from peer	Shared secret	Shared Secret Computation with EC Diffie-Hellman	Crypto Officer - EC Diffie-Hellman shared secret:

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

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
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-generated ECDSA public key; Module-generated EC Diffie-Hellman private key, Module-generated EC Diffie-Hellman public key	Random Number Generation with CTR_DRBG Key Pair Generation with ECDSA	Crypto Officer - Module-generated ECDSA private 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

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						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
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
HKDF Key Derivation (derivation of HKDF	Perform key derivation using HKDF	GNUTLS_FIPS140_OP_APPROVED	Shared Secret	HKDF Derived key	Key Derivation (as part of TLSv1.3)	Crypto Officer - Diffie-Hellman shared secret:

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Derived key)					with KDA HKDF	W,E - EC Diffie-Hellman shared secret: W,E - HKDF Derived key: G,R
Key Derivation with PBKDF	Perform password-based key derivation	GNUTLS_FIPS140_OP_APPROVED	Password or passphrase	PBKDF Derived key	Key Derivation with PBKDF	Crypto Officer - PBKDF password or passphrase: W,E - PBKDF 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 Message Digest with SHA Message Authentication Code (MAC) with HMAC Message	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,G - TLS Pre-master Secret: E,G

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					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 Public Key Verification with ECDSA TLS Handshake Key Pair Generation with RSA Key Pair Generation with ECDSA Key Pair Generation with Safe Primes Authenticated Encryption (in context of the TLS 1.2/1.3	- 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
					protocol) with AES-GCM Authenticated Decryption (in context of the TLS 1.2/1.3 protocol) with AES-GCM	
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 Message Digest with SHA Random Number Generation with CTR_DRBG Message Authentication Code (MAC) with HMAC Message Authentication Code (MAC)	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					with AES Digital Signature Generation with RSA Digital Signature Verification with RSA Digital Signature Generation with ECDSA Digital Signature Verification with 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	

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					(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 Authenticated Encryption (in context of the TLS 1.2/1.3 protocol) with AES-GCM Authenticated Decryption (in context of the TLS 1.2/1.3 protocol) with AES-GCM	
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

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
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 - Intermediate key generation value: Z - Module-generated ECDSA private key: Z - Module-generated ECDSA

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						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-Hellman private key: Z - Module-generated Diffie-Hellman public key: Z - Diffie-Hellman private

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

Table 13: Approved Services

The above table lists all approved services that can be used in the approved mode of operation.

For the above table, the convention below applies when specifying the access permissions (types) that the service has for each SSP.

- **Generate (G):** The module generates or derives the SSP.
- **Read (R):** The SSP is read from the module (e.g. the SSP is output).

- **Write (W):** The SSP is updated, imported, or written to the module.
- **Execute (E):** The module uses the SSP in performing a cryptographic operation.
- **Zeroize (Z):** The module zeroizes the SSP.

The service indicator is invoked by calling the function "gnutls_fips140_get_operation_state()" and it returns "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 key generation	Generate symmetric key other than AES and HMAC keys	DRBG when key length is less than 112 bits	CO
Symmetric encryption	Compute the cipher for encryption	Blowfish Camellia CAST ChaCha20 DES GOST RC2 RC4 Salsa20 Serpent Triple-DES Twofish	CO
Symmetric decryption	Compute the cipher for decryption	Blowfish Camellia CAST ChaCha20 DES GOST RC2 RC4 Salsa20 Serpent Triple-DES Twofish	CO
Digital signature generation	Sign RSA, DSA, and ECDSA signatures	DSA ECDSA with curves not listed in Table "Approved Algorithms" or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512 RSA (with keys smaller than 2048 bits	CO

Name	Description	Algorithms	Role
		and/or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512)	
Digital signature verification	Verify RSA, DSA, and ECDSA signatures	DSA ECDSA with curves not listed in Table "Approved Algorithms" 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)	CO
Key generation	Generate RSA, DSA, and ECDSA key pairs	DSA ECDSA with curves not listed in Table "Approved Algorithms" RSA (with keys smaller than 2048 bits)	CO
Public key verification	Verify ECDSA public key	ECDSA with curves not listed in Table "Approved Algorithms"	CO
Message digest	Compute message digest	SHA-1 GOST MD2 MD4 MD5 RMD160 SM3 STREEBOG	CO
Message Authentication Code (MAC)	Compute HMAC	HMAC with keys smaller than 112-bit HMAC with GOST SHA-1 MD2 MD4 MD5 RMD160 STREEBOG UMAC	CO
Key encapsulation	Perform RSA key encapsulation	RSA (with any key sizes)	CO
Key unencapsulation	Perform RSA key unencapsulation	RSA (with any key sizes)	CO
Shared Secret Computation with Diffie-Hellman	Perform DH key agreement	Diffie-Hellman with keys generated with domain parameters other than safe primes	CO

Name	Description	Algorithms	Role
Shared Secret Computation with EC Diffie-Hellman	Perform ECDH key agreement	EC Diffie-Hellman with curves not listed in Table "Approved Algorithms"	CO
Key agreement	Perform DH (or ECDH) key agreement	Diffie-Hellman with keys generated with domain parameters other than safe primes EC Diffie-Hellman with curves not listed in Table "Approved Algorithms"	CO
Key Derivation with PBKDF	Perform password-based key derivation	PBKDF with non-approved message digest algorithms, key lengths, salt lengths, minimum iteration counts, password lengths.	CO
Transport Layer Security (TLS) Network Protocol	Provide non-supported cipher suites	Non-supported cipher suites (not listed in Appendix A)	CO
Random number generation	Generate random number	Yarrow	CO
Authenticated encryption	Perform authenticated encryption	ChaCha20 and Poly1305 AES-GCM (when not used in the context of the TLS protocol)	CO
Authenticated decryption	Perform authenticated decryption	ChaCha20 and Poly1305 AES-GCM (when not used in the context of the TLS protocol)	CO
Domain parameter generation	Generate domain parameter	DSA	CO

Table 14: Non-Approved Services

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

4.5 External Software/Firmware Loaded

The module does not load external software or firmware.

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. The .hmac file has HMAC value for libgnutls, libnettle and libhogweed listed in section 2. 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 module provides the Self-Test service to perform self-tests on demand which includes the pre-operational test (i.e., integrity test) and the cryptographic algorithm self-tests (CASTs). The Self-tests service can be called on demand by invoking the gnutls_fips140_run_self_tests() function which will perform integrity tests and the cryptographic algorithms self-tests.

Additionally, the Self-test service can be invoked by powering-off and reloading the module. During the execution of the on-demand self-tests, 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

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

7 Physical Security

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

8 Non-Invasive Security

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

9 Sensitive Security Parameters Management

9.1 Storage Areas

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

Table 15: Storage Areas

The SSPs are temporarily stored in the RAM in plaintext form. SSPs are provided to the module by the calling process and are destroyed when released by the appropriate zeroization function calls.

Symmetric keys, public and private keys are provided to the module by the calling application via API input parameters and are destroyed by the module when invoking the appropriate API function calls.

9.2 SSP Input-Output Methods

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

Table 16: SSP Input-Output Methods

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 [FIPS140-3_IG] 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 operating system calls.	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable. The completion of the zeroization routine indicates that 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() gnutls_rsa_params_deinit() ECDSA Public Key, ECDSA Private Key: gnutls_privkey_deinit() gnutls_x509_privkey_deinit()

Zeroization Method	Description	Rationale	Operator Initiation
		zeroization procedure succeeded.	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

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;	Symmetric key - CSP			Symmetric Encryption with AES Symmetric Decryption with AES Authenticated Encryption with AES

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		Other modes: 128, 192, 256 bits - AES-XTS, AES-CCM, AES-GCM, AES-CMAC: 128, 256 bits; Other modes: 128, 192, 256 bits				Authenticated Decryption with AES Message Authentication Code (MAC) with AES
GCM IV	IV used for Authenticated Encryption with AES and Authenticated Decryption with AES	96 bits - N/A	Initialization vector - PSP	Authenticated Encryption with AES		Authenticated Encryption (in context of the TLS 1.2/1.3 protocol) with AES-GCM Authenticated Decryption (in context of the TLS 1.2/1.3 protocol) with AES-GCM
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	2048-15360 bits - 112-256 bits	Private key - CSP	Key Pair Generation with RSA		

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Module-generated RSA public key	RSA public key generated through asymmetric key generation	2048-15360 bits - 112-256 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
PBKDF password or passphrase	Password used to derive symmetric keys	20 characters minimum - 10^(-20) minimum probability	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		
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	P-256, P-384, P-	Public key - PSP			Public Key Verification

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	signature verification	521 - 128, 192, 256 bits				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
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,	Private key - CSP	Key Pair Generation with Safe Primes		TLS Handshake

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		176, 200 bits				
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, 176, 200 bits	Shared Secret - CSP		Shared Secret Computation with Diffie-Hellman	
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	
Entropy Input	Entropy input used to seed the DRBG	256-384 bits - 256-384 bits	Entropy Input - CSP			Random Number Generation with CTR_DRBG

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DRBG seed	DRBG seed derived from entropy input	256-384 bits - 128-256 bits	Seed - CSP	Random Number Generation with CTR_DRBG		Random Number Generation with CTR_DRBG
DRBG internal state (V value, key)	Internal state of the CTR_DRBG (for IG D.L)	384 bits - 256 bits	Internal State - CSP	Random Number Generation with CTR_DRBG		Random Number Generation with CTR_DRBG
TLS Pre-master Secret	TLS Pre-master Secret used for deriving the TLS Master Secret	112 to 256 bits - 112 to 256 bits	Secret - CSP		Shared Secret Computation with EC Diffie-Hellman Shared Secret Computation with Diffie-Hellman	TLS Handshake
TLS Master Secret	TLS Master Secret used for deriving the TLS Derived Secret	384 bits - 112-256 bits	Secret - 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
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
Intermediate key generation value	Temporary value generated during key generation services	256-15360 bits - 112-256 bits	Intermediate value - CSP	Key Pair Generation with RSA Key Pair Generation with ECDSA		Key Pair Generation with RSA Key Pair Generation with ECDSA

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
				Key Pair Generation with Safe Primes		Key Pair Generation with Safe Primes

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	
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
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

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
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
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-	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Module-generated Diffie-Hellman private key:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Hellman public key					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
Diffie-Hellman shared secret	API output parameters API input 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 API input 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
Entropy Input		RAM:Plaintext	From generation until DRBG Seed is created	Zeroize Context Automatic	DRBG seed:Derives
DRBG seed		RAM:Plaintext	While the DRBG is instantiated	Zeroize Context Automatic	Entropy Input:Derived From
DRBG internal state (V value, key)		RAM:Plaintext	While the module is operational	Zeroize Context Reset	DRBG seed:Used With
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

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
					key:Established by Module-generated Diffie-Hellman public key:Established by Module-generated EC Diffie-Hellman private key:Established by Module-generated EC 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
HKDF Derived key	API output parameters	RAM:Plaintext	For the duration of the service	Zeroize Context Reset	Diffie-Hellman shared secret:Derived From EC Diffie-Hellman shared secret:Derived From
Intermediate key generation value		RAM:Plaintext	For the duration of the service	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

Table 19: SSP Table 2

10 Self-Tests

10.1 Pre-Operational Self-Tests

The module performs the following pre-operational tests: the integrity test of the shared libraries that comprise the module using HMAC-SHA2-256. The details of integrity test are provided in section 5.1.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A7426)	256-bit key	Integrity	SW/FW Integrity	Module becomes operational and services are available for use	MAC tag computation
HMAC-SHA2-256 (A7431)	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 the pre-operational self-test and CASTs automatically when the module is loaded into memory. Pre-operational self-test ensure that the module is not corrupted, and the CASTs ensure that the cryptographic algorithms work as expected. While the module is executing the self-tests, the module services are not available, and input and output are inhibited. The module is not available for use by the calling application until the pre-operational self-test and the CASTs are completed successfully. After the pre-operational test and the CASTs succeed, the module becomes operational. If any of the pre-operational test or any of the CASTs fail an error message is returned, and the module transitions to the error state. When the module is in the Error state, no data is output, and cryptographic operations are not allowed.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC - Encrypt (A7423)	128-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC - Encrypt (A7424)	128-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC - Encrypt (A7425)	128-bit key	KAT	CAST	Module becomes operational	Encryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				and services are available for use		
AES-CBC - Encrypt (A7426)	128-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC - Encrypt (A7431)	128-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC - Decrypt (A7423)	128-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CBC - Decrypt (A7424)	128-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CBC - Decrypt (A7425)	128-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CBC - Decrypt (A7426)	128-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC - Decrypt (A7431)	128-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM - Encrypt (A7423)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Encrypt (A7424)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Encrypt (A7425)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Encrypt (A7426)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Encrypt (A7431)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM - Decrypt (A7423)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-GCM - Decrypt (A7424)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM - Decrypt (A7425)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM - Decrypt (A7426)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM - Decrypt (A7431)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CFB8 - Encrypt (A7428)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CFB8 - Encrypt (A7429)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CFB8 - Encrypt (A7434)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CFB8 - Decrypt (A7428)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CFB8 - Decrypt (A7429)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CFB8 - Decrypt (A7434)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-XTS Testing Revision 2.0 - Encrypt (A7432)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-XTS Testing Revision 2.0 - Decryption (A7432)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
SHA3-224 (A7427)	32-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-224 (A7433)	32-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA3-256 (A7427)	32-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-256 (A7433)	32-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-384 (A7427)	64-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-384 (A7433)	64-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-512 (A7427)	136-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
SHA3-512 (A7433)	136-bit message	KAT	CAST	Module becomes operational and services are available for use	Message digest	Module initialization
HMAC-SHA2-224 (A7426)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-224 (A7431)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-256 (A7426)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-256 (A7431)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-384 (A7426)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-384 (A7431)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-512 (A7426)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization
HMAC-SHA2-512 (A7431)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message authentication	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CMAC (A7423)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	MAC generation	Module initialization
AES-CMAC (A7426)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	MAC generation	Module initialization
AES-CMAC (A7431)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	MAC generation	Module initialization
RSA SigGen (A7431)	2048-bit key using SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Digital signature generation	Module initialization
RSA SigVer (A7431)	2048-bit key using SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Digital signature verification	Module initialization
ECDSA SigGen (A7431)	P-256 using SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Digital signature generation	Module initialization
ECDSA SigVer (A7431)	P-256 using SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Digital signature verification	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
Counter DRBG (A7431)	256-bit keys without DF, without PR	KAT	CAST	Module becomes operational and services are available for use	KAT CTR_DRBG with AES with 256-bit keys without DF, without PR	Test runs at power-on before the integrity test (instantiate, generate and reseed functions in compliance with SP800-90Ar1, Section 11)
KAS-FFC-SSC Sp800-56Ar3 (A7431)	ffdhe3072	KAT	CAST	Module becomes operational and services are available for use	Primitive "Z" Computation	Module initialization
KAS-ECC-SSC Sp800-56Ar3 (A7431)	P-256	KAT	CAST	Module becomes operational and services are available for use	Primitive "Z" Computation	Module initialization
TLS v1.2 KDF RFC7627 (A7431)	SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Industry-based TLS v1.2 KDF key derivation	Module initialization
PBKDF (A7431)	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
KDA HKDF Sp800-56Cr1 (A7430)	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 (A7431)	SHA2-256	PCT	PCT	Successful key pair generation	Signature generation and verification	Key Pair Generation

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
ECDSA KeyGen (A7431)	SHA-256 with the respective curve	PCT	PCT	Successful key pair generation	Signature generation and verification	Key Pair Generation
Safe Primes Key Generation (A7431)	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 (A7426)	Integrity	SW/FW Integrity	On demand	Manual
HMAC-SHA2-256 (A7431)	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 (A7423)	KAT	CAST	On demand	Manually
AES-CBC - Encrypt (A7424)	KAT	CAST	On demand	Manually
AES-CBC - Encrypt (A7425)	KAT	CAST	On demand	Manually
AES-CBC - Encrypt (A7426)	KAT	CAST	On demand	Manually
AES-CBC - Encrypt (A7431)	KAT	CAST	On demand	Manually
AES-CBC - Decrypt (A7423)	KAT	CAST	On demand	Manually
AES-CBC - Decrypt (A7424)	KAT	CAST	On demand	Manually
AES-CBC - Decrypt (A7425)	KAT	CAST	On demand	Manually
AES-CBC - Decrypt (A7426)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC - Decrypt (A7431)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A7423)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A7424)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A7425)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A7426)	KAT	CAST	On demand	Manually
AES-GCM - Encrypt (A7431)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A7423)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A7424)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A7425)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A7426)	KAT	CAST	On demand	Manually
AES-GCM - Decrypt (A7431)	KAT	CAST	On demand	Manually
AES-CFB8 - Encrypt (A7428)	KAT	CAST	On demand	Manually
AES-CFB8 - Encrypt (A7429)	KAT	CAST	On demand	Manually
AES-CFB8 - Encrypt (A7434)	KAT	CAST	On demand	Manually
AES-CFB8 - Decrypt (A7428)	KAT	CAST	On demand	Manually
AES-CFB8 - Decrypt (A7429)	KAT	CAST	On demand	Manually
AES-CFB8 - Decrypt (A7434)	KAT	CAST	On demand	Manually
AES-XTS Testing Revision 2.0 - Encrypt (A7432)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-XTS Testing Revision 2.0 - Decryption (A7432)	KAT	CAST	On demand	Manually
SHA3-224 (A7427)	KAT	CAST	On demand	Manually
SHA3-224 (A7433)	KAT	CAST	On demand	Manually
SHA3-256 (A7427)	KAT	CAST	On demand	Manually
SHA3-256 (A7433)	KAT	CAST	On demand	Manually
SHA3-384 (A7427)	KAT	CAST	On demand	Manually
SHA3-384 (A7433)	KAT	CAST	On demand	Manually
SHA3-512 (A7427)	KAT	CAST	On demand	Manually
SHA3-512 (A7433)	KAT	CAST	On demand	Manually
HMAC-SHA2-224 (A7426)	KAT	CAST	On demand	Manually
HMAC-SHA2-224 (A7431)	KAT	CAST	On demand	Manually
HMAC-SHA2-256 (A7426)	KAT	CAST	On demand	Manually
HMAC-SHA2-256 (A7431)	KAT	CAST	On demand	Manually
HMAC-SHA2-384 (A7426)	KAT	CAST	On demand	Manually
HMAC-SHA2-384 (A7431)	KAT	CAST	On demand	Manually
HMAC-SHA2-512 (A7426)	KAT	CAST	On demand	Manually
HMAC-SHA2-512 (A7431)	KAT	CAST	On demand	Manually
AES-CMAC (A7423)	KAT	CAST	On demand	Manually
AES-CMAC (A7426)	KAT	CAST	On demand	Manually
AES-CMAC (A7431)	KAT	CAST	On demand	Manually
RSA SigGen (A7431)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
RSA SigVer (A7431)	KAT	CAST	On demand	Manually
ECDSA SigGen (A7431)	KAT	CAST	On demand	Manually
ECDSA SigVer (A7431)	KAT	CAST	On demand	Manually
Counter DRBG (A7431)	KAT	CAST	On demand	Manually
KAS-FFC-SSC Sp800-56Ar3 (A7431)	KAT	CAST	On demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A7431)	KAT	CAST	On demand	Manually
TLS v1.2 KDF RFC7627 (A7431)	KAT	CAST	On demand	Manually
PBKDF (A7431)	KAT	CAST	On demand	Manually
KDA HKDF Sp800-56Cr1 (A7430)	KAT	CAST	On demand	Manually
RSA KeyGen (A7431)	PCT	PCT	On demand	Manually
ECDSA KeyGen (A7431)	PCT	PCT	On demand	Manually
Safe Primes Key Generation (A7431)	PCT	PCT	On demand	Manually

Table 23: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Error State	Prevents any cryptographic related operations and data output	When the integrity test or KAT fail; When the KAT of DRBG fails during CASTs; When the newly generated RSA, ECDSA, Diffie-Hellman or	The module must be restarted and perform the pre-operational self-test and the CASTs to	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
		EC Diffie-Hellman key pair fails the PCT; When the module is in error state and caller requests cryptographic operations	recover from these errors.	

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 PCTs can be invoked on demand by requesting the Key Pair 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

11.1.1 Configuration of the Operating Environment

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

- For Rocky Linux 9.6:
    - gnutls-3.8.3-6.el9_6ciqfips.1.3.x86_64.rpm
    - nettle-3.10-1.el9_6ciqfips.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

The binaries of the module are contained in the RPM packages for delivery, listed in section 11.1.1. The Crypto Officer shall follow section 11 to configure the operational environment and install the module to be operated as a FIPS 140-3 validated module.

The "Show module name and version" service returns the value:

“library is in FIPS140-3 mode

FIPS module name: Rocky Linux 9 GnuTLS Cryptographic Module Version rocky9.20250825

Version: rocky9.20250825”.

11.3 Non-Administrator Guidance

The approved security functions are listed in section 2.6 of this security policy.

The logical interfaces available to the users of the cryptographic module are listed in section 3.1.

For the secure operation of the module, the operator must follow the instructions in section 11.1 of this security policy.

11.4 End of Life

For secure sanitization of the cryptographic module, the module needs 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 needed.

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 1.2 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
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x28 }	RFC5289

Cipher Suite	ID	Reference
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
AES-NI	Advanced Encryption Standard New Instructions
API	Application Programming Interface
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CKG	Cryptographic Key Generation
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DF	Derivation Function
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
GMAC	Galois Counter Mode Message Authentication Code
HKDF	HMAC-based Key Derivation Function
HMAC	Hash Message Authentication Code
IKE	Internet Key Exchange
KAS	Key Agreement Scheme
KAT	Known Answer Test

KTS	Key Transport Scheme
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
PAA	Processor Algorithm Acceleration
PAI	Processor Algorithm Implementation
PBKDF2	Password-based Key Derivation Function v2
PCT	Pairwise 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
SSC	Shared Secret Computation
SSP	Sensitive Security Parameter
TLS	Transport Layer Security

Appendix C. References

FIPS 140-3	**FIPS PUB 140-3 - Security Requirements For Cryptographic Modules** March 22, 2019 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf
FIPS 140-3 IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program** April 18, 2025 https://csrc.nist.gov/csrc/media/Projects/cryptographic-module-validation-program/documents/fips 140-3/FIPS 140-3 IG.pdf
FIPS 140-3 Management Manual	**FIPS 140-3 Cryptographic Module Validation Program Management Manual** May 14, 2025 https://csrc.nist.gov/csrc/media/Projects/cryptographic-module-validation-program/documents/fips 140-3/FIPS-140-3-CMVP Management Manual.pdf
FIPS 180-4	**Secure Hash Standard (SHS)** August 2015 https://doi.org/10.6028/NIST.FIPS.180-4
FIPS 186-5	**Digital Signature Standard (DSS)** February 3, 2023 https://doi.org/10.6028/NIST.FIPS.186-5
FIPS 197	**Advanced Encryption Standard** May 9, 2023 https://doi.org/10.6028/NIST.FIPS.197-upd1
FIPS 198-1	**The Keyed Hash Message Authentication Code (HMAC)** July 2008 https://doi.org/10.6028/NIST.FIPS.198-1
FIPS 202	**SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions** August 2015 https://doi.org/10.6028/NIST.FIPS.202
PKCS#1	**Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.2** November 2016 https://www.rfc-editor.org/rfc/rfc8017.txt
RFC 3526	**More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)** May 2003 https://www.ietf.org/rfc/rfc3526.txt

RFC 5288	**AES Galois Counter Mode (GCM) Cipher Suites for TLS** August 2008 https://www.ietf.org/rfc/rfc5288.txt
RFC 7919	**Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)** August 2016 https://www.ietf.org/rfc/rfc7919.txt
RFC 8446	**The Transport Layer Security (TLS) Protocol Version 1.3** August 2018 https://www.ietf.org/rfc/rfc8446.txt
RFC 7627	**Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension)** September 2015 https://www.ietf.org/rfc/rfc7627.txt
SP 800-38A	**Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 https://doi.org/10.6028/NIST.SP.800-38A
SP 800-38B	**Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 https://doi.org/10.6028/NIST.SP.800-38B
SP 800-38D	**Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 https://doi.org/10.6028/NIST.SP.800-38A
SP 800-52 Rev. 2	**Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations** August 2019 https://doi.org/10.6028/NIST.SP.800-52r2
SP 800-56A Rev. 3	**Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography** April 2018 https://doi.org/10.6028/NIST.SP.800-56Ar3
SP 800-56C Rev. 2	**Recommendation for Key-Derivation Methods in Key-Establishment Schemes** August 2020 https://doi.org/10.6028/NIST.SP.800-56Cr2

SP 800-90A Rev. 1	**Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 https://doi.org/10.6028/NIST.SP.800-90Ar1
SP 800-90B	**Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://doi.org/10.6028/NIST.SP.800-90B
SP 800-132	**Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** December 2010 https://doi.org/10.6028/NIST.SP.800-132
SP 800-133 Rev. 2	**Recommendation for Cryptographic Key Generation** June 2020 https://doi.org/10.6028/NIST.SP.800-133r2
SP 800-135 Rev. 1	**Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 https://doi.org/10.6028/NIST.SP.800-135r1