Amazon Web Services, Inc.

Amazon Linux 2023 GnuTLS Cryptographic Module

FIPS 140-3 Non-Proprietary Security Policy

Document Version 1.4
Last update: 2026-08-03

Prepared by:

atsec information security corporation

4516 Seton Center Pkwy, Suite 250

Austin, TX 78759

www.atsec.com

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 the Amazon Linux 2023 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.

1.2 Security Levels

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

Table 1: Security Levels

1.3 Additional Information

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

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.

2 Cryptographic Module Specification

2.1 Description

Purpose and Use: The Amazon Linux 2023 GnuTLS Cryptographic Module (hereafter referred to as "the module") is a cryptographic module that 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 within the module and between the module and operator (depicted through the arrows).

The module components consist of the libgnutls.so.30, libnettle.so.8, libhogweed.so.6, and libgmp.so.10, which are verified by computing their HMAC values and comparing the computed value with the stored .libgnutls.so.30.hmac, .libnettle.so.8.hmac, .libhogweed.so.6.hmac, and .libgmp.so.10.hmac values.

Tested Operational Environment's Physical Perimeter (TOEPP): The TOEPP of the module is defined as the general-purpose computer on which the module is installed.

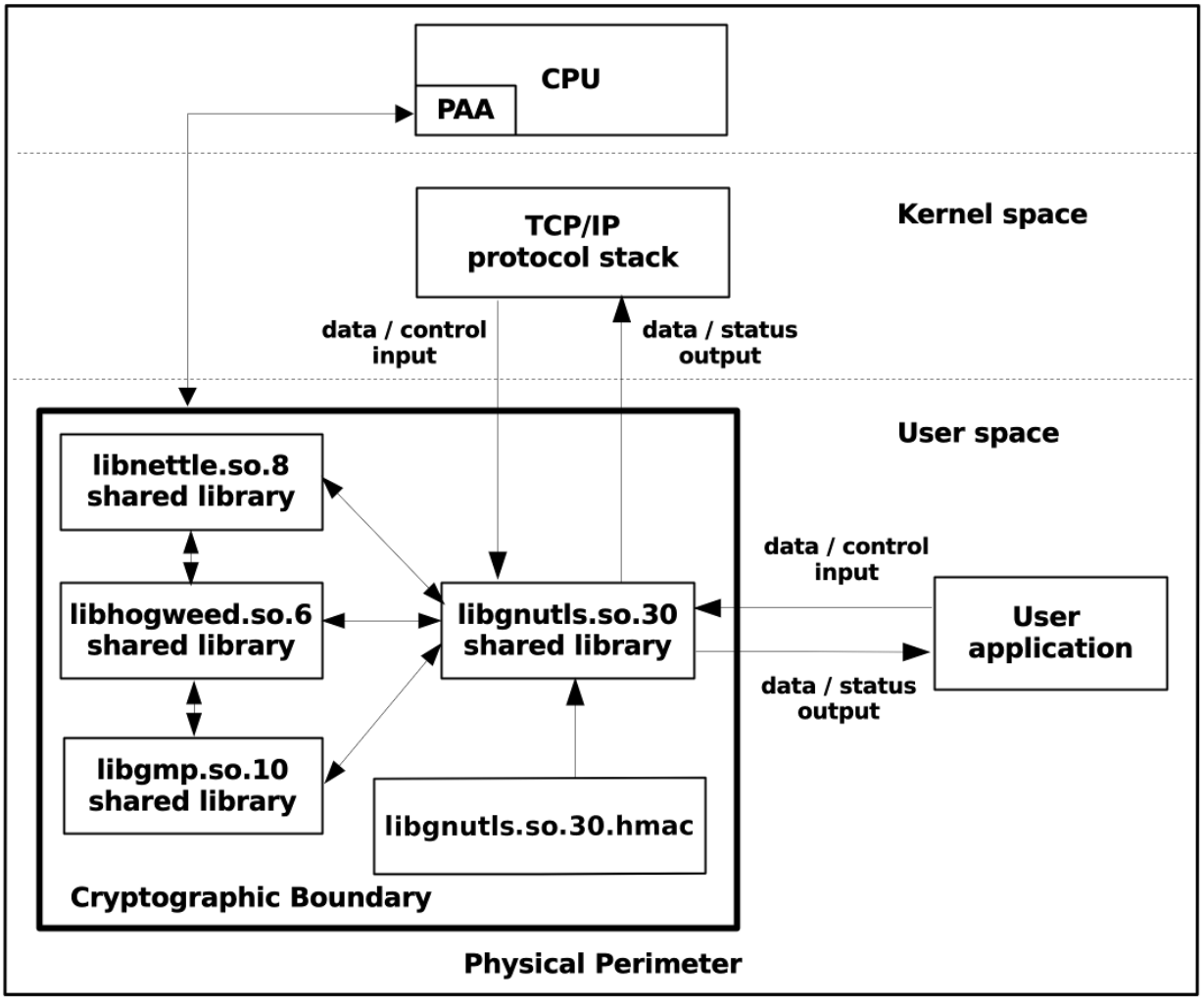

Figure 1: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Hardware:

N/A for this module.

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libgnutls.so.30, libnettle.so.8, libhogweed.so.6 on EC2 c7g.metal with AWS Graviton3 (Note: libgmp is statically linked to libgnutls)	3.8.3-98ce29ed6c83f8a2	N/A	HMAC-SHA2-256
libgnutls.so.30, libnettle.so.8, libhogweed.so.6 on EC2 c6i.metal with Intel Xeon Platinum 8375C	3.8.3-98ce29ed6c83f8a2	N/A	HMAC-SHA2-256

Package or File Name	Software/ Firmware Version	Features	Integrity Test
(Note: libgmp is statically linked to libgnutls)			

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: The module has been tested on the following platforms with the corresponding module variants and configuration options with and without PAA:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Amazon Linux 2023	EC2 c7g.metal	AWS Graviton3 (ARMv8-A)	Yes	N/A	3.8.3-98ce29ed6c83f8a2
Amazon Linux 2023	EC2 c6i.metal	Intel Xeon Platinum 8375C (Sunny Cove)	Yes	N/A	3.8.3-98ce29ed6c83f8a2
Amazon Linux 2023	EC2 c7g.metal	AWS Graviton3 (ARMv8-A)	No	N/A	3.8.3-98ce29ed6c83f8a2
Amazon Linux 2023	EC2 c6i.metal	Intel Xeon Platinum 8375C (Sunny Cove)	No	N/A	3.8.3-98ce29ed6c83f8a2

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

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

Operating System	Hardware Platform
Amazon Linux 2023	AWS Snowball with AMD EPYC 9R14
Amazon Linux 2023	AWS Snowball with AMD EPYC 7702
Amazon Linux 2023	AWS Snowcone with Intel Denverton Atom C3558

Table 4: Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid

2.3 Excluded Components

The module does not claim any excluded components.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved mode of operation	Entered by default, after passing the pre-operational and all conditional cryptographic algorithms self-tests (CASTs). Also, automatically entered whenever an approved service is requested	Approved	Equivalent to the indicator of the requested service as defined in section 4.3
Non-approved mode of operation	Automatically entered whenever a non-approved service is requested	Non-Approved	Equivalent to the indicator of the requested service as defined in section 4.4

Table 5: Modes List and Description

When the module starts up successfully, after passing the pre-operational and all conditional cryptographic algorithms self-tests (CASTs), the module is operating in the approved mode of operation by default and can only be transitioned into the non-approved mode by calling one of the non-approved services listed in the Non-Approved Services table. Please see Section 4 or 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	A7464, A7466, A7515, A7516, A7517, A7518	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A7464, A7515	Key Length - 128, 256 Tag Length - 112, 128, 32, 48, 64, 80, 96 IV Length - IV Length: 56, 64, 72, 80, 88, 96, 104 Payload Length - Payload Length: 0-256 Increment 8 AAD Length - AAD Length: 0, 256, 65536	SP 800-38C
AES-CFB8	A7469, A7520, A7521	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A7466, A7515, A7518	Direction - Generation Key Length - 128, 256 MAC Length - MAC Length: 128 Message Length - Message Length: 8-524288 Increment 8	SP 800-38B
AES-GCM	A7464, A7466, A7515, A7516, A7517, A7518	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	A7466	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	A7467	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	A7466	Prediction Resistance - No Supports Reseed - Yes Mode - AES-256 Derivation Function Enabled - No Additional Input - Additional Input: 0, 256 Entropy Input - Entropy Input: 384	SP 800-90A Rev. 1

Algorithm	CAVP Cert	Properties	Reference
		Nonce - Nonce: 0 Personalization String Length - Personalization String Length: 0, 256 Returned Bits - 512	
ECDSA KeyGen (FIPS186-5)	A7466	Curve - P-256, P-384, P-521 Secret Generation Mode - testing candidates	FIPS 186-5
ECDSA KeyVer (FIPS186-5)	A7466	Curve - P-256, P-384, P-521	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A7466	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)	A7466	Component - No Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-5
HMAC-SHA-1	A7464, A7466, A7518	MAC - MAC: 160 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A7464, A7466, A7518	MAC - MAC: 224 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A7464, A7466, A7518	MAC - MAC: 256 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A7464, A7466, A7518	MAC - MAC: 384 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A7464, A7466, A7518	MAC - MAC: 512 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A7466	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	A7466	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	A7465	Fixed Info Pattern - uPartyInfo||vPartyInfo Fixed Info Encoding - concatenation Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-65336 Increment 8 HMAC Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	SP 800-56C Rev. 2
KDF TLS (CVL)	A7466	TLS Version - v1.0/1.1	SP 800-135 Rev. 1
PBKDF	A7466	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)	A7466	Key Generation Mode - provable Hash Algorithm - SHA2-384 Modulo - 2048, 3072, 4096, 6144, 8192	FIPS 186-5

Algorithm	CAVP Cert	Properties	Reference
		p mod 8 - 0 Primality Tests - 2powSecStr q mod 8 - 0 Info Generated By Server - Yes Private Key Format - standard Public Exponent Mode - random	
RSA SigGen (FIPS186-5)	A7466	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)	A7466	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	A7466	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3
SHA-1	A7464, A7466, A7518	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-224	A7464, A7466, A7518	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A7464, A7466, A7518	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-384	A7464, A7466, A7518	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A7464, A7466, A7518	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA3-224	A7468, A7519	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-256	A7468, A7519	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-384	A7468, A7519	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-512	A7468, A7519	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
TLS v1.2 KDF RFC7627 (CVL)	A7466	Hash Algorithm - SHA2-256, SHA2-384 Key Block Length - Key Block Length: 1024	SP 800-135 Rev. 1

Table 6: Approved Algorithms

Vendor-Affirmed Algorithms:

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

Table 7: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

Name	Caveat	Use and Function
MD5	Only allowed per IG 2.4.A	Message digest used in TLS 1.0 / 1.1 KDF only

Table 8: Non-Approved, Allowed Algorithms with No Security Claimed

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
Blowfish	Symmetric Encryption; Symmetric Decryption - Not FIPS 140-3 compliant
Camellia	Symmetric Encryption; Symmetric Decryption - Not FIPS 140-3 compliant
CAST	Symmetric Encryption; Symmetric Decryption - Not FIPS 140-3 compliant
Chacha20 and Poly1305	Authenticated Encryption; Authenticated Decryption - Not FIPS 140-3 compliant
CMAC with Triple-DES	Message Authentication Code (MAC) - Not FIPS 140-3 compliant
DES	Symmetric Encryption; Symmetric Decryption - Not FIPS 140-3 compliant
Diffie-Hellman with keys generated with domain parameters other than safe primes	Key Agreement; Shared Secret Computation- Not SP 800-56Br2 compliant
DSA	Key Generation; Domain Parameter Generation; Digital Signature Generation; Digital Signature Verification - Not FIPS 140-3 compliant
ECDSA with curves not listed in the Approved Algorithms table	Key Generation; Public Key Verification - Not CAVP tested
ECDSA with curves/hash functions not listed in the Approved Algorithms table	Digital Signature Generation; Digital Signature Verification - Not CAVP tested
EC Diffie-Hellman with curves not listed in the Approved Algorithms table	Key Agreement; Shared Secret Computation - Not CAVP tested
GOST	Symmetric Encryption; Symmetric Decryption; Message Digest - Not FIPS 140-3 compliant
HMAC with keys smaller than 112-bit	Message Authentication Code (MAC) - Not FIPS 140-3 compliant
HMAC with GOST	Message Authentication Code (MAC) - Not FIPS 140-3 compliant
MD2, MD4, MD5	Message Digest; Message Authentication Code (MAC) - Not FIPS 140-3 compliant

Name	Use and Function
PBKDF with HMAC not listed in the Approved Algorithms table or using input parameters not meeting requirements stated in section 2.7	Key Derivation - Not FIPS 198-1 compliant
RC2, RC4	Symmetric Encryption; Symmetric Decryption - Not FIPS 140-3 compliant
RMD160	Message Digest; Message Authentication Code (MAC) - Not FIPS 140-3 compliant
RSA with keys smaller than 2048 bits or greater than 4096 bits.	Key Generation - Not CAVP tested
RSA with keys smaller than 2048 bits or greater than 4096 bits and/or hash functions not listed in the Approved Algorithms table	Digital Signature Generation - Not CAVP tested
RSA with keys smaller than 2048 bits or greater than 4096 bits and/or hash functions not listed in the Approved Algorithms table	Digital Signature Verification - Not CAVP tested
Salsa20	Symmetric Encryption; Symmetric Decryption - Not FIPS 140-3 compliant
SEED	Symmetric Encryption; Symmetric Decryption - Not FIPS 140-3 compliant
Serpent	Symmetric Encryption; Symmetric Decryption - Not FIPS 140-3 compliant
SRP	Key Agreement - Not FIPS 140-3 compliant
STREEBOG	Message Digest; Message Authentication Code (MAC) - Not FIPS 140-3 compliant
Triple-DES	Symmetric Encryption; Symmetric Decryption - Not FIPS 140-3 compliant
Twofish	Symmetric Encryption; Symmetric Decryption - Not FIPS 140-3 compliant
UMAC	Message Authentication Code (MAC) - Not FIPS 140-3 compliant
Yarrow	Random Number Generation - Not FIPS 140-3 compliant
DRBG when key length is less than 112 bits	Random Number Generation - Not SP 800-56Br2 compliant
RSA	Key Encapsulation; Key Un-encapsulation - Not CAVP tested
Non-supported cipher suites (not listed in Appendix A)	Transport Layer Security (TLS) Network Protocol - Not CAVP tested

Table 9: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Symmetric Encryption with AES	BC-UnAuth	Symmetric encryption using AES		AES-CBC: (A7464, A7466, A7515, A7516, A7517, A7518) AES-CFB8: (A7469, A7520, A7521) AES-XTS Testing Revision 2.0: (A7467)

Name	Type	Description	Properties	Algorithms
Symmetric Decryption with AES	BC-UnAuth	Symmetric decryption using AES		AES-CBC: (A7464, A7466, A7515, A7516, A7517, A7518) AES-CFB8: (A7469, A7520, A7521) AES-XTS Testing Revision 2.0: (A7467)
Authenticated Symmetric Encryption with AES	BC-Auth	Authenticated symmetric encryption using AES		AES-CCM: (A7464, A7515) AES-GCM: (A7464, A7466, A7515, A7516, A7517, A7518)
Authenticated Symmetric Decryption with AES	BC-Auth	Authenticated symmetric decryption using AES		AES-CCM: (A7464, A7515) AES-GCM: (A7464, A7466, A7515, A7516, A7517, A7518)
Message Authentication Code Generation with AES	MAC	Message authentication code Generation with AES		AES-CMAC: (A7466, A7515, A7518) AES-GMAC: (A7466)
Message Authentication Code Verification with AES-GCM	MAC	Message authentication code Generation with AES		AES-GMAC: (A7466)
Message Authentication Code with HMAC	MAC	Message authentication code with HMAC		HMAC-SHA-1: (A7464, A7466, A7518) HMAC-SHA2-224: (A7464, A7466, A7518) HMAC-SHA2-256: (A7464, A7466, A7518) HMAC-SHA2-384: (A7464, A7466, A7518) HMAC-SHA2-512: (A7464, A7466, A7518)
Message Digest with SHA	SHA	Message digest using SHA		SHA-1: (A7464, A7466, A7518) SHA2-224: (A7464, A7466, A7518) SHA2-256: (A7464, A7466, A7518) SHA2-384: (A7464, A7466, A7518) SHA2-512: (A7464, A7466, A7518) SHA3-224: (A7468, A7519) SHA3-256: (A7468, A7519) SHA3-384: (A7468, A7519) SHA3-512: (A7468, A7519)

Name	Type	Description	Properties	Algorithms
Key Derivation with TLS KDF	KAS-135KDF	Key derivation using TLS KDF		KDF TLS: (A7466) TLS v1.2 KDF RFC7627: (A7466)
Key Derivation with KDA HKDF	KAS-56CKDF	Key derivation using KDA HKDF		KDA HKDF Sp800-56Cr1: (A7465)
Key Derivation with PBKDF	PBKDF	Key derivation using PBKDF		PBKDF: (A7466)
Digital Signature Generation with RSA	DigSig-SigGen	Digital signature generation using RSA		RSA SigGen (FIPS186-5): (A7466)
Digital Signature Generation with ECDSA	DigSig-SigGen	Digital signature generation using ECDSA		ECDSA SigGen (FIPS186-5): (A7466)
Digital Signature Verification with RSA	DigSig-SigVer	Digital signature verification using RSA		RSA SigVer (FIPS186-5): (A7466)
Digital Signature Verification with ECDSA	DigSig-SigVer	Digital signature verification using ECDSA		ECDSA SigVer (FIPS186-5): (A7466)
Key Pair Generation with RSA	AsymKeyPair-KeyGen CKG	Key pair generation using RSA		RSA KeyGen (FIPS186-5): (A7466) Cryptographic Key Generation (CKG): ()
Key Pair Generation with ECDSA	AsymKeyPair-KeyGen CKG	Key pair generation using ECDSA		ECDSA KeyGen (FIPS186-5): (A7466) Cryptographic Key Generation (CKG): ()
Key Pair Generation with Safe Primes	AsymKeyPair-KeyGen CKG	Key pair generation using Safe Primes		Safe Primes Key Generation: (A7466) Cryptographic Key Generation (CKG): ()
Public Key Verification with ECDSA	AsymKeyPair-KeyVer	Public key verification using ECDSA		ECDSA KeyVer (FIPS186-5): (A7466)
Shared Secret Computation with KAS-ECC-SSC	KAS-SSC	Shared secret computation using KAS-ECC-SSC		KAS-ECC-SSC Sp800-56Ar3: (A7466)
Shared Secret Computation with KAS-FFC-SSC	KAS-SSC	Shared secret computation using KAS-FFC-SSC		KAS-FFC-SSC Sp800-56Ar3: (A7466)
Random Number Generation with Counter DRBG	DRBG	Random number generation using CTR_DRBG		Counter DRBG: (A7466)
TLS Handshake	KAS-Full	Key agreement		KAS-ECC-SSC Sp800-56Ar3: (A7466) KAS-FFC-SSC Sp800-56Ar3: (A7466) KDA HKDF Sp800-56Cr1: (A7465)
Symmetric Key Generation with Counter DRBG	CKG DRBG	Symmetric key generation using Counter DRBG		Counter DRBG: (A7466) Cryptographic Key Generation (CKG): ()

Name	Type	Description	Properties	Algorithms
Key Wrapping with AES	KTS-Wrap	Key wrapping using AES (as part of the cipher suite in the TLS protocol)	Keys:128, 256-bit keys with 128 or 256 bits of key strength Compliance:IG D.G	AES-CCM: (A7464, A7515) AES-GCM: (A7464, A7466, A7515, A7516, A7517, A7518)
Key Unwrapping with AES	KTS-Wrap	Key unwrapping using AES (as part of the cipher suite in the TLS protocol)	Keys:128, 256-bit keys with 128 or 256 bits of key strength Compliance:IG D.G	AES-CCM: (A7464, A7515) AES-GCM: (A7464, A7466, A7515, A7516, A7517, A7518)
Key Wrapping with AES and HMAC	KTS-Wrap	Key wrapping using AES and HMAC (as part of the cipher suite in the TLS protocol)	Keys:128, 256-bit keys with 128 or 256 bits of key strength Compliance:IG D.G	AES-CBC: (A7464, A7466, A7515, A7516, A7517, A7518) HMAC-SHA-1: (A7464, A7466, A7518) HMAC-SHA2-224: (A7464, A7466, A7518) HMAC-SHA2-256: (A7464, A7466, A7518) HMAC-SHA2-384: (A7464, A7466, A7518) HMAC-SHA2-512: (A7464, A7466, A7518)
Key Unwrapping with AES and HMAC	KTS-Wrap	Key unwrapping using AES and HMAC (as part of the cipher suite in the TLS protocol)	Keys:128, 256-bit keys with 128 or 256 bits of key strength Compliance:IG D.G	AES-CBC: (A7464, A7466, A7515, A7516, A7517, A7518) HMAC-SHA-1: (A7464, A7466, A7518) HMAC-SHA2-224: (A7464, A7466, A7518) HMAC-SHA2-256: (A7464, A7466, A7518) HMAC-SHA2-384: (A7464, A7466, A7518) HMAC-SHA2-512: (A7464, A7466, A7518)

Table 10: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 TLS

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 800-131Ar2.

2.7.2 AES XTS

The AES algorithm in XTS mode can be only used for the cryptographic protection of data on storage devices, as specified in SP 800-38E. The length of a single data unit encrypted with the 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. Key_1 and Key_2 shall be generated and/or established independently according to the rules for component symmetric keys from NIST SP 800-133r2, Section 6.3.

AES-XTS shall be used with 128 and 256-bit keys only. AES-XTS with 192-bit keys is not an approved algorithm.

2.7.3 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 in compliance with FIPS 140-3 IG C.H for both protocols as follows:

- For TLS v1.2, IV generation is in compliance with scenario 1.a of IG C.H and RFC5288. The module supports acceptable AES-GCM ciphersuites from section 3.3.1 of SP 800-52r2.
- For TLS v1.3, IV generation is in compliance with scenario 5 of IG C.H and RFC8446. The module supports acceptable AES-GCM ciphersuites from section 3.3.1 of SP 800-52r2.

The IV generated in both scenarios is only used within the context of the TLS protocol implementation. The nonce_explicit part of the IV does not exhaust the maximum number of possible values for a given session key. The design of the TLS protocol in this module implicitly ensures that the nonce_explicit, or counter portion of the IV will not exhaust all of its possible values.

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

2.7.4 Key Derivation Using SP800-132 PBKDF

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

In accordance with SP 800-132, the module ensures that the following requirements are met when running the PBKDF approved service.

- The length of the MK or DPK is 112 bits or more.
- A portion of the salt, with a length of at least 128 bits (it shall be generated randomly using the SP 800-90Ar1 DRBG).
- The minimum value of the iteration count is 1000 (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 length of the password or passphrase is of at least 20 characters (it shall consist of lower-case, upper-case and numeric characters). The probability of guessing the value is estimated to be $1/62^{20} = 10^{-36}$, which is less than 2^{-112}. If the password or passphrase only consists of numeric characters, the probability of guessing the value is estimated to be 10^{-20}.

Passwords or passphrases, used as an input for the PBKDF, shall not be used as cryptographic keys. Derived keys shall only be used in storage applications. The Master Key (MK) shall not be used for other purposes. The calling application shall also observe the rest of the requirements and recommendations specified in SP 800-132.

2.7.5 Compliance to SP 800-56Ar3 Assurances

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

1. The entity using the module, must use the module's "Asymmetric Key Generation" service for generating DH/ECDH ephemeral keys. This meets the assurances required by key pair owner defined in the section 5.6.2.1 of SP 800-56Ar3.
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 800-56Ar3.
3. The module does not support static keys therefore the "assurance of peer's possession of private key" is not applicable.

2.7.7 SHA-1

Digital signature generation using SHA-1 is non-approved and not allowed in approved services.

2.8 RBG and Entropy

Cert Number	Vendor Name
E124	Amazon

Table 11: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Userspace CPU Time Jitter RNG Entropy Source	Non-Physical	Amazon Linux 2023 on EC2 c7g.metal with AWS Graviton3; Amazon Linux 2023 on EC2 c6i.metal with Intel Xeon Platinum 8375C	256 bits	Full entropy	SHA3-256 (A4551); HMAC-SHA2-512 DRBG (A4551)

Table 12: Entropy Sources

The module employs a Deterministic Random Bit Generator (DRBG) based on SP 800-90Ar1 for the creation of seeds for symmetric keys, asymmetric keys, random numbers for security functions (e.g. ECDSA signature generation), and server and client random numbers for the TLS protocol. In addition, the module provides a Random Number Generation service to calling applications.

The DRBG supports the CTR_DRBG with AES-256, without a derivation function and without prediction resistance. The module uses an SP 800-90B-compliant entropy source specified in the Entropy Sources table. This entropy source is located within the physical perimeter, but outside of the cryptographic boundary of the module. The module obtains 384 bits to seed the DRBG, and 256 bits to reseed it, sufficient to provide a DRBG with 256 bits of security strength. Outputs of multiple GetEntropy() calls are concatenated to receive the entropy input length greater than 256 bits. The output is truncated to get the entropy input string which is not a multiple of 256.

2.9 Key Generation

The module implements key generation methods according to SP 800-133r2 section 4 example 1, without the use of V. The key generation methods are specified in the *Vendor Affirmed Algorithms* table and the *Security Function Implementations* table.

Additionally, the module implements key derivation methods according to section 6.2 of SP 800-133r2. The key derivation methods are specified in the *Security Function Implementations* table.

2.10 Key Establishment

The module implements SSP agreement, compliant with IG D.F scenario 2(1) and scenario 2(2). Additionally, the module implements SSP transport, compliant with IG D.G. The Key Establishment methods are specified in the *Security Function Implementations* table.

2.11 Industry Protocols

The module implements KDF for the TLS protocol TLSv1.0, TLSv1.1, TLSv1.2.

No parts of the TLS 1.0/1.1/1.2, other than the key derivation functions mentioned above, have been tested by the CAVP and CMVP.

The module implements HKDF for the TLS protocol TLSv1.3.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

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

Table 13: Ports and Interfaces

4 Roles, Services, and Authentication

4.1 Authentication Methods

N/A for this module.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 14: Roles

The module supports the Crypto Officer role only. This sole role is implicitly and always assumed by the operator of the module. No support is provided for multiple concurrent operators.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Symmetric Key Generation	Generate AES or HMAC key	GNUTLS_FIPS140_OP_APPROVED	Key size	Key	Symmetric Key Generation with Counter DRBG	Crypto Officer - Module-generated AES Key: G - Module-generated HMAC Key: G
Symmetric Encryption	Perform AES encryption	GNUTLS_FIPS140_OP_APPROVED	Key, IV (for AEAD), Plaintext	Ciphertext	Symmetric Encryption with AES Authenticated Symmetric Encryption with AES	Crypto Officer - AES Key: W,E - AES-GCM IV: W,E
Symmetric Decryption	Perform AES decryption	GNUTLS_FIPS140_OP_APPROVED	Key, IV (for AEAD), Ciphertext	Plaintext	Symmetric Decryption with AES Authenticated Symmetric Decryption with AES	Crypto Officer - AES Key: W,E - AES-GCM IV: W,E
Key Wrapping	Key wrapping (as part of the cipher suites in the TLS protocol)	GNUTLS_FIPS140_OP_APPROVED	AES key only or AES key and HMAC key, CSP	Wrapped CSP	Key Wrapping with AES Key Wrapping with AES and HMAC	Crypto Officer - AES Key: W,E - HMAC Key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Key Unwrapping	Key Unwrapping (as part of the cipher suites in the TLS protocol)	GNUTLS_FIPS140_OP_APPROVED	AES key only or AES key and HMAC key, Wrapped CSP	CSP	Key Unwrapping with AES Key Unwrapping with AES and HMAC	Crypto Officer - AES Key: W,E - HMAC Key: W,E
Asymmetric Key Generation	Generate RSA or ECDSA key pairs	GNUTLS_FIPS140_OP_APPROVED	RSA key size or Elliptic Curve	Key pair	Key Pair Generation with RSA Key Pair Generation with ECDSA	Crypto Officer - Module-generated RSA Public Key: G,R - Module-generated RSA Private Key: G,R - Module-generated ECDSA Public Key: G,R - Module-generated ECDSA Private Key: G,R - Module-generated EC Diffie-Hellman Public Key: G,R - Module-generated EC Diffie-Hellman Private Key: G,R - Intermediate Key Generation Value: G,E
Diffie-Hellman Key Generation using Safe Primes	Perform DH key agreement with safe primes	GNUTLS_FIPS140_OP_APPROVED	Key size	Key pair	Key Pair Generation with Safe Primes	Crypto Officer - Module-generated Diffie-Hellman Public Key: G - Module-generated Diffie-Hellman

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Private Key: G - Intermediate Key Generation Value: G,E
ECDSA Digital Signature Generation	Generate a digital signature	GNUTLS_FIPS140_OP_APPROVED	Message, hash algorithm, private key	Digital signature	Digital Signature Generation with ECDSA	Crypto Officer - ECDSA Private Key: W,E
RSA Digital Signature Generation	Generate a digital signature	GNUTLS_FIPS140_OP_APPROVED	Message, hash algorithm, private key	Digital signature	Digital Signature Generation with RSA	Crypto Officer - RSA Private Key: W,E
ECDSA Digital Signature Verification	Verify a digital signature	GNUTLS_FIPS140_OP_APPROVED	Digital signature, hash algorithm, public key	Verification result	Digital Signature Verification with ECDSA	Crypto Officer - ECDSA Public Key: W,E
RSA Digital Signature Verification	Verify a digital signature	GNUTLS_FIPS140_OP_APPROVED	Digital signature, hash algorithm, public key	Verification result	Digital Signature Verification with RSA	Crypto Officer - RSA Public Key: W,E
Public Key Verification	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
Random Number Generation	Generate random bit strings	GNUTLS_FIPS140_OP_APPROVED	Number of bits	Random number	Random Number Generation with Counter DRBG	Crypto Officer - Entropy Input: W,E - Counter DRBG Seed: G,E - Counter DRBG Internal State: V Value, Key: G,E
Message Digest	Compute SHA hashes	GNUTLS_FIPS140_OP_APPROVED	Message	Digest of the message	Message Digest with SHA	Crypto Officer
HMAC Message Authentication Code (MAC)	Compute HMAC	GNUTLS_FIPS140_OP_APPROVED	Message, HMAC key	Message authentication code (MAC)	Message Authentication Code with HMAC	Crypto Officer - HMAC Key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
AES Message Authentication Code Generation (MAC)	Compute AES-based CMAC or AES-based GMAC	GNUTLS_FIPS140_OP_APPROVED	Message, AES key	Message authentication code (MAC)	Message Authentication Code Generation with AES	Crypto Officer - AES Key: W,E - AES-GCM IV: W,E
AES Message Authentication Code Verification (MAC)	Verify AES-based GMAC	GNUTLS_FIPS140_OP_APPROVED	Message, AES key	Plaintext	Message Authentication Code Verification with AES-GCM	Crypto Officer - AES Key: W,E - AES-GCM IV: W,E
Diffie-Hellman Shared Secret Computation	Perform DH shared secret computation	GNUTLS_FIPS140_OP_APPROVED	DH private key, DH public key from peer	Shared secret	Shared Secret Computation with KAS-FFC-SSC	Crypto Officer - Diffie-Hellman Public Key: W,E - Diffie-Hellman Private Key: W,E
EC Diffie-Hellman Shared Secret Computation	Perform ECDH shared secret computation	GNUTLS_FIPS140_OP_APPROVED	EC private key, EC public key from peer	Shared secret	Shared Secret Computation with KAS-ECC-SSC	Crypto Officer - EC Diffie-Hellman Public Key: W,E - EC Diffie-Hellman Private Key: W,E
TLS KDF and HKDF Key Derivation (derivation of TLS Master Secret)	Perform key derivation	GNUTLS_FIPS140_OP_APPROVED	TLS pre-master secret	TLS master secret	Key Derivation with TLS KDF Key Derivation with KDA HKDF	Crypto Officer - TLS Pre-master Secret: W,E - TLS Master Secret: G,R
TLS KDF and HKDF Key Derivation (derivation of TLS Derived Secret)	Perform key derivation	GNUTLS_FIPS140_OP_APPROVED	TLS master secret	TLS Derived Secret	Key Derivation with TLS KDF Key Derivation with KDA HKDF	Crypto Officer - TLS Master Secret: W,E - TLS Derived Secret: G,R
PBKDF Key Derivation	Perform password-based key derivation	GNUTLS_FIPS140_OP_APPROVED	Password/passphrase	PBKDF Derived key	Key Derivation with PBKDF	Crypto Officer - PBKDF Password or Passphrase: W,E,Z - PBKDF

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						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 Symmetric Encryption with AES Authenticated Symmetric Decryption with AES Message Authentication Code with HMAC Message Digest with SHA Digital Signature Generation with RSA Digital Signature Generation with ECDSA Digital Signature Verification with RSA Digital Signature Verification with ECDSA Key Pair Generation with Safe Primes Public Key Verification with ECDSA TLS Handshake	Crypto Officer - AES Key: W,E - HMAC Key: W,E - RSA Public Key: W,E - RSA Private Key: W,E - ECDSA Public Key: W,E - ECDSA Private Key: W,E - Module-generated Diffie-Hellman Public Key: G,E - Module-generated Diffie-Hellman Private Key: G,E - Module-generated EC Diffie-Hellman Public Key: G,E - TLS Pre-master Secret: G,E - TLS Master Secret: G,E - TLS Derived Secret: G,R - Intermediate Key Generation Value: G,E
Show Status	Show module status	N/A	N/A	Return codes and/or log messages	None	Crypto Officer
Zeroization	Zeroize SSPs	N/A	Context containing SSPs	N/A	None	Crypto Officer - Module-

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						generated AES Key: Z - AES Key: Z - Module-generated HMAC Key: Z - HMAC Key: Z - Module-generated RSA Public Key: Z - Module-generated RSA Private Key: Z - RSA Public Key: Z - RSA Private Key: Z - Module-generated ECDSA Public Key: Z - Module-generated ECDSA Private Key: Z - ECDSA Public Key: Z - ECDSA Private Key: Z - Module-generated Diffie-Hellman Public Key: Z - Module-generated Diffie-Hellman Private Key: Z - Diffie-Hellman Public Key: Z - Diffie-Hellman Private Key: Z - Module-generated EC Diffie-Hellman

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Public Key: Z - Module-generated EC Diffie-Hellman Private Key: Z - EC Diffie-Hellman Public Key: Z - EC Diffie-Hellman Private Key: Z - (Diffie-Hellman) Shared Secret: Z - (EC Diffie-Hellman) Shared Secret: Z - PBKDF Password or Passphrase: Z - PBKDF Derived Key: Z - Entropy Input: Z - Counter DRBG Seed: Z - Counter DRBG Internal State: V Value, Key: Z - TLS Pre-master Secret: Z - TLS Master Secret: Z - TLS Derived Secret: Z - Intermediate Key Generation Value: Z
Self-tests	Perform self-tests	N/A	Module reset	Result of self-test (pass/fail)	Symmetric Encryption with AES Symmetric Decryption	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					with AES Authenticated Symmetric Encryption with AES Authenticated Symmetric Decryption with AES Message Authentication Code Generation with AES Message Authentication Code Verification with AES-GCM Message Authentication Code with HMAC Message Digest with SHA Key Derivation with TLS KDF Key Derivation with KDA HKDF Key Derivation with PBKDF Digital Signature Generation with RSA Digital Signature Generation with ECDSA Digital Signature Verification with RSA Digital Signature Verification with ECDSA Key Pair Generation with RSA Key Pair Generation with ECDSA Key Pair Generation with Safe	

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					Primes Public Key Verification with ECDSA Shared Secret Computation with KAS-ECC-SSC Shared Secret Computation with KAS-FFC-SSC Random Number Generation with Counter DRBG	
Show Module Name and Version	Show module name and version	N/A	None	Name and version information	None	Crypto Officer

Table 15: Approved Services

The following convention is used to specify access rights to a SSP:

- **G = Generate:** The module generates or derives the SSP.
- **R = Read:** The SSP is read from the module (e.g., the SSP is output).
- **W = Write:** The SSP is updated, imported, or written to the module.
- **E = Execute:** The module uses the SSP in performing a cryptographic operation.
- **Z = Zeroize:** The module zeroizes the SSP.
- **N/A:** the calling application does not access any SSP or key during its operation.

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

The “Indicator” column shows the service indicator API functions that must be used to verify the service indicator for each of the services. The function gnutls_fips140_get_operation_state() indicates GNUTLS_FIPS140_OP_NOT_APPROVED or GNUTLS_FIPS140_OP_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	Crypto Officer
Symmetric Encryption	Compute the cipher for encryption	Blowfish Camellia CAST Chacha20 and Poly1305 DES GOST	Crypto Officer

Name	Description	Algorithms	Role
		RC2, RC4 Salsa20 SEED Serpent Triple-DES Twofish	
Symmetric Decryption	Compute the plaintext for decryption	Blowfish Camellia CAST Chacha20 and Poly1305 DES GOST RC2, RC4 Salsa20 SEED Serpent Triple-DES Twofish	Crypto Officer
Asymmetric Key Generation	Generate key pairs	DSA ECDSA with curves not listed in the Approved Algorithms table RSA with keys smaller than 2048 bits or greater than 4096 bits.	Crypto Officer
Digital Signature Generation	Sign RSA, DSA, and ECDSA signatures	DSA ECDSA with curves/hash functions not listed in the Approved Algorithms table RSA with keys smaller than 2048 bits or greater than 4096 bits and/or hash functions not listed in the Approved Algorithms table	Crypto Officer
Digital Signature Verification	Verify RSA, DSA, and ECDSA signatures	DSA ECDSA with curves/hash functions not listed in the Approved Algorithms table RSA with keys smaller than 2048 bits or greater than 4096 bits and/or hash functions not listed in the Approved Algorithms table	Crypto Officer
Message Digest	Compute message digest	GOST MD2, MD4, MD5 RMD160 STREEBOG	Crypto Officer
Message Authentication Code (MAC)	Compute HMAC or CMAC	CMAC with Triple-DES HMAC with keys smaller than 112-bit HMAC with GOST RMD160 UMAC	Crypto Officer
Key Encapsulation	Perform RSA key encapsulation	RSA	Crypto Officer
Key Un-encapsulation	Perform RSA key un-encapsulation	RSA	Crypto Officer
Diffie-Hellman Shared Secret Computation	Perform DH shared secret computation	Diffie-Hellman with keys generated with domain parameters other than safe primes	Crypto Officer
EC Diffie-Hellman Shared Secret Computation	Perform ECDH shared secret computation	EC Diffie-Hellman with curves not listed in the Approved Algorithms table	Crypto Officer

Name	Description	Algorithms	Role
Key Derivation	Perform key derivation	PBKDF with HMAC not listed in the Approved Algorithms table or using input parameters not meeting requirements stated in section 2.7	Crypto Officer
Transport Layer Security (TLS) Network Protocol	Provide non-supported cipher suites	Non-supported cipher suites (not listed in Appendix A)	Crypto Officer
Random Number Generation	Generate random numbers	Yarrow	Crypto Officer
Key Agreement	Perform key agreement	Diffie-Hellman with keys generated with domain parameters other than safe primes EC Diffie-Hellman with curves not listed in the Approved Algorithms table SRP	Crypto Officer

Table 16: Non-Approved Services

4.5 External Software/Firmware Loaded

The module does not support loading software or firmware from an external source.

5 Software/Firmware Security

5.1 Integrity Techniques

The integrity of the module is verified by comparing an HMAC-SHA2-256 value calculated at run time with the HMAC value stored in the .hmac file that was computed at build time for each software component of the module listed in Section 2. If the HMAC values do not match, the test fails, and the module enters the Error state. The HMAC key used in the integrity checks is embedded in the libgnutls.so.30 file.

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 CASTs. 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 executes on a general-purpose operating system (Amazon Linux 2023), which allows modification, loading, and execution of software that is not part of the validated module. The module should be compiled and installed as stated in Section 11.1.

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.

6.3 Additional Information

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.

7 Physical Security

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

8 Non-Invasive Security

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

9 Sensitive Security Parameters Management

9.1 Storage Areas

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

Table 17: Storage Areas

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 18: SSP Input-Output Methods

The module does not support manual SSP entry or intermediate SSP generation output. The SSPs are provided to the module via API input parameters in plaintext form and output via API output parameters in plaintext form within the physical perimeter of the operational environment. This is allowed by FIPS 140-3 IG 9.5.A, according to the “CM Software to/from App via TOEPP Path” entry in the Key Establishment Table.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Free cipher handle	Zeroizes the SSPs referenced in the function name	Memory occupied by SSPs is overwritten with zeros, which renders the SSP values irretrievable	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() 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() TLS Derived Secret: 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()
Remove power from the module	De-allocates the volatile memory used to store SSPs	Volatile memory used by the module is overwritten within nanoseconds when power is removed	By removing power

Zeroization Method	Description	Rationale	Operator Initiation
Automatic	Automatically zeroized by the module when no longer needed	Memory occupied by SSPs is overwritten with zeros, which renders the SSP values irretrievable	N/A

Table 19: 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
Module-generated AES Key	AES key generated during Symmetric Key Generation	128, 192, 256 bits - 128, 192, 256 bits	Symmetric key - CSP	Symmetric Key Generation with Counter DRBG		
AES Key	AES key used for Symmetric Encryption, Symmetric Decryption and Message Authentication Code	128, 192, 256 bits - 128, 192, 256 bits	Symmetric key - CSP			Symmetric Encryption with AES Symmetric Decryption with AES Authenticated Symmetric Encryption with AES Authenticated Symmetric Decryption with AES Message Authentication Code Generation with AES Message Authentication Code Verification with AES-GCM Key Wrapping with AES Key Unwrapping with AES Key Wrapping with AES and HMAC Key Unwrapping with AES and HMAC
Module-generated HMAC Key	HMAC key generated during Symmetric Key Generation	112-256 bit keys (Increment 8) - 112-256 bits	Authentication key - CSP	Symmetric Key Generation		

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
				with Counter DRBG		
HMAC Key	HMAC key used for computing MAC tags	112-524288 bit keys (Increment 8) - 112-256 bits	Authentication key - CSP			Message Authentication Code with HMAC
Module-generated RSA Public Key	RSA Public key generated during Asymmetric Key Generation	2048, 3072, 4096 bits - 112, 128, 149 bits	Public key - PSP	Key Pair Generation with RSA		
Module-generated RSA Private Key	RSA Private key generated during Asymmetric Key Generation	2048, 3072, 4096 bits - 112, 128, 149 bits	Private key - CSP	Key Pair Generation with RSA		
RSA Public Key	RSA public key used for Signature Verification	2048, 3072, 4096 bits - 112, 128, 149 bits	Public key - PSP			Digital Signature Verification with RSA
RSA Private Key	RSA private key used for Signature Generation	2048, 3072, 4096 bits - 112, 128, 149 bits	Private key - CSP			Digital Signature Generation with RSA
Module-generated ECDSA Public Key	ECDSA 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		
Module-generated ECDSA Private Key	ECDSA 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		
ECDSA Public Key	Public key used for Digital Signature Verification, Public Key Verification	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP			Digital Signature Verification with ECDSA Public Key Verification with ECDSA
ECDSA Private Key	Private key used for Digital Signature Generation, Public Key Verification	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP			Digital Signature Generation with ECDSA Public Key Verification with ECDSA
Module-generated Diffie-Hellman Public Key	Diffie-Hellman public key generated during Diffie-Hellman Key Generation using Safe Primes	MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 2048, 3072, 4096, 6144, 8192 bits - 112-200 bits	Public key - PSP	Key Pair Generation with Safe Primes		TLS Handshake
Module-generated	Diffie-Hellman private key generated during Diffie-Hellman Key	MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-	Private key - CSP	Key Pair Generation		TLS Handshake

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Diffie-Hellman Private Key	Generation using Safe Primes	8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192; 2048, 3072, 4096, 6144, 8192 bits - 112-200 bits		with Safe Primes		
Diffie-Hellman Public Key	Public key used for Shared Secret Computation	MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192; 2048, 3072, 4096, 6144, 8192 bits - 112-200 bits	Public key - PSP			Shared Secret Computation with KAS-FFC-SSC
Diffie-Hellman Private Key	Private key used for Shared Secret Computation	MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192; 2048, 3072, 4096, 6144, 8192 bits - 112-200 bits	Private key - CSP			Shared Secret Computation with KAS-FFC-SSC
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 KAS-ECC-SSC
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 KAS-ECC-SSC
(Diffie-Hellman) Shared Secret	Shared secret computed during Shared Secret Computation with Diffie-Hellman	2048, 3072, 4096, 6144, 8192 bits - 112-200 bits	Shared secret - CSP		Shared Secret Computation with KAS-FFC-SSC	Key Derivation with TLS KDF Key Derivation with KDA HKDF
(EC Diffie-Hellman) Shared Secret	Shared secret computed from EC Diffie-Hellman Key Agreement	P-256, P-384, P-521 - 128-256 bits	Shared secret - CSP		Shared Secret Computation with KAS-FFC-SSC	Key Derivation with TLS KDF Key Derivation with KDA HKDF
PBKDF Password or Passphrase	Password/passphrase used for Key Derivation	8-524288 bits - N/A	Password/passphrase - CSP			Key Derivation with PBKDF

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
PBKDF Derived Key	Key derived from PBKDF password/passphrase during Key Derivation	112-4096 bits - 112-256 bits	Derived key - CSP	Key Derivation with PBKDF		
Entropy Input	Entropy input used to seed the Counter DRBG	256 bits - 256 bits	Entropy - CSP			Random Number Generation with Counter DRBG
Counter DRBG Seed	Counter DRBG seed derived from Entropy Input	256 bits - 256 bits	DRBG seed - CSP	Random Number Generation with Counter DRBG		Random Number Generation with Counter DRBG
Counter DRBG Internal State: V Value, Key	Internal state of Counter DRBG	256 bits - 256 bits	Internal state - CSP	Random Number Generation with Counter DRBG		Random Number Generation with Counter DRBG
TLS Pre-master Secret	TLS Pre-master Secret used for deriving the TLS Master Secret	112-256 bits - 112-256 bits	Pre-master secret - CSP		Shared Secret Computation with KAS-ECC-SSC Shared Secret Computation with KAS-FFC-SSC	TLS Handshake
TLS Master Secret	TLS Master Secret used for deriving the TLS Derived Secret	112-256 bits - 112-256 bits	Master secret - CSP	Key Derivation with TLS KDF Key Derivation with KDA HKDF		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 KDF Key Derivation with KDA HKDF		TLS Handshake
Intermediate Key Generation Value	Intermediate key pair generation value generated during key generation services	224-4096 bits - 112 to 256 bits	Intermediate value - CSP	Key Pair Generation with RSA Key Pair Generation with ECDSA Key Pair Generation with Safe Primes		Key Pair Generation with RSA Key Pair Generation with ECDSA Key Pair Generation with Safe Primes
AES-GCM IV	Initialization Vector used with AES-GCM	96 bits - N/A	Initialization Vector - PSP	TLS Handshake		Authenticated Symmetric Encryption with

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
						AES Authenticated Symmetric Decryption with AES

Table 20: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Module-generated AES Key	API output parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	Counter DRBG Internal State: V Value, Key:Derived From
AES Key	API input parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	
Module-generated HMAC Key	API output parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	Counter DRBG Internal State: V Value, Key:Derived From
HMAC Key	API input parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	
Module-generated RSA Public Key	API output parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	Module-generated RSA Private Key:Paired With Intermediate Key Generation Value:Derived From
Module-generated RSA Private Key	API output parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	Module-generated RSA Public Key:Paired With Intermediate Key Generation Value:Derived From
RSA Public Key	API input parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	RSA Private Key:Paired With
RSA Private Key	API input parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	RSA Public Key:Paired With
Module-generated ECDSA Public Key	API output parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	Module-generated ECDSA Private Key:Paired With Intermediate Key Generation Value:Derived From
Module-generated ECDSA Private Key	API output parameters	RAM:Plaintext	For the duration of the service	Free cipher handle	Module-generated ECDSA Public Key:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
				Remove power from the module	Intermediate Key Generation Value:Derived From
ECDSA Public Key	API input parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	ECDSA Private Key:Paired With Intermediate Key Generation Value:Derived From
ECDSA Private Key	API input parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	ECDSA Public Key:Paired With
Module-generated Diffie-Hellman Public Key	API output parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	Module-generated Diffie-Hellman Private Key:Paired With Intermediate Key Generation Value:Derived From
Module-generated Diffie-Hellman Private Key	API output parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	Module-generated Diffie-Hellman Public Key:Paired With Intermediate Key Generation Value:Derived From
Diffie-Hellman Public Key	API input parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	Diffie-Hellman Private Key:Derived From Intermediate Key Generation Value:Derived From (Diffie-Hellman) Shared Secret:Used With
Diffie-Hellman Private Key	API input parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	Diffie-Hellman Public Key:Paired With Intermediate Key Generation Value:Derived From (Diffie-Hellman) Shared Secret:Used With
Module-generated EC Diffie-Hellman Public Key	API output parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	Module-generated EC Diffie-Hellman Private Key:Paired With Intermediate Key Generation Value:Derived From
Module-generated EC Diffie-Hellman Private Key	API output parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	Module-generated EC Diffie-Hellman Public Key:Paired With Intermediate Key Generation Value:Derived From
EC Diffie-Hellman Public Key	API input parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	EC Diffie-Hellman Private Key:Paired With (EC Diffie-Hellman) Shared Secret:Used With
EC Diffie-Hellman Private Key	API input parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	EC Diffie-Hellman Public Key:Paired With (EC Diffie-Hellman) Shared Secret:Used With
(Diffie-Hellman) Shared Secret	API output parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	Diffie-Hellman Public Key:Used With Diffie-Hellman Private Key:Used With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
(EC Diffie-Hellman) Shared Secret	API output parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	EC Diffie-Hellman Public Key:Used With EC Diffie-Hellman Private Key:Used With
PBKDF Password or Passphrase	API input parameters	RAM:Plaintext	For the duration of the service	Automatic	PBKDF Derived Key:Derived From
PBKDF Derived Key		RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	PBKDF Password or Passphrase:Derived From
Entropy Input		RAM:Plaintext	From generation until Counter DRBG Seed is created	Free cipher handle Remove power from the module	Counter DRBG Seed:Derived From
Counter DRBG Seed		RAM:Plaintext	While the Counter DRBG is being instantiated	Free cipher handle Remove power from the module	Entropy Input:Derived From Counter DRBG Internal State: V Value, Key:Derived From
Counter DRBG Internal State: V Value, Key		RAM:Plaintext	While the Counter DRBG is being instantiated	Free cipher handle Remove power from the module	Counter DRBG Seed:Derived From
TLS Pre-master Secret		RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	TLS Master Secret:Derived From Module-generated Diffie-Hellman Public Key:Used With Module-generated Diffie-Hellman Private Key:Used With Module-generated EC Diffie-Hellman Public Key:Used With Module-generated EC Diffie-Hellman Private Key:Used With
TLS Master Secret		RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	TLS Pre-master Secret:Derived From TLS Derived Secret:Derived From
TLS Derived Secret	API output parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	TLS Master Secret:Derived From
Intermediate Key Generation Value		RAM:Plaintext	Intermediate value	Automatic	Module-generated RSA Public Key:Derived From Module-generated RSA Private Key:Derived From Module-generated Diffie-Hellman Public Key:Derived From Module-generated Diffie-Hellman Private Key:Derived From Module-generated EC Diffie-Hellman Public Key:Derived From Module-generated EC Diffie-

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
					Hellman Private Key:Derived From
AES-GCM IV	API input parameters API output parameters	RAM:Plaintext	For the duration of the service	Free cipher handle Remove power from the module	AES Key:Used With

Table 21: SSP Table 2

9.5 Transitions

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

Beginning in 2031, keys providing 112 bits of security strength shall no longer be used to apply cryptographic protection and are limited to legacy use for processing, as specified in Table 4 of NIST SP 800-57 Part 1 Rev. 5.

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A7464) - arm64	256-bit key	Message Authentication	SW/FW Integrity	Module becomes operational and services are available for use	Integrity test for libgnutls.so.30, libnettle.so.8, libhogweed.so.6
HMAC-SHA2-256 (A7518) - x86	256-bit key	Message Authentication	SW/FW Integrity	Module becomes operational and services are available for use	Integrity test for libgnutls.so.30, libnettle.so.8, libhogweed.so.6

Table 22: Pre-Operational Self-Tests

The module performs the pre-operational self-test and CASTs automatically when the module is loaded into memory. The 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, 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 test and CASTs are completed successfully. After the pre-operational test and the CASTs succeed, the module becomes operational. If the pre-operational test or any of the CASTs fail, an error message is returned, and the module transitions to the Error state.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
ECDSA KeyVer (FIPS186-5)	N/A	KAT	CAST	Module becomes operational	ECDSA Key verification	Test runs at power-on before the integrity test
KDF TLS	N/A	KAT	CAST	Module becomes operational	Industry-based TLS v1.0/1.1 KDF key derivation	Test runs at power-on before the integrity test
AES-XTS Testing Revision 2.0 (A7467) Encrypt	128-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A7464) Encrypt	128 and 256-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A7466) Encrypt	128 and 256-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A7515) Encrypt	128 and 256-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A7516) Encrypt	128 and 256-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC (A7517) Encrypt	128 and 256-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A7518) Encrypt	128 and 256-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CFB8 (A7469) Encrypt	128 and 256-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CFB8 (A7520) Encrypt	128 and 256-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CFB8 (A7521) Encrypt	128-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A7464) Encrypt	256-bit keys, 96-bit IV	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A7466) Encrypt	256-bit keys, 96-bit IV	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A7515) Encrypt	256-bit keys, 96-bit IV	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A7516) Encrypt	256-bit keys, 96-bit IV	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A7517) Encrypt	256-bit keys, 96-bit IV	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A7518) Encrypt	256-bit keys, 96-bit IV	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-XTS Testing Revision 2.0 (A7467) Decrypt	128-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A7464) Decrypt	128-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A7466) Decrypt	128-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A7515) Decrypt	128-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC (A7516) Decrypt	128-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A7517) Decrypt	128-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A7518) Decrypt	128-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CFB8 (A7469) Decrypt	128-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CFB8 (A7520) Decrypt	128-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CFB8 (A7521) Decrypt	128-bit keys	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A7464) Decrypt	256-bit keys, 96-bit IV	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A7466) Decrypt	256-bit keys, 96-bit IV	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A7515) Decrypt	256-bit keys, 96-bit IV	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A7516) Decrypt	256-bit keys, 96-bit IV	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A7517) Decrypt	256-bit keys, 96-bit IV	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A7518) Decrypt	256-bit keys, 96-bit IV	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
KAS-FFC-SSC Sp800-56Ar3 (A7466)	ffdhe2048	PCT	CAST	Module becomes operational	Shared secret computation	Test runs at power-on before the integrity test
KAS-ECC-SSC Sp800-56Ar3 (A7466)	P-256	KAT	CAST	Module becomes operational	Shared secret computation	Test runs at power-on before the integrity test
Counter DRBG (A7466)	AES-128 with prediction resistance	KAT	CAST	Module becomes operational	SP 800-90Ar1 (instantiate, reseed, generate) health test	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
ECDSA SigGen (FIPS186-5) (A7466)	SHA-256; P-256, P-384, P-521	KAT	CAST	Module becomes operational	Digital signature generation	Test runs at power-on before the integrity test
ECDSA SigVer (FIPS186-5) (A7466)	SHA-256; P-256, P-384, P-521	KAT	CAST	Module becomes operational	Digital signature verification	Test runs at power-on before the integrity test
RSA SigGen (FIPS186-5) (A7466)	PKCS#1 v1.5 with SHA-256; 2048-bit key	KAT	CAST	Module becomes operational	Digital signature generation	Test runs at power-on before the integrity test
KDA HKDF Sp800-56Cr1 (A7465)	SHA2-224, SHA2-256; 48-bit, 392-bit input secret	KAT	CAST	Module becomes operational	Shared secret key derivation	Test runs at power-on before the integrity test
SHA3-224 (A7468)	SHA3-224	KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA3-224 (A7519)	SHA3-224	KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA3-256 (A7468)	SHA3-256	KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA3-256 (A7519)	SHA3-256	KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA3-384 (A7468)	SHA3-384	KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA3-384 (A7519)	SHA3-384	KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA3-512 (A7468)	SHA3-512	KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA3-512 (A7519)	SHA3-512	KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
ECDSA KeyGen (FIPS186-5) (A7466)	SHA2-256	PCT	PCT	Successful key pair generation	Signature generation & verification	Key pair generation
RSA KeyGen (FIPS186-5) (A7466)	N/A	PCT	PCT	Successful key pair generation	Signature generation & verification	Key pair generation
Safe Primes Key Generation (A7466)	N/A	PCT	PCT	Successful key pair generation	Signature generation & verification	Key pair generation
HMAC-SHA-1 (A7464)	SHA-1	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA-1 (A7466)	SHA-1	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA-1 (A7518)	SHA-1	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-224 (A7464)	SHA-224	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-224 (A7466)	SHA-224	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-224 (A7518)	SHA-224	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-256 (A7464)	SHA-256	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-256 (A7466)	SHA-256	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-256 (A7518)	SHA-256	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-384 (A7464)	SHA-384	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-384 (A7466)	SHA-384	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-384 (A7518)	SHA-384	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-512 (A7464)	SHA-512	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-512 (A7466)	SHA-512	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-512 (A7518)	SHA-512	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
SHA-1 (A7464)	SHA-1	KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA-1 (A7466)	SHA-1	KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA-1 (A7518)	SHA-1	KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA2-512 (A7464)	SHA2-512	KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA2-512 (A7466)	SHA2-512	KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA2-512 (A7518)	SHA2-512	KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
TLS v1.2 KDF RFC7627 (A7466)	SHA-256; 384-bit input secret	KAT	CAST	Module becomes operational	Industry-based TLS v1.2 KDF key derivation	Test runs at power-on before the integrity test
PBKDF (A7466)	SHA-256; 24 character password; 288-bit salt; Iteration count: 4096	KAT	CAST	Module becomes operational	Password-based key derivation	Test runs at power-on before the integrity test
RSA SigVer (FIPS186-5) (A7466)	PKCS#1 v1.5 with SHA-256; 2048-bit key	KAT	CAST	Module becomes operational	Digital signature verification	Test runs at power-on before the integrity test

Table 23: Conditional Self-Tests

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A7464) - arm64	Message Authentication	SW/FW Integrity	On demand	Manually
HMAC-SHA2-256 (A7518) - x86	Message Authentication	SW/FW Integrity	On demand	Manually

Table 24: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
ECDSA KeyVer (FIPS186-5)	KAT	CAST	On Demand	Manually
KDF TLS	KAT	CAST	On Demand	Manually
AES-XTS Testing Revision 2.0 (A7467) Encrypt	KAT	CAST	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC (A7464) Encrypt	KAT	CAST	On Demand	Manually
AES-CBC (A7466) Encrypt	KAT	CAST	On Demand	Manually
AES-CBC (A7515) Encrypt	KAT	CAST	On Demand	Manually
AES-CBC (A7516) Encrypt	KAT	CAST	On Demand	Manually
AES-CBC (A7517) Encrypt	KAT	CAST	On Demand	Manually
AES-CBC (A7518) Encrypt	KAT	CAST	On Demand	Manually
AES-CFB8 (A7469) Encrypt	KAT	CAST	On Demand	Manually
AES-CFB8 (A7520) Encrypt	KAT	CAST	On Demand	Manually
AES-CFB8 (A7521) Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A7464) Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A7466) Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A7515) Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A7516) Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A7517) Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A7518) Encrypt	KAT	CAST	On Demand	Manually
AES-XTS Testing Revision 2.0 (A7467) Decrypt	KAT	CAST	On Demand	Manually
AES-CBC (A7464) Decrypt	KAT	CAST	On Demand	Manually
AES-CBC (A7466) Decrypt	KAT	CAST	On Demand	Manually
AES-CBC (A7515) Decrypt	KAT	CAST	On Demand	Manually
AES-CBC (A7516) Decrypt	KAT	CAST	On Demand	Manually
AES-CBC (A7517) Decrypt	KAT	CAST	On Demand	Manually
AES-CBC (A7518) Decrypt	KAT	CAST	On Demand	Manually
AES-CFB8 (A7469) Decrypt	KAT	CAST	On Demand	Manually
AES-CFB8 (A7520) Decrypt	KAT	CAST	On Demand	Manually
AES-CFB8 (A7521) Decrypt	KAT	CAST	On Demand	Manually
AES-GCM (A7464) Decrypt	KAT	CAST	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-GCM (A7466) Decrypt	KAT	CAST	On Demand	Manually
AES-GCM (A7515) Decrypt	KAT	CAST	On Demand	Manually
AES-GCM (A7516) Decrypt	KAT	CAST	On Demand	Manually
AES-GCM (A7517) Decrypt	KAT	CAST	On Demand	Manually
AES-GCM (A7518) Decrypt	KAT	CAST	On Demand	Manually
KAS-FFC-SSC Sp800-56Ar3 (A7466)	PCT	CAST	On Demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A7466)	KAT	CAST	On Demand	Manually
Counter DRBG (A7466)	KAT	CAST	On Demand	Manually
ECDSA SigGen (FIPS186-5) (A7466)	KAT	CAST	On Demand	Manually
ECDSA SigVer (FIPS186-5) (A7466)	KAT	CAST	On Demand	Manually
RSA SigGen (FIPS186-5) (A7466)	KAT	CAST	On Demand	Manually
KDA HKDF Sp800-56Cr1 (A7465)	KAT	CAST	On Demand	Manually
SHA3-224 (A7468)	KAT	CAST	On Demand	Manually
SHA3-224 (A7519)	KAT	CAST	On Demand	Manually
SHA3-256 (A7468)	KAT	CAST	On Demand	Manually
SHA3-256 (A7519)	KAT	CAST	On Demand	Manually
SHA3-384 (A7468)	KAT	CAST	On Demand	Manually
SHA3-384 (A7519)	KAT	CAST	On Demand	Manually
SHA3-512 (A7468)	KAT	CAST	On Demand	Manually
SHA3-512 (A7519)	KAT	CAST	On Demand	Manually
ECDSA KeyGen (FIPS186-5) (A7466)	PCT	PCT	On Demand	Manually
RSA KeyGen (FIPS186-5) (A7466)	PCT	PCT	On Demand	Manually
Safe Primes Key Generation (A7466)	PCT	PCT	On Demand	Manually
HMAC-SHA-1 (A7464)	KAT	CAST	On Demand	Manually
HMAC-SHA-1 (A7466)	KAT	CAST	On Demand	Manually
HMAC-SHA-1 (A7518)	KAT	CAST	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-224 (A7464)	KAT	CAST	On Demand	Manually
HMAC-SHA2-224 (A7466)	KAT	CAST	On Demand	Manually
HMAC-SHA2-224 (A7518)	KAT	CAST	On Demand	Manually
HMAC-SHA2-256 (A7464)	KAT	CAST	On Demand	Manually
HMAC-SHA2-256 (A7466)	KAT	CAST	On Demand	Manually
HMAC-SHA2-256 (A7518)	KAT	CAST	On Demand	Manually
HMAC-SHA2-384 (A7464)	KAT	CAST	On Demand	Manually
HMAC-SHA2-384 (A7466)	KAT	CAST	On Demand	Manually
HMAC-SHA2-384 (A7518)	KAT	CAST	On Demand	Manually
HMAC-SHA2-512 (A7464)	KAT	CAST	On Demand	Manually
HMAC-SHA2-512 (A7466)	KAT	CAST	On Demand	Manually
HMAC-SHA2-512 (A7518)	KAT	CAST	On Demand	Manually
SHA-1 (A7464)	KAT	CAST	On Demand	Manually
SHA-1 (A7466)	KAT	CAST	On Demand	Manually
SHA-1 (A7518)	KAT	CAST	On Demand	Manually
SHA2-512 (A7464)	KAT	CAST	On Demand	Manually
SHA2-512 (A7466)	KAT	CAST	On Demand	Manually
SHA2-512 (A7518)	KAT	CAST	On Demand	Manually
TLS v1.2 KDF RFC7627 (A7466)	KAT	CAST	On Demand	Manually
PBKDF (A7466)	KAT	CAST	On Demand	Manually
RSA SigVer (FIPS186-5) (A7466)	KAT	CAST	On Demand	Manually

Table 25: Conditional Periodic Information

10.4 Error States

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

Table 26: Error States

In the Error state, the output interface is inhibited, and the module accepts no more inputs or requests (as the module is no longer running). The calling application can obtain the module state by calling the gnutls_fips140_get_operation_state() API function. A complete list of the error codes can be found in Appendix C “Error Codes and Descriptions” in the gnutls.pdf provided with the module's code.

10.5 Operator Initiation of Self-Tests

The module provides the Self-Test service to perform self-tests on demand which includes the pre-operational test (i.e., integrity test) and 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. The PCTs can be invoked on demand by requesting the Asymmetric Key Generation service.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

Before the RPM packages are installed, the Amazon Linux 2023 system must operate in the FIPS validated configuration. To achieve this, the Crypto Officer must ensure that the crypto-policies utilities are installed and up to date by executing sudo dnf -y install crypto-policies crypto-policies-scripts. Then, the Crypto Officer must execute the sudo fips-mode-setup --enable command, then, restart the system by executing sudo reboot.

The Crypto Officer must verify the Amazon Linux 2023 system operates in the FIPS validated configuration by executing the sudo fips-mode-setup --check command, which should output "FIPS mode is enabled."

Once the operating environment is booted in FIPS validated configuration, the Crypto Officer can install the Amazon packages containing the module using the Yellowdog Updater Modified (yum) with the following commands:

```
$ sudo yum install gnutls-3.8.3-6.amzn2023.0.1

$ sudo yum install gmp-6.2.1-2.amzn2023.0.2

$ sudo yum install nettle-3.10.1-1.amzn2023.0.1
```

Note: libhogweed is provided by nettle-3.10.1-1.amzn2023.0.1.

All the Amazon packages are associated with hashes for integrity check. The integrity of the Amazon package is automatically verified by the packing tool during the installation of the module. The Crypto Officer shall not install the package if the integrity fails.

11.2 Administrator Guidance

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

The Crypto Officer shall follow this Security Policy to configure the operational environment and install the module to be operated in the approved mode.

The module cannot use the following environment variables:

- GNUTLS_NO_EXPLICIT_INIT
- GNUTLS_SKIP_FIPS_INTEGRITY_CHECKS

The module can only be used with the cryptographic algorithms provided. Therefore, the following API functions are forbidden in the approved mode of operation:

- gnutls_crypto_register_cipher
- gnutls_crypto_register_aead_cipher
- gnutls_crypto_register_mac
- gnutls_crypto_register_digest
- gnutls_privkey_import_ext4

The “Show Module Name and Version” service returns the value

“fips-module-name: Amazon Linux 2023 gnutls”

“fips-module-version: 3.8.3-98ce29ed6c83f8a2”

11.3 Non-Administrator Guidance

There is no non-administrator guidance.

11.4 End of Life

As the module does not persistently store SSPs, secure sanitization of the module consists of unloading the module. This will zeroize all SSPs in volatile memory. Then, if desired, the gnutls-3.8.3-6.amzn2023.0.1, gmp-6.2.1-2.amzn2023.0.2, nettle-3.10.1-1.amzn2023.0.1 RPM packages can be uninstalled from the Amazon Linux 2023 systems.

12 Mitigation of Other Attacks

The module does not mitigate other attacks.

Appendix A. TLS Cipher Suites

The module supports the following cipher suites for the TLS protocol version 1.0, 1.1, 1.2 and 1.3, compliant with section 3.3.1 of SP 800-52rev2. 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
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

Cipher Suite	ID	Reference
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
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
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
GMAC	Galois Counter Mode Message Authentication Code
HMAC	Hash Message Authentication Code
KAS	Key Agreement Scheme
KAT	Known Answer Test
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
OFB	Output Feedback
PAA	Processor Algorithm Acceleration
PAI	Processor Algorithm Implementation
PBKDF2	Password-based Key Derivation Function v2
PKCS	Public-Key Cryptography Standards

PCT	Pairwise Consistency Test
PR	Prediction Resistance
RNG	Random Number Generator
RSA	Rivest, Shamir, Adleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard

Appendix C. References

FIPS140-3	**FIPS PUB 140-3 - Security Requirements For Cryptographic Modules** March 2019 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf
FIPS140-3 IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program [09-02-2025]** September 2025 https://csrc.nist.gov/csrc/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/FIPS%20140-3%20IG.pdf
FIPS180-4	**Secure Hash Standard (SHS)** August 2015 http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS186-5	**Digital Signature Standard (DSS)** February 2023 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf
FIPS197	**Advanced Encryption Standard** November 2001 http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS198-1	**The Keyed Hash Message Authentication Code (HMAC)** July 2008 http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
FIPS202	**SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions** August 2015 http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
PKCS#1	**Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1** February 2003 http://www.ietf.org/rfc/rfc3447.txt
SP800-38A	**NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
SP800-38B	**NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38b.pdf

SP800-38C	**NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality** May 2004 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf
SP800-38D	**NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38d.pdf
SP800-38E	**NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices** January 2010 http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf
SP800-52r2	**NIST Special Publication 800-52 Revision 2 - Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations** August 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r2.pdf
SP800-56Ar3	**NIST Special Publication 800-56A Revision 3 - Recommendation for Pair Wise Key Establishment Schemes Using Discrete Logarithm Cryptography** April 2018 https://doi.org/10.6028/NIST.SP.800-56Ar3
SP800-56Cr2	**Recommendation for Key Derivation through Extraction-then-Expansion** August 2020 https://doi.org/10.6028/NIST.SP.800-56Cr2
SP800-57r5	**NIST Special Publication 800-57 Part 1 Revision 5 - Recommendation for Key Management Part 1: General** May 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf
SP800-90Ar1	**NIST Special Publication 800-90A - Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 http://dx.doi.org/10.6028/NIST.SP.800-90Ar1
SP800-90B	**NIST Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://doi.org/10.6028/NIST.SP.800-90B

SP800-131Ar2	**NIST Special Publication 800-131 Revision 2 - Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths** March 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf
SP800-132	**NIST Special Publication 800-132 - Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** December 2010 http://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf
SP800-133r2	**NIST Special Publication 800-133 - Recommendation for Cryptographic Key Generation** June 2020 https://doi.org/10.6028/NIST.SP.800-133r2
SP800-135r1	**NIST Special Publication 800-135 Revision 1 - Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf
SP800-140B	**NIST Special Publication 800-140B - CMVP Security Policy Requirements** March 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140B.pdf