Cloudlinux Inc., TuxCare division

GnuTLS cryptography module for AlmaLinux 9

FIPS 140-3 Non-Proprietary Security Policy

Prepared by:

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

Document version: 1.0
Last update: 2025-05-06

Table of Contents

1 - General 6
1.1 Overview 6
1.2 Security Levels 6
1.3 Additional Information 7
2 Cryptographic Module Specification 8
2.1 Description 8
2.2 Tested and Vendor Affirmed Module Version and Identification 9
2.3 Excluded Components 10
2.4 Modes of Operation 10
2.5 Algorithms 11
2.6 Security Function Implementations 19
2.7 Algorithm Specific Information 25
2.8 RBG and Entropy 27
2.9 Key Generation 28
2.10 Key Establishment 28
2.11 Industry Protocols 29
3 Cryptographic Module Interfaces 30
3.1 Ports and Interfaces 30
4 Roles, Services, and Authentication 31
4.1 Authentication Methods 31
4.2 Roles 31
4.3 Approved Services 31
4.4 Non-Approved Services 44
4.5 External Software/Firmware Loaded 46
5 Software/Firmware Security 47
5.1 Integrity Techniques 47
5.2 Initiate on Demand 47
6 Operational Environment 48
6.1 Operational Environment Type and Requirements 48
6.2 Configuration Settings and Restrictions 48

7 Physical Security 49
8 Non-Invasive Security 50
9 Sensitive Security Parameters Management 51
9.1 Storage Areas 51
9.2 SSP Input-Output Methods 51
9.3 SSP Zeroization Methods 52
9.4 SSPs 53
9.5 Transitions 62
10 Self-Tests 63
10.1 Pre-Operational Self-Tests 63
10.2 Conditional Self-Tests 63
10.3 Periodic Self-Test Information 73
10.4 Error States 76
10.5 Operator Initiation of Self-Tests 77
11 Life-Cycle Assurance 78
11.1 Installation, Initialization, and Startup Procedures 78
11.2 Administrator Guidance 78
11.3 Non-Administrator Guidance 78
11.4 End of Life 78
12 Mitigation of Other Attacks 79
12.1 Attack List 79
12.2 Mitigation Effectiveness 79
Appendix A. TLS Cipher Suites 80
Appendix B. Glossary and Abbreviations 82
Appendix C. References 83

List of Tables

Table 1: Security Levels....6
Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)....9
Table 3: Tested Operational Environments - Software, Firmware, Hybrid....10
Table 4: Modes List and Description....10
Table 5: Approved Algorithms....16
Table 6: Vendor-Affirmed Algorithms....16
Table 7: Non-Approved, Not Allowed Algorithms....18
Table 8: Security Function Implementations....25
Table 9: Entropy Certificates....27
Table 10: Entropy Sources....27
Table 11: Ports and Interfaces....30
Table 12: Roles....31
Table 13: Approved Services....43
Table 14: Non-Approved Services....46
Table 15: Storage Areas....51
Table 16: SSP Input-Output Methods....51
Table 17: SSP Zeroization Methods....52
Table 18: SSP Table 1....58
Table 19: SSP Table 2....62
Table 20: Pre-Operational Self-Tests....63
Table 21: Conditional Self-Tests....72
Table 22: Pre-Operational Periodic Information....73
Table 23: Conditional Periodic Information....76
Table 24: Error States....77

List of Figures

Figure 1: Block Diagram....9

1 - General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for version 3.7.6-396796fe0a32b434 of GnuTLS cryptography module for AlmaLinux 9. It has a one-to-one mapping to the [SP 800-140Br1] starting with section B.2.1 named “General” that maps to section 1 in this document and ending with section B.2.12 named “Mitigation of other attacks” that maps to section 12 in this document. 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	1
	Overall Level	1

Table 1: Security Levels

1.3 Additional Information

This Security Policy describes the features and design of the module named GnuTLS cryptography module for AlmaLinux 9 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.

In preparing the Security Policy document, the laboratory formatted the vendor-supplied documentation for consolidation without altering the technical statements therein contained. The further refining of the Security Policy document was conducted iteratively throughout the conformance testing, wherein the Security Policy was submitted to the vendor, who would then edit, modify, and add technical contents. The vendor would also supply additional documentation, which the laboratory formatted into the existing Security Policy, and resubmitted to the vendor for their final editing.

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The GnuTLS cryptography module for AlmaLinux 9 (hereafter referred to as "the module") is a software library. The module is an open-source, general-purpose set of libraries designed to support cross-platform development of security-enabled client and server applications. The module is a multiple-chip standalone cryptographic module.

Module Type: Software

Module Embodiment: MultiChipStand

Module Characteristics [O]:

Cryptographic Boundary:

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

The module is implemented as a shared library. The cryptographic module boundary consists of the following components:

- libgnutls
- libnettle
- libhogweed
- libgmp – statically linked
- gnutls hmac file

Tested Operational Environment's Physical Perimeter (TOEPP):

The software block diagram below shows the module, its interfaces with the operational environment and the delimitation of its cryptographic boundary.

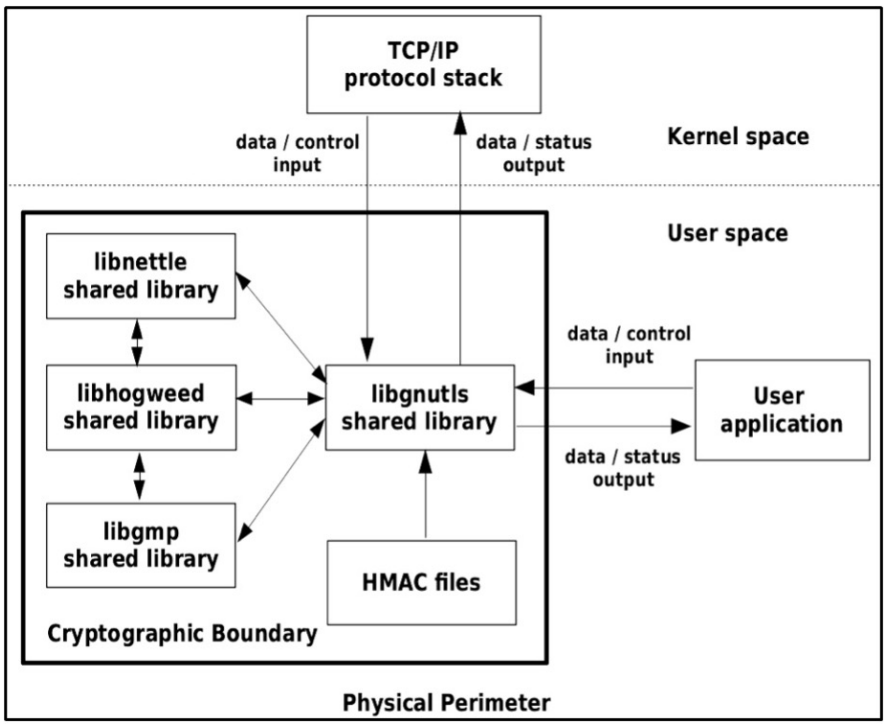

Figure 1: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

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

Tested Module Identification – Hardware:

N/A for this module.

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
/usr/lib64/libgnutls.so.30, /usr/lib64/libnettle.so.8, /usr/lib64/libhogweed.so.6, /usr/lib64/.libgnutls.so.30.hmac. Note: libgmp is statically linked to libgnutls	3.7.6-396796fe0a32b434	N/A	HMAC SHA2-256

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

Tested Module Identification – Hybrid Disjoint Hardware:

N/A for this module.

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
AlmaLinux 9.2	Amazon Web Services (AWS) m5.metal	Intel Xeon Platinum 8259CL	Yes	N/A	3.7.6-396796fe0a32b434
AlmaLinux 9.2	Amazon Web Services (AWS) m5.metal	Intel Xeon Platinum 8259CL	No	N/A	3.7.6-396796fe0a32b434

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

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

N/A for this module.

CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

2.3 Excluded Components

The module does not claim any excluded components.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved mode	Automatically entered when the module starts up successfully, after passing all the pre-operational and conditional cryptographic algorithms self-tests.	Approved	Equivalent to the indicator of the requested service.
Non-approved mode	Automatically entered whenever a non-approved service is requested.	Non-Approved	Equivalent to the indicator of the requested service.

Table 4: Modes List and Description

Mode Change Instructions and Status:

When the module starts up successfully, after passing all the pre-operational and conditional cryptographic algorithms self-tests (CASTs), the module is operating in the approved mode of operation by default and can only be transitioned into the non-Approved mode by calling one of the non-Approved services listed in Table 15. Please see section 4 for the details on service indicator provided by the module that identifies when an approved service is called.

Degraded Mode Description:

The module does not implement a degraded mode of operation.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A5114	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A5115	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A5116	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A5117	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A5122	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A5114	Key Length - 128, 256	SP 800-38C
AES-CFB8	A5119	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A5120	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A5125	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
AES-CMAC	A5114	Direction - Generation, Verification Key Length - 128, 256	SP 800-38B
AES-CMAC	A5117	Direction - Generation, Verification Key Length - 128, 256	SP 800-38B
AES-CMAC	A5122	Direction - Generation, Verification Key Length - 128, 256	SP 800-38B
AES-ECB	A5126	Direction - Encrypt Key Length - 256	SP 800-38A
AES-GCM	A5114	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-GCM	A5115	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-GCM	A5116	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-GCM	A5117	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-GCM	A5122	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-GMAC	A5122	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D

Algorithm	CAVP Cert	Properties	Reference
AES-XTS Testing Revision 2.0	A5123	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E
Counter DRBG	A5122	Prediction Resistance - No Mode - AES-256 Derivation Function Enabled - No	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-5)	A5122	Curve - P-256, P-384, P-521 Secret Generation Mode - testing candidates	FIPS 186-5
ECDSA KeyVer (FIPS186-5)	A5122	Curve - P-256, P-384, P-521	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A5122	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)	A5122	Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-5
HMAC-SHA-1	A5117	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA-1	A5122	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A5117	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A5122	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A5117	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A5122	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A5117	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1

Algorithm	CAVP Cert	Properties	Reference
HMAC-SHA2-384	A5122	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A5117	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A5122	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A5122	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	A5122	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	A5121	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-65336 Increment 8 HMAC Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	SP 800-56C Rev. 2
PBKDF	A5122	Iteration Count - Iteration Count: 1000-10000 Increment 1 Password Length - Password Length: 8-128 Increment 1	SP 800-132
RSA KeyGen (FIPS186-5)	A5122	Key Generation Mode - provable Hash Algorithm - SHA2-384 Modulo - 2048, 3072, 4096 Private Key Format - standard	FIPS 186-5
RSA SigGen (FIPS186-5)	A5122	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
RSA SigVer (FIPS186-5)	A5122	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5

Algorithm	CAVP Cert	Properties	Reference
Safe Primes Key Generation	A5122	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	A5117	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA-1	A5122	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-224	A5117	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-224	A5122	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A5117	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A5122	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-384	A5117	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-384	A5122	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A5117	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A5122	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA3-224	A5118	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-224	A5124	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-256	A5118	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202

Algorithm	CAVP Cert	Properties	Reference
SHA3-256	A5124	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-384	A5118	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-384	A5124	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-512	A5118	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-512	A5124	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
TLS v1.2 KDF RFC7627 (CVL)	A5122	Hash Algorithm - SHA2-256, SHA2-384	SP 800-135 Rev. 1

Table 5: Approved Algorithms

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

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
Key Pair Generation with RSA	RSA:2048, 3072, 4096-bit keys with 112-149 bits key strength	GnuTLS cryptography module for AlmaLinux 9 (Generic C)	SP 800-133r2 section 5
Key Pair Generation with ECDSA	ECDSA:P-256, P-384, P-521 elliptic curves with 128-256 bits key strength	GnuTLS cryptography module for AlmaLinux 9 (Generic C)	SP 800-133r2 section 5
Key Pair Generation with Safe Primes	Safe Primes:ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Keys:2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits key strength	GnuTLS cryptography module for AlmaLinux 9 (Generic C)	SP 800-133r2 section 5

Table 6: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

The module does not implement non-approved algorithms that are allowed in the approved mode of operation.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

The module does not implement any non-approved, allowed algorithm in the approved mode of operation with no security claimed.

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
Blowfish	Symmetric encryption; Symmetric decryption
Camellia	Symmetric encryption; Symmetric decryption
CAST	Symmetric encryption; Symmetric decryption
ChaCha20	Symmetric encryption; Symmetric decryption
Chacha20, Poly1305 and AES-GCM	Authenticated encryption; Authenticated decryption
DES	Symmetric encryption; Symmetric decryption
Diffie-Hellman with keys generated with domain parameters other than safe primes	Key agreement; Shared secret computation
DRBG when key length is less than 112 bits	Symmetric key generation
DSA	Key generation; Domain parameter generation; Digital signature generation; Digital signature verification
ECDSA with curves not listed in Table "Approved Algorithms"	Key generation; Public key verification; Digital signature generation; Digital signature verificatio
EC Diffie-Hellman with curves not listed in Table "Approved Algorithms"	Key agreement; Shared secret computation
GOST	Symmetric encryption; Symmetric decryption; Message digest
HMAC with keys smaller than 112-bit	Message authentication code (MAC)

Name	Use and Function
HMAC with GOST	Message authentication code (MAC)
MD2, MD4, MD5	Message digest; Message authentication code (MAC)
Non-supported cipher suites (not listed in Appendix A)	Transport Layer Security (TLS) Network Protocol
PBKDF with non-approved message digest algorithms	Key derivation
RC2, RC4	Symmetric encryption; Symmetric decryption
RMD160	Message digest; Message authentication code (MAC)
RSA with keys smaller than 2048 bits or greater than 4096 bits.	Key generation; Digital signature generation
RSA with keys smaller than 1024 bits or greater than 4096 bits.	Digital signature verification
RSA encryption and decryption with any key sizes.	Key encapsulation; Key unencapsulation
Salsa20	Symmetric encryption; Symmetric decryption
SM3	Hashing
Serpent	Symmetric encryption; Symmetric decryption
SHA-1	Digital signature generation; Digital Signature Verification
STREEBOG	Message digest; Message authentication code (MAC)
Triple-DES	Symmetric encryption; Symmetric decryption
Twofish	Symmetric encryption; Symmetric decryption
UMAC	Message authentication code (MAC)
Yarrow	Random number generation

Table 7: Non-Approved, Not Allowed Algorithms

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

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Symmetric Encryption with AES	BC-UnAuth	Encryption using AES	AES-XTS mode keys:128, 256 bits with 128, 256 of key strength Other modes keys:128, 192, 256 bits with 128-256 of key strength	AES-CBC AES-CBC AES-CBC AES-CBC AES-CFB8 AES-CFB8 AES-CBC AES-CFB8 AES-XTS Testing Revision 2.0
Symmetric Decryption with AES	BC-UnAuth	Decryption using AES	AES-XTS mode keys:128, 256 bits keys with 128, 256 of key strength Other modes keys:128, 192, 256 bits keys with 128-256 of key strength	AES-CBC AES-CBC AES-CBC AES-CBC AES-CFB8 AES-CFB8 AES-CBC AES-CFB8 AES-XTS Testing Revision 2.0
Authenticated Symmetric Encryption with AES	BC-Auth	Authenticated encryption using AES	Keys:128, 256 bits with 128, 256 bits key strength	AES-CCM
Authenticated Symmetric Decryption with AES	BC-Auth	Authenticated decryption using AES	Keys:128, 256 bits with 128, 256 bits key strength	AES-CCM
Authenticated Symmetric Encryption (in the context of the TLS 1.2/1.3 protocol) with AES-GCM	BC-Auth	Authenticated encryption using AES-GCM (as part of TLS protocol)	Keys:128, 256 bits with 128, 256 bits key strength	AES-GCM AES-GCM AES-GCM AES-GCM AES-GCM

Name	Type	Description	Properties	Algorithms
Authenticated Symmetric Decryption (in the context of the TLS 1.2/1.3 protocol) with AES-GCM	BC-Auth	Authenticated decryption using AES-GCM (as part of TLS protocol)	Keys:128, 256 bits with 128, 256 bits key strength	AES-GCM AES-GCM AES-GCM AES-GCM AES-GCM
Message Digest with SHA	SHA	Message digest using SHA		SHA-1 SHA2-224 SHA2-384 SHA2-512 SHA3-224 SHA3-256 SHA3-384 SHA3-512 SHA-1 SHA2-224 SHA2-256 SHA2-384 SHA2-512 SHA3-224 SHA3-256 SHA3-384 SHA3-512 SHA2-256
Random Number Generation with CTR_DRBG	DRBG	Random number generation using CTR_DRBG	Keys:AES-256 bits with 256 bits key strength AES-256:without DF, without PR	Counter DRBG AES-ECB
Message Authentication Code (MAC) with HMAC	MAC	Message authentication generation using HMAC	Hash Algorithm:SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512 HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512

Name	Type	Description	Properties	Algorithms
Message Authentication Code (MAC) with AES	MAC	Message authentication generation using AES CMAC/GMAC	Keys:128 or 256 bits with 128 or 256 bits of strength	AES-CMAC AES-CMAC AES-CMAC AES-GMAC
Key Pair Generation with RSA	CKG	Key Generation using RSA	Keys:2048, 3072, 4096 bits with 112-149 bits of strength Hash Algorithm:SHA-384	RSA KeyGen (FIPS186-5)
Digital Signature Generation with RSA	DigSig-SigGen	Digital signature generation using RSA	Keys:2048, 3072, 4096 bits with 112, 128, 149 bits of strength PKCS#1v1.5:SHA-224, SHA-256, SHA-384, SHA-512 PSS:SHA-256, SHA-384, SHA-512	RSA SigGen (FIPS186-5)
Digital Signature Verification with RSA	DigSig-SigVer	Signature Verification with RSA	Keys:2048, 3072, 4096 bits with 112-149 bits of strength PKCS#1v1.5:SHA-224, SHA-256, SHA-384, SHA-512 PSS:SHA-256, SHA-384, SHA-512	RSA SigVer (FIPS186-5)
Digital Signature Generation with ECDSA	DigSig-SigGen	Digital signature generation using ECDSA	Curves:P-256, P-384, P-521 with 128-256 bits of key strength Hash Algorithm:SHA-224, SHA-256, SHA-384, SHA-512	ECDSA SigGen (FIPS186-5)
Digital Signature Verification with ECDSA	DigSig-SigVer	Signature verification using ECDSA	Curves:P-256, P-384, P-521 with 128-256 bits of	ECDSA SigVer (FIPS186-5)

Name	Type	Description	Properties	Algorithms
			strength Hash Algorithm:SHA2-224, SHA2-256, SHA2-384, SHA2-512	
Public Key Verification with ECDSA	AsymKeyPair-KeyVer	Public key verification using ECDSA	Curves:P-256, P-384, P-521 elliptic curves with 128-256 bits key strength Compliance:B.4.2 Testing Candidates	ECDSA KeyVer (FIPS186-5)
Key Pair Generation with ECDSA	CKG	Generate ECDSA key pairs	Curves:P-256, P-384, P-521 elliptic curves with 128-256 bits key strength	ECDSA KeyGen (FIPS186-5)
Shared Secret Computation with EC Diffie-Hellman	KAS-SSC	Shared secret computation per SP 800-56ARev3	Curves:P-256, P-384, P-521 elliptic curves keys with 128-256 bits key strength	KAS-ECC-SSC Sp800-56Ar3
Shared Secret Computation with Diffie-Hellman	KAS-SSC	Shared secret computation per SP 800-56ARev3	Domain Parameter Generation Methods:ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Keys:2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits key strength	KAS-FFC-SSC Sp800-56Ar3

Name	Type	Description	Properties	Algorithms
Key Derivation with PBKDF	PBKDF	Key derivation using PBKDF	PBKDF Derived key:112 to 256 bits of key strength HMAC Algorithm:SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	PBKDF
Key Derivation with TLS 1.2 KDF	KAS-135KDF	Key derivation using TLS KDF	TLS Derived Secret:112-256 with 112-256 bits of key strength Hash Algorithm:SHA2-256, SHA2-384	TLS v1.2 KDF RFC7627
Key Derivation (as part of TLSv1.3) with KDA HKDF	KAS-56CKDF	Key derivation using KDA HKDF	HKDF Derived Key:112-256 bits with 112-256 bits key strength HMAC Algorithm:SHA2-224, SHA2-256, SHA2-384, SHA2-512	KDA HKDF Sp800-56Cr1
Key Pair Generation with Safe Primes	CKG	Key Pair Generation with Safe Primes	Safe Prime Groups::ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Keys:2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits key strength	Safe Primes Key Generation

Name	Type	Description	Properties	Algorithms
			Compliance:SP800-56Arev3	
Key Wrapping	KTS-Wrap	Key Wrapping (as part of the cipher suite in the TLS protocol)	Keys:AES-CCM 128, 256-bit keys with 128 or 256 bits of key strength; AES-GCM: 128, 256-bit keys with 128 or 256 bits of key strength; AES-CBC and HMAC: 128, 256-bit keys with 128 or 256 bits of key strength	AES-CCM AES-GCM AES-GCM AES-GCM AES-GCM AES-CBC AES-CBC AES-CBC AES-CBC HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512 AES-CBC AES-GCM HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512
Key Unwrapping	KTS-Wrap	Key Unwrapping (as part of the cipher suite in the TLS protocol)	Keys:AES-CCM 128, 256-bit keys with 128 or 256 bits of key strength; AES-GCM: 128, 256-bit keys with 128 or 256 bits of key strength; AES-CBC and HMAC: 128, 256-bit keys with 128 or 256 bits of key strength	AES-CCM AES-CBC AES-GCM AES-GCM AES-CBC AES-CBC AES-GCM AES-CBC AES-GCM HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512 AES-CBC AES-GCM HMAC-SHA-1 HMAC-SHA2-224

Name	Type	Description	Properties	Algorithms
				HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512
TLS Handshake	KAS-Full	Key Agreement	Curves:P-256, P-384, P-521 elliptic curves with 128, 192, 256 bits of strength Keys:MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192; 2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of key strength Compliance:IG D.F scenario 2(2)	KAS-ECC-SSC Sp800-56Ar3 KAS-FFC-SSC Sp800-56Ar3 KDA HKDF Sp800-56Cr1

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

AES XTS

The AES algorithm in XTS mode can be only used for the cryptographic protection of data on storage devices, as specified in [SP800-38E]. The length of a single data unit encrypted with the XTS-AES shall not exceed 220 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.

Note: AES-XTS shall be used with 128 and 256-bit keys only. AES-XTS with 192-bit keys is not an Approved service.

AES-GCM IV

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

For TLS 1.2, the module offers the AES GCM implementation and uses the context of Scenario 1 of FIPS 140-3 IG C.H. The module is compliant with SP 800-52r2 Section 3.3.1 and the mechanism for IV generation is compliant with RFC 5288 and 8446.

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

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

Finally, for TLS 1.3, the AES GCM implementation uses the context of Scenario 5 of FIPS 140-3 IG C.H. The protocol that provides this compliance is TLS 1.3, defined in RFC8446 of August 2018, using the cipher-suites that explicitly select AES GCM as the encryption/decryption cipher (Appendix B.4 of RFC8446). The module supports acceptable AES GCM cipher suites from Section 3.3.1 of SP800-52r2. TLS 1.3 employs separate 64-bit sequence numbers, one for protocol records that are received, and one for protocol records that are sent to a peer. These sequence numbers are set at zero at the beginning of a TLS 1.3 connection and each time when the AES-GCM key is changed. After reading or writing a record, the respective sequence number is incremented by one. The protocol specification determines that the sequence number should not wrap, and if this condition is observed, then the protocol implementation must either trigger a re-key of the session (i.e., a new key for AES-GCM), or terminate the connection.

Key Derivation using SP 800-132 PBKDF

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

In accordance with [SP800-132], the following requirements shall be met.

- Derived keys shall only be used in storage applications. The Master Key (MK) shall not be used for other purposes. The length of the MK or DPK shall be of 112 bits or more (this is verified by the module to determine the service is approved).
- A portion of the salt, with a length of at least 128 bits (this is verified by the module to determine the service is approved), shall be generated randomly using the SP800-90Arev1 DRBG,
- The iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users. The minimum value shall be 1000 (this is verified by the module to determine the service is approved).
- Passwords or passphrases, used as an input for the PBKDF, shall not be used as cryptographic keys.
- The minimum length of the password or passphrase accepted by the module is 14 characters. The probability of guessing the value, assuming a worst-case scenario of all digits, is estimated to be at most 10^{-14}.

The calling application shall also observe the rest of the requirements and recommendations specified in [SP800-132].

Compliance to SP 800-56ARev3 assurances

In order to meet the required assurances listed in section 5.6 of SP 800-56ARev3, 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 "Key pair generation" service for generating DH/ECDH ephemeral keys. This meets the assurances required by key pair owner defined in the section 5.6.2.1 of SP 800-56ARev3.
2. As part of the module's shared secret computation (SSC) service, the module internally performs the public key validation on the peer's public key passed in as input to the SSC function. This meets the public key validity assurance required by the sections 5.6.2.2.1/5.6.2.2.2 of SP 800-56ARev3.

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

2.8 RBG and Entropy

Cert Number	Vendor Name
E127	Cloudlinux Inc., TuxCare division

Table 9: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Userspace CPU Time Jitter RNG Entropy Source Version 3.4.0	Non-Physical	AlmaLinux 9.2 on Amazon Web Services (AWS) m5.metal on Intel Xeon Platinum 8259CL; AlmaLinux 9.2 on Amazon Web Services (AWS) a1.metal on AWS Graviton	64 bits	64 bits	SHA3-256 (Cert. A4026), HMAC-SHA2-512-DRBG (Cert. A4025)

Table 10: Entropy Sources

The module employs a Deterministic Random Bit Generator (DRBG) based on [SP800-90ARev1] for the generation of random value used in asymmetric keys, and for providing a RNG service to calling applications. The approved DRBG provided by the module is the CTR_DRBG with AES-256. The DRBG does not employ prediction resistance or a derivation function. The module uses an SP800-90B-compliant Entropy Source specified in the table above to seed the DRBG.

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

2.9 Key Generation

In accordance with FIPS 140-3 IG D.H, the cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys according to section 5.1 and 5.2 of [SP800-133rev2].

- For generating RSA and ECDSA keys, the module implements asymmetric cryptographic key generation (CKG) services compliant with [FIPS186-5].
- The public and private keys used in the EC Diffie-Hellman key agreement schemes are generated internally by the module using the ECDSA key generation method compliant with [FIPS186-5] and [SP800-56Arev3].
- The public and private keys used in the Diffie-Hellman key agreement scheme are also compliant with [SP800-56Arev3]. The module generates keys using safe primes defined in RFC7919 and RFC3526, as described in the next section.

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

2.10 Key Establishment

The module provides Diffie-Hellman and EC Diffie-Hellman shared secret computation compliant with SP800-56Arev3, in accordance with scenario 2 (1) of IG D.F and used as part of the TLS protocol key exchange in accordance with scenario 2 (2) of IG D.F; that is, the shared secret computation (KAS-FFC-SSC and KAS-ECC-SSC) followed by key derivation using TLS KDF.

For Diffie-Hellman, the module supports the use of safe primes from RFC7919 for domain parameters and key generation, which are used in the TLS key agreement implemented by the module.

- TLS (RFC7919)
    - ffdhe2048 (ID = 256)
    - ffdhe3072 (ID = 257)
    - ffdhe4096 (ID = 258)
    - ffdhe6144 (ID = 259)
    - ffdhe8192 (ID = 260)

The module also supports the use of safe primes from RFC3526, which are part of the Modular Exponential (MODP) Diffie-Hellman groups that can be used for Internet Key

Exchange (IKE). Note that the module only implements key generation and verification, and shared secret computation using safe primes, but no part of the IKE protocol.

- IKEv2 (RFC3526)
- MODP-2048 (ID=14)
- MODP-3072 (ID=15)
- MODP-4096 (ID=16)

- MODP-6144 (ID=17)
- MODP-8192 (ID=18)

The module also provides the following key transport mechanisms:

- Key wrapping using AES-CCM, AES-GCM, and AES-CBC with HMAC, used in the context of the TLS protocol cipher suites with 128-bit or 256-bit keys.

According to Table 2: Comparable strengths in [SP 800-57rev5], the key sizes of AES, Diffie-Hellman and EC Diffie-Hellman provides the following security strength in Approved mode of operation:

- AES key wrapping using AES-CCM, AES-GCM, and AES in CBC mode and HMAC, provides between 128 or 256 bits of encryption strength.
- Diffie-Hellman shared secret computation provides between 112 and 200 bits of security strength.
- EC Diffie-Hellman shared secret computation provides between 128 and 256 bits of security strength.

The module supports the following key derivation methods according to [SP800-135rev1]:

- KDF for the TLS protocol, used as pseudo-random functions (PRF) for TLSv1.2, HKDF as part of TLSv1.3.

The module also supports password-based key derivation (PBKDF). The implementation is compliant with option 1a of [SP800-132]. Keys derived from passwords or passphrases using this method can only be used in storage applications.

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

The logical interfaces are the API through which the applications request services. The following table summarizes the logical interfaces:

Physical Port	Logical Interface(s)	Data That Passes
As a software-only module, the module does not have physical ports. Physical Ports are interpreted to be the physical ports of the hardware platform on which it runs.	Data Input	API input parameters
As a software-only module, the module does not have physical ports. Physical Ports are interpreted to be the physical ports of the hardware platform on which it runs.	Data Output	API output parameters
As a software-only module, the module does not have physical ports. Physical Ports are interpreted to be the physical ports of the hardware platform on which it runs.	Control Input	API function calls for control
As a software-only module, the module does not have physical ports. Physical Ports are interpreted to be the physical ports of the hardware platform on which it runs.	Status Output	API return codes, status parameters

Table 11: Ports and Interfaces

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

The module does not implement a control output interface.

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

4 Roles, Services, and Authentication

FIPS 140-3 does not require an authentication mechanism for level 1 modules. Therefore, the module does not implement an authentication mechanism for Crypto Officer. The module supports the Crypto Officer role only.

The Crypto Officer role is authorized to access all services provided by the module and this sole role is implicitly assumed by the operator of the module when performing a service.

4.1 Authentication Methods

N/A for this module.

The module does not support authentication.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 12: Roles

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

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Symmetric Encryption	Perform AES encryption	GNUTLS_FIPS140_OP_APPROVED	Key, Plaintext	Ciphertext	Symmetric Encryption with AES	Crypto Officer - AES key: W,E
Symmetric Decryption	Perform AES decryption	GNUTLS_FIPS140_OP_APPROVED	Key, Ciphertext	Plaintext	Symmetric Decryption with AES	Crypto Officer - AES key: W,E
Authenticated Symmetric Encryption	Encrypt a plaintext	GNUTLS_FIPS140_OP_APPROVED	Key, Plaintext, IV	Ciphertext, MAC tag	Authenticated Symmetric Encryption with AES	Crypto Officer - AES key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Authenticated Symmetric Decryption	Decrypt a ciphertext	GNUTLS_FIPS140_OP_APPROVED	Key, Ciphertext, IV, MAC tag	Plaintext	Authenticated Symmetric Decryption with AES	Crypto Officer - AES key: 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	Crypto Officer - AES key: W,E Crypto Officer - HMAC key: W,E
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	Crypto Officer - AES key: W,E Crypto Officer - HMAC key: W,E
Message Digest	Compute message digest	GNUTLS_FIPS140_OP_APPROVED	Message	Digest of the message	Message Digest with SHA	Crypto Officer
Random Number Generation	Generate random bitstrings	GNUTLS_FIPS140_OP_APPROVED	Number of bits	Random number	Random Number Generation with CTR_DRBG	Crypto Officer - Entropy Input: W,E - DRBG seed: G,E - DRBG internal state (V value, key): G,W,E
Message Authentication Code	Compute HMAC	GNUTLS_FIPS140_OP_APPROVED	HMAC key, message	Message authentication code	Message Authentication Code	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
(MAC) with HMAC					(MAC) with HMAC	- HMAC key: W,E
Message Authentication Code (MAC) with AES	Compute AES-based CMAC or GMAC	GNUTLS_FIPS140_OP_APPROVED	AES key, message	Message authentication code	Message Authentication Code (MAC) with AES	Crypto Officer - AES key: W,E
Shared Secret Computation with Diffie-Hellman	Compute a shared secret	GNUTLS_FIPS140_OP_APPROVED	Private key, public key from peer	Shared secret	Shared Secret Computation with Diffie-Hellman	Crypto Officer - Diffie-Hellman shared secret: G,R - Diffie-Hellman public key: W,E - Diffie-Hellman private key: W,E
Shared Secret Computation with EC Diffie-Hellman	Compute a shared secret	GNUTLS_FIPS140_OP_APPROVED	Private key, public key from peer	Shared secret	Shared Secret Computation with EC Diffie-Hellman	Crypto Officer - EC Diffie-Hellman shared secret: G,R - EC Diffie-Hellman public key: W,E - EC Diffie-Hellman private key: W,E
Digital Signature	Generate RSA signature	GNUTLS_FIPS140_OP_APPROVED	Message, hash	Digital signature	Message Digest with SHA Digital	Crypto Officer - RSA

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Generation with RSA			algorithm, private key		Signature Generation with RSA	private key: W,E
Digital Signature Generation with ECDSA	Generate ECDSA signature	GNUTLS_FIPS140_OP_APPROVED	Message, hash algorithm, private key	Digital signature	Message Digest with SHA Digital Signature Generation with ECDSA	Crypto Officer - ECDSA private key: W,E
Digital Signature Verification with RSA	Verify RSA	GNUTLS_FIPS140_OP_APPROVED	Message, signature, hash algorithm, public key	Verification result	Message Digest with SHA Digital Signature Verification with RSA	Crypto Officer - RSA public key: W,E
Digital Signature Verification with ECDSA	Verify ECDSA signature	GNUTLS_FIPS140_OP_APPROVED	Message, signature, hash algorithm, public key	Verification result	Message Digest with SHA Digital Signature Verification with ECDSA	Crypto Officer - ECDSA public key: W,E
Key Pair Generation with RSA	Generate RSA key pairs	GNUTLS_FIPS140_OP_APPROVED	Key size	Key pair	Random Number Generation with CTR_DRBG Key Pair Generation with RSA	Crypto Officer - Module-generated RSA private key: G,W,E - Module-generated RSA public key: G,W,E
Key Pair Generation with ECDSA	Generate ECDSA key pairs	GNUTLS_FIPS140_OP_APPROVED	Key size, enabled-curve	Key pair	Random Number Generation with	Crypto Officer - Module-generated

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					CTR_DRBG Key Pair Generation with ECDSA	ECDSA private key: G,W,E - Module-generated ECDSA public key: G,W,E
Key Pair Generation with Safe Primes	Generate DH key pairs	GNUTLS_FIPS140_OP_APPROVED	Key size	Key pair	Random Number Generation with CTR_DRBG Key Pair Generation with Safe Primes	Crypto Officer - Module-generated Diffie-Hellman Private Key: G,W,E - Module-generated Diffie-Hellman Public Key: G,W,E
Public Key Verification with ECDSA	Verify ECDSA public key	GNUTLS_FIPS140_OP_APPROVED	Key	Return codes/log messages	Public Key Verification with ECDSA	Crypto Officer - ECDSA public key: W,E
TLS 1.2 Key Derivation (derivation of TLS Master Secret)	Perform key derivation using TLS 1.2 KDF	GNUTLS_FIPS140_OP_APPROVED	TLS Pre-master Secret	TLS Master secret	Key Derivation with TLS 1.2 KDF	Crypto Officer - TLS Pre-master Secret: W,E - TLS Master Secret: G,W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
TLS 1.2 Key Derivation (derivation of TLS Derived Secret)	Perform key derivation using TLS 1.2 KDF	GNUTLS_FIPS140_OP_APPROVED	TLS Master Secret	TLS Derived Secret	Key Derivation with TLS 1.2 KDF	Crypto Officer - TLS Master Secret: W,E - TLS Derived Secret: G,W,E
HKDF Key Derivation (derivation of HKDF Derived Key)	Perform key derivation using HKDF	GNUTLS_FIPS140_OP_APPROVED	Shared Secret	HKDF Derived Key	Key Derivation (as part of TLSv1.3) with KDA HKDF	Crypto Officer - HKDF Derived Key: G,R - Diffie-Hellman shared secret: W,E - EC Diffie-Hellman shared secret: W,E
Key Derivation with PBKDF	Perform password-based key derivation	GNUTLS_FIPS140_OP_APPROVED	Password or passphrase	PBKDF Derived key	Key Derivation with PBKDF	Crypto Officer - PBKDF password or passphrase: W,E - PBKDF Derived key: G
Transport Layer Security (TLS) Network Protocol	Provide supported cipher suites (listed in Appendix A) in	GNUTLS_FIPS140_OP_APPROVED	Cipher-suites listed in Appendix A, Digital Certificate, Public and Private Keys,	Return codes and/or log messages, Application data	Symmetric Encryption with AES Symmetric Decryption with AES Authenticated Symmetric	Crypto Officer - AES key: W,E - HMAC key: W,E - RSA public key:

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	approved mode		Application Data		Encryption with AES Authenticated Symmetric Decryption with AES Authenticated Symmetric Encryption (in the context of the TLS 1.2/1.3 protocol) with AES-GCM Authenticated Symmetric Decryption (in the context of the TLS 1.2/1.3 protocol) with AES-GCM Message Authentication Code (MAC) with HMAC Message Digest with SHA Digital Signature Generation with RSA Digital Signature Generation with ECDSA Digital Signature	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 - Module-generated EC Diffie-Hellman Private Key: G,E - TLS Pre-master Secret: G,E - TLS Master Secret: G,E - TLS Derived

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

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					(in the context of the TLS 1.2/1.3 protocol) with AES-GCM Message Authentication Code (MAC) with AES Message Authentication Code (MAC) with HMAC Message Digest with SHA Key Derivation with TLS 1.2 KDF Key Derivation (as part of TLSv1.3) with KDA HKDF Key Derivation with PBKDF Digital Signature Generation with RSA Digital Signature Generation with ECDSA Digital Signature Verification	

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					with RSA Digital Signature Verification with ECDSA Key Pair Generation with RSA Key Pair Generation with ECDSA Key Pair Generation with Safe Primes Public Key Verification with ECDSA Shared Secret Computation with Diffie-Hellman Shared Secret Computation with EC Diffie-Hellman Random Number Generation with CTR_DRBG Key Wrapping Key Unwrapping	
Show module name and version	Show module name and version	N/A	N/A	Name and version information	None	Crypto Officer

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

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						private key: Z - ECDSA public key: Z - Module-generated EC Diffie-Hellman Private Key: Z - Module-generated EC Diffie-Hellman Public Key: Z - EC Diffie-Hellman private key: Z - EC Diffie-Hellman public key: Z - Module-generated Diffie-Hellman Private Key: Z - Module-generated Diffie-Hellman Public Key: Z - Diffie-Hellman private key: Z - Diffie-

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

Table 13: Approved Services

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

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

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

- **Zeroize (Z):** The module zeroizes the SSP.
- **N/A:** The module does not access any SSP or key during its operation.

The service indicator is invoked by calling the function "gnutls_fips140_get_operation_state()" and it returns "GNUTLS_FIPS140_OP_APPROVED" or "GNUTLS_FIPS140_OP_NOT_APPROVED" depending on whether the API invoked corresponds to an approved or non-approved algorithm.

4.4 Non-Approved Services

Name	Description	Algorithms	Role
Symmetric key generation	Generate symmetric key other than AES and HMAC keys	DRBG when key length is less than 112 bits	CO
Symmetric encryption	Compute the cipher for encryption	Blowfish Camellia CAST ChaCha20 DES GOST RC2, RC4 Salsa20 Serpent Triple-DES Twofish	CO
Symmetric decryption	Compute the cipher for decryption	Blowfish Camellia CAST ChaCha20 DES GOST RC2, RC4 Salsa20 Serpent Triple-DES Twofish	CO
Asymmetric key generation with RSA	Generate RSA key pairs	RSA with keys smaller than 2048 bits or greater than 4096 bits.	CO
Asymmetric key generation with DSA	Generate DSA key pairs	DSA	CO

Name	Description	Algorithms	Role
Asymmetric key generation with ECDSA	Generate ECDSA key pairs	ECDSA with curves not listed in Table "Approved Algorithms"	CO
Digital signature generation	Sign RSA, DSA, and ECDSA signatures	DSA ECDSA with curves not listed in Table "Approved Algorithms" RSA with keys smaller than 2048 bits or greater than 4096 bits. SHA-1	CO
Digital signature verification	Verify RSA, DSA, and ECDSA signatures	DSA ECDSA with curves not listed in Table "Approved Algorithms" RSA with keys smaller than 1024 bits or greater than 4096 bits. SHA-1	CO
Asymmetric key generation	Generate RSA, DSA, and ECDSA key pairs	DSA ECDSA with curves not listed in Table "Approved Algorithms" RSA with keys smaller than 1024 bits or greater than 4096 bits.	CO
Message digest	Compute message digest	GOST MD2, MD4, MD5 RMD160 SM3 STREEBOG	CO
Message Authentication Code (MAC)	Compute HMAC	HMAC with keys smaller than 112-bit HMAC with GOST MD2, MD4, MD5 RMD160 STREEBOG UMAC	CO
Key encapsulation	Perform RSA key encapsulation	RSA encryption and decryption with any key sizes.	CO
Key unencapsulation	Perform RSA key unencapsulation	RSA encryption and decryption with any key sizes.	CO

Name	Description	Algorithms	Role
Shared Secret Computation with Diffie-Hellman	Perform DH key agreement	Diffie-Hellman with keys generated with domain parameters other than safe primes	CO
Shared Secret Computation with EC Diffie-Hellman	Perform ECDH key agreement	EC Diffie-Hellman with curves not listed in Table "Approved Algorithms"	CO
Key Derivation with PBKDF	Perform password-based key derivation	PBKDF with non-approved message digest algorithms	CO
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 number	Yarrow	CO
Authenticated encryption	Perform authenticated encryption	Chacha20, Poly1305 and AES-GCM	CO
Authenticated decryption	Perform authenticated decryption	Chacha20, Poly1305 and AES-GCM	CO
Domain parameter generation	Generate domain parameter	DSA	CO

Table 14: Non-Approved Services

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

4.5 External Software/Firmware Loaded

The module does not have the capability of loading software or firmware from an external source.

5 Software/Firmware Security

5.1 Integrity Techniques

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. The .hmac file has HMAC value for libgnutls, libnettle and libhogweed listed in section 2. If the HMAC values do not match, the test fails, and the module enters the error state.

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

5.2 Initiate on Demand

The module provides the Self-Test service to perform self-tests on demand which includes the pre-operational test (i.e., integrity test) and the cryptographic algorithm self-tests (CASTs). The Self-Tests service can be called on demand by invoking the gnutls_fips140_run_self_tests() function which will perform integrity tests and the cryptographic algorithms self-tests. Additionally, the Self-Test service can be invoked by powering-off and reloading the module. During the execution of the on-demand self-tests, services are not available, and no data output is possible.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

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

The module shall be installed as stated in Section 11. The user should confirm that the module is installed correctly by running:

1. fips-mode-setup --check command to verify that the system is operating in Approved mode
2. check the output of the the gnutls_get_library_config() API, which should output GnuTLS cryptography module for AlmaLinux 9 3.7.6-396796fe0a32b434

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

There are no concurrent operators.

6.2 Configuration Settings and Restrictions

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

7 Physical Security

The module is comprised of software only and therefore this section is Not Applicable (N/A).

8 Non-Invasive Security

This module does not implement any non-invasive security mechanism and therefore this section is Not Applicable (N/A).

9 Sensitive Security Parameters Management

9.1 Storage Areas

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

Table 15: Storage Areas

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

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

9.2 SSP Input-Output Methods

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

Table 16: SSP Input-Output Methods

SSPs are provided to the module via API input parameters in plaintext form and output via API output parameters in plaintext form within the physical perimeter of the operational environment. This is allowed by [FIPS140-3_IG] IG 9.5.A, according to the “CM Software to/from App via TOEPP Path” entry on the Key Establishment Table.

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

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Zeroize Context	The memory occupied by SSPs is allocated by regular memory allocation operating system calls.	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable. The completion of the zeroization routine indicates that the zeroization procedure succeeded.	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() HKDF Derived Key: gnutls_deinit() Diffie-Hellman Public Key, Diffie-Hellman private key: gnutls_pk_params_clear() EC Diffie-Hellman public key, EC Diffie-Hellman private key: gnutls_pk_params_clear() Diffie-Hellman Shared Secret: zeroize key() EC Diffie-Hellman Shared Secret: zeroize_key() All SSPs: gnutls_global_deinit()
Automatic	Automatically zeroized by the module when no longer needed	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable.	N/A
Reset	De-allocates the volatile memory used to store SSPs	Volatile memory used by the module is overwritten within nanoseconds when power is removed. Module power off indicates that the zeroization procedure succeeded.	Unloading and reloading the module

Table 17: SSP Zeroization Methods

The memory occupied by SSPs is allocated by regular memory allocation operating system calls. The application that is acting as the CO is responsible for calling the appropriate zeroization functions provided in the module's API and listed in Table 20. Calling the gnutls_deinit() will zeroize the SSPs stored in the TLS protocol internal state and also invoke the corresponding API functions listed in Table 20 to zeroize SSPs. The zeroization functions overwrite the memory occupied by SSPs with “zeros” and deallocate the memory with the regular memory deallocation operating system call. The completion of a zeroization routine(s) will indicate that a zeroization procedure succeeded. All data output is inhibited during zeroization.

9.4 SSPs

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

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
						AES-GCM Key Wrapping Key Unwrapping
HMAC key	HMAC key used for computing MAC tags	112 to 256 bits - 112 to 256 bits	Symmetric key - CSP			Message Authentication Code (MAC) with HMAC Key Wrapping Key Unwrapping
Module-generated RSA private key	RSA private key generated through asymmetric key generation	2048, 3072, 4096 bits - 112, 128, 149 bits	Private key - CSP	Key Pair Generation with RSA		
Module-generated RSA public key	RSA public key generated through asymmetric key generation	2048, 3072, 4096 bits - 112, 128, 149 bits	Public key - PSP	Key Pair Generation with RSA		
RSA private key	RSA private key used for digital signature generation	2048, 3072, 4096 bits - 112-149 bits	Private key - CSP			Digital Signature Generation with RSA
RSA public key	RSA private key used for digital signature verification	2048, 3072, 4096 bits - 112-149 bits	Public key - PSP			Digital Signature Verification with RSA
PBKDF password or passphrase	Password used to derive symmetric keys	14 characters minimum - 10^-14 minimum probability	Password - 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	128-256 bits - 128-256 bits	Derived key - CSP	Key Derivation with PBKDF		
Module-generated ECDSA private key	ECDSA private key generated through the asymmetric key generation	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP	Key Pair Generation with ECDSA		
Module-generated ECDSA public key	ECDSA private key generated through the asymmetric key generation	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP	Key Pair Generation with ECDSA		
ECDSA private key	ECDSA private key used for digital signature generation	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP			Digital Signature Generation with ECDSA Public Key Verification with ECDSA
ECDSA public key	ECDSA public key used for digital signature generation	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP			Digital Signature Verification with ECDSA Public Key Verification with ECDSA
Module-generated EC Diffie-Hellman Public Key	EC Diffie-Hellman public key generated during asymmetric key generation	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP	Key Pair Generation with ECDSA		TLS Handshake
Module-generated	EC Diffie-Hellman private key	P-256, P-384, P-521	Private key - CSP	Key Pair Generation		TLS Handshake

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
EC Diffie-Hellman Private Key	generated during asymmetric key generation	- 128, 192, 256 bits		with ECDSA		
EC Diffie-Hellman public key	Public key used for Shared Secret Computation	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP			Shared Secret Computation with EC Diffie-Hellman
EC Diffie-Hellman private key	Private key used for Shared Secret Computation	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP			Shared Secret Computation with EC Diffie-Hellman
Module-generated Diffie-Hellman Public Key	Diffie-Hellman public key generated during Safe Primes Key Generation	2048, 3072, 4096, 6144, 8192 bits - 112-200 bits	Public key - PSP	Key Pair Generation with Safe Primes		TLS Handshake
Module-generated Diffie-Hellman Private Key	Diffie-Hellman private key generated during Safe Primes Key Generation	2048, 3072, 4096, 6144, 8192 bits - 112-200 bits	Private key - CSP	Key Pair Generation with Safe Primes		TLS Handshake
Diffie-Hellman public key	Public key used for Shared Secret Computation	2048-8192 bits - 112-200 bits	Public key - PSP			Shared Secret Computation with Diffie-Hellman
Diffie-Hellman private key	Private key used for Shared Secret Computation	2048-8192 bits - 112-200 bits	Private key - CSP			Shared Secret Computation with Diffie-Hellman

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Diffie-Hellman shared secret	Shared secret generated by Diffie-Hellman	2048-8192 bits - 112 to 200 bits	Shared Secret - CSP		Shared Secret Computation with Diffie-Hellman	
EC Diffie-Hellman shared secret	Shared secret generated by EC Diffie-Hellman	P-256, P-384, P-521 - 128 to 256 bits	Shared Secret - CSP		Shared Secret Computation with EC Diffie-Hellman	
Entropy Input	Entropy input used to seed the DRBG	128-256 bits - 128-256 bits	Entropy Input - CSP			Random Number Generation with CTR_DRBG
DRBG seed	DRBG seed derived from entropy input	128 to 256 bits - 128 to 256 bits	Seed - CSP	Random Number Generation with CTR_DRBG		Random Number Generation with CTR_DRBG
DRBG internal state (V value, key)	Internal state of the CTR_DRBG	384 bits - 128 to 256 bits	Internal State - CSP	Random Number Generation with CTR_DRBG		Random Number Generation with CTR_DRBG
TLS Pre-master Secret	TLS Pre-master Secret used for deriving the TLS Master Secret	112 to 256 bits - 112 to 256 bits	Secret - CSP		Shared Secret Computation with Diffie-Hellman Shared Secret Computation with EC Diffie-Hellman	TLS Handshake

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
TLS Master Secret	TLS Master Secret used for deriving the TLS Derived Secret	384 bits - 384 bits	Secret - CSP	Key Derivation with TLS 1.2 KDF		TLS Handshake
TLS Derived Secret	Used as encryption key or MAC key	112-256 bits - 112-256 bits	Derived secret - CSP	Key Derivation with TLS 1.2 KDF		TLS Handshake
HKDF Derived Key	HKDF (used as part of TLS 1.3 protocol) derived key	112-256 bits - 112-256 bits	Derived secret - CSP	Key Derivation (as part of TLSv1.3) with KDA HKDF		TLS Handshake

Table 18: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES key	API input parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	
HMAC key	API input parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	
Module-generated RSA private key	API output parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Module-generated RSA public key:Paired With DRBG internal state (V value, key):Derived From
Module-generated RSA public key	API output parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Module-generated RSA private key:Paired With DRBG internal state (V value, key):Derived From

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
RSA private key	API input parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	RSA public key:Paired With
RSA public key	API input parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	RSA private key:Paired With
PBKDF password or passphrase	API input parameters (plaintext)	RAM:Plaintext	For the duration of the service	Zeroize Context Reset	PBKDF Derived key:Derived From
PBKDF Derived key	API output parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	PBKDF password or passphrase:Derived From
Module-generated ECDSA private key	API output parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Module-generated ECDSA public key:Paired With DRBG internal state (V value, key):Derived From
Module-generated ECDSA public key	API output parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Module-generated ECDSA private key:Paired With DRBG internal state (V value, key):Derived From
ECDSA private key	API input parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	ECDSA public key:Paired With
ECDSA public key	API input parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	ECDSA private key:Paired With
Module-generated EC Diffie-Hellman Public Key	API output parameters (plaintext)	RAM:Plaintext	For the duration of the service	Zeroize Context Reset	Module-generated EC Diffie-Hellman Private Key:Paired With DRBG internal state (V

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
					value, key):Derived From
Module-generated EC Diffie-Hellman Private Key	API output parameters (plaintext)	RAM:Plaintext	For the duration of the service	Zeroize Context Reset	Module-generated EC Diffie-Hellman Public Key:Paired With DRBG internal state (V value, key):Derived From
EC Diffie-Hellman public key	API input parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	EC Diffie-Hellman private key:Paired With EC Diffie-Hellman shared secret:Used With
EC Diffie-Hellman private key	API input parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	EC Diffie-Hellman public key:Paired With EC Diffie-Hellman shared secret:Used With
Module-generated Diffie-Hellman Public Key	API output parameters (plaintext)	RAM:Plaintext	For the duration of the service	Zeroize Context Reset	Module-generated Diffie-Hellman Private Key:Paired With DRBG internal state (V value, key):Derived From
Module-generated Diffie-Hellman Private Key	API output parameters (plaintext)	RAM:Plaintext	For the duration of the service	Zeroize Context Reset	Module-generated Diffie-Hellman Public Key:Paired With DRBG internal state (V value, key):Derived From
Diffie-Hellman public key	API input parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Diffie-Hellman private key:Paired With Diffie-Hellman shared secret:Used With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Diffie-Hellman private key	API input parameters (plaintext) API output parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Diffie-Hellman public key:Paired With Diffie-Hellman shared secret:Used With
Diffie-Hellman shared secret	API output parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	Diffie-Hellman public key:Used With Diffie-Hellman private key:Used With
EC Diffie-Hellman shared secret	API output parameters (plaintext)	RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	EC Diffie-Hellman public key:Used With EC Diffie-Hellman private key:Used With
Entropy Input		RAM:Plaintext	From generation until DRBG Seed is created	Zeroize Context Automatic	DRBG seed:Derives
DRBG seed		RAM:Plaintext	While the DRBG is instantiated	Zeroize Context Automatic	Entropy Input:Derived From
DRBG internal state (V value, key)		RAM:Plaintext	While the module is operational	Zeroize Context Automatic	DRBG seed:Used With
TLS Pre-master Secret		RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	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

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
					Diffie-Hellman Private Key:Used With
TLS Master Secret		RAM:Plaintext	Until explicitly zeroized by operator	Zeroize Context Reset	TLS Pre-master Secret:Derived From TLS Derived Secret:Derived From
TLS Derived Secret	API output parameters (plaintext)	RAM:Plaintext	For the duration of the service	Zeroize Context Reset	TLS Master Secret:Derived From
HKDF Derived Key	API output parameters (plaintext)	RAM:Plaintext	For the duration of the service	Zeroize Context Reset	Diffie-Hellman shared secret:Derived From EC Diffie-Hellman shared secret:Derived From

Table 19: SSP Table 2

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

9.5 Transitions

The SHA-1 algorithm as implemented by the module will be non-approved for all purposes, starting January 1, 2030.

10 Self-Tests

10.1 Pre-Operational Self-Tests

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

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A5122)	256-bit key	Message authentication	SW/FW Integrity	Module becomes operational and services are available for use	Integrity test of the shared libraries that comprise the module (for libgnutls, libnettle and libhogweed)

Table 20: Pre-Operational Self-Tests

The module performs the pre-operational self-test and CASTs automatically when the module is loaded into memory. Pre-operational self-test ensure that the module is not corrupted, and the CASTs ensure that the cryptographic algorithms work as expected. While the module is executing the self-tests, the module services are not available, and input and output are inhibited. The module is not available for use by the calling application until the pre-operational self-test and the CASTs are completed successfully. After the pre-operational test and the CASTs succeed, the module becomes operational. If any of the pre-operational test or any of the CASTs fail an error message is returned, and the module transitions to the error state.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC (A5114)	128 and 256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC (A5115)	128 and 256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC (A5116)	128 and 256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC (A5117)	128 and 256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC (A5122)	128 and 256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CBC (A5114)	128 and 256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CBC (A5115)	128 and 256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CBC (A5116)	128 and 256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC (A5117)	128 and 256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CBC (A5122)	128 and 256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CFB8 (A5120)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CFB8 (A5125)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-CFB8 (A5120)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-CFB8 (A5125)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-GCM (A5114)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM (A5115)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM (A5116)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM (A5117)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM (A5122)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-GCM (A5114)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-GCM (A5115)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM (A5116)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM (A5117)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-GCM (A5122)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization
AES-XTS Testing Revision 2.0 (A5123)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Encryption	Module initialization
AES-XTS Testing Revision 2.0 (A5123)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	Decryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CMAC (A5114)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	MAC generation	Module initialization
AES-CMAC (A5117)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	MAC generation	Module initialization
AES-CMAC (A5122)	256-bit keys	KAT	CAST	Module becomes operational and services are available for use	MAC generation	Module initialization
Counter DRBG (A5122)	256-bit keys without DF, without PR	KAT	CAST	Module becomes operational and services are available for use	KAT CTR_DRBG with AES with 256-bit keys without DF, without PR	Module initialization
Counter DRBG (A5122)	Health tests	Health tests according to section 11.3 of [SP800-90Ar1]	CAST	Module is operational and services are available for use	Health tests	Module initialization
KAS-FFC-SSC Sp800-56Ar3 (A5122)	ffdhe3072	KAT	CAST	Module becomes operational and services are available for use	Primitive “Z” Computation	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KAS-ECC-SSC Sp800-56Ar3 (A5122)	P-256	KAT	CAST	Module becomes operational and services are available for use	Primitive "Z" Computation	Module initialization
ECDSA SigGen (FIPS186-5) (A5122)	P-256 using SHA-256, P-384 using SHA-384, and P-521 using SHA-512	KAT	CAST	Module becomes operational and services are available for use	Signature Generation	Module initialization
ECDSA SigVer (FIPS186-5) (A5122)	P-256 using SHA-256, P-384 using SHA-384, and P-521 using SHA-512	KAT	CAST	Module becomes operational and services are available for use	Signature Verification	Module initialization
KDA HKDF Sp800-56Cr1 (A5121)	SHA-256	KAT	CAST	Module becomes operational and services are available for use	Key Derivation (as part of TLSv1.3) with KDA HKDF	Module initialization
HMAC-SHA-1 (A5117)	128-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
HMAC-SHA2-224 (A5117)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-256 (A5117)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
HMAC-SHA2-384 (A5117)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
HMAC-SHA2-512 (A5117)	160-bit key	KAT	CAST	Module becomes operational and services are available for use	Message Authentication	Module initialization
PBKDF (A5122)	SHA-256 with 4096 iterations and 288-bit salt	KAT	CAST	Module becomes operational and services are available for use	Key Derivation with PBKDF	Module initialization
RSA SigGen (FIPS186-5) (A5122)	RSA PKCS#1 v1.5 with 2048-bit key using SHA-256	KAT	CAST	Module becomes operational and services are available for use	Signature Generation	Module initialization
RSA SigVer (FIPS186-5) (A5122)	RSA PKCS#1 v1.5 with 2048-bit key using SHA-256	KAT	CAST	Module becomes operational and services are available for use	Signature Verification	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA3-224 (A5118)	32-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization
SHA3-256 (A5118)	32-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization
SHA3-384 (A5118)	64-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization
SHA3-512 (A5118)	136-bit message	KAT	CAST	Module becomes operational and services are available for use	Message Digest	Module initialization
TLS v1.2 KDF RFC7627 (A5122)	SHA-256	KAT	CAST	Module becomes operational and services are available for use	Key Derivation with TLS v1.2 KDF RFC7627	Module initialization
ECDSA KeyGen (FIPS186-5) (A5122)	SHA-256 with the respective curve	Signature generation and verification	PCT	Successful key pair generation	Signature generation and verification	Key Pair Generation

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
RSA KeyGen (FIPS186-5) (A5122)	PKCS#1v1.5 with SHA-256	Signature generation and verification	PCT	Successful key pair generation	Signature generation and verification	Key Pair Generation
Safe Primes Key Generation (A5122)	N/A	PCT according to section 5.6.2.1.4 of [SP800-56Arev3]	PCT	Successful key pair generation	PCT according to section 5.6.2.1.4 of [SP800-56Arev3]	Key Pair Generation
ECDSA KeyGen (FIPS186-5) (A5122)	SHA-256 with the respective curve	Signature generation and verification	PCT	Signature generation and verification	Signature generation PCT that covers key pair generation for EC Diffie-Hellman	Key Pair Generation

Table 21: Conditional Self-Tests

Conditional Cryptographic Algorithm Tests

The module performs self-tests on approved cryptographic algorithms, using the tests shown in the table above. Data output through the data output interface is inhibited during the self-tests. All CASTs performed are in the form of the Known Answer Tests (KATs) and are run prior to performing the integrity test. The KAT includes comparison of the calculated output with the expected known answer, hard coded as part of the test vectors used in the test. If one of the conditional self-tests fail, the module transitions to the 'Error' state and a corresponding error indication is given.

The entropy source performs its required self-tests; those are not listed here, as the entropy source is not part of the cryptographic boundary of the module.

Conditional Pair-Wise Consistency Tests

The module implements RSA, ECDSA, DH and ECDH key generation service and performs the respective pairwise consistency test (PCT) using sign and verify functions when the keys are generated.

If any of the tests fails, the module returns an error code and enters the Error state. When the module is in the Error state, no data is output, and cryptographic operations are not allowed.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A5122)	Message authentication	SW/FW Integrity	On demand	Manually

Table 22: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC (A5114)	KAT	CAST	On demand	Manually
AES-CBC (A5115)	KAT	CAST	On demand	Manually
AES-CBC (A5116)	KAT	CAST	On demand	Manually
AES-CBC (A5117)	KAT	CAST	On demand	Manually
AES-CBC (A5122)	KAT	CAST	On demand	Manually
AES-CBC (A5114)	KAT	CAST	On demand	Manually
AES-CBC (A5115)	KAT	CAST	On demand	Manually
AES-CBC (A5116)	KAT	CAST	On demand	Manually
AES-CBC (A5117)	KAT	CAST	On demand	Manually
AES-CBC (A5122)	KAT	CAST	On demand	Manually
AES-CFB8 (A5120)	KAT	CAST	On demand	Manually
AES-CFB8 (A5125)	KAT	CAST	On demand	Manually
AES-CFB8 (A5120)	KAT	CAST	On demand	Manually
AES-CFB8 (A5125)	KAT	CAST	On demand	Manually
AES-GCM (A5114)	KAT	CAST	On demand	Manually
AES-GCM (A5115)	KAT	CAST	On demand	Manually
AES-GCM (A5116)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-GCM (A5117)	KAT	CAST	On demand	Manually
AES-GCM (A5122)	KAT	CAST	On demand	Manually
AES-GCM (A5114)	KAT	CAST	On demand	Manually
AES-GCM (A5115)	KAT	CAST	On demand	Manually
AES-GCM (A5116)	KAT	CAST	On demand	Manually
AES-GCM (A5117)	KAT	CAST	On demand	Manually
AES-GCM (A5122)	KAT	CAST	On demand	Manually
AES-XTS Testing Revision 2.0 (A5123)	KAT	CAST	On demand	Manually
AES-XTS Testing Revision 2.0 (A5123)	KAT	CAST	On demand	Manually
AES-CMAC (A5114)	KAT	CAST	On demand	Manually
AES-CMAC (A5117)	KAT	CAST	On demand	Manually
AES-CMAC (A5122)	KAT	CAST	On demand	Manually
Counter DRBG (A5122)	KAT	CAST	On demand	Manually
Counter DRBG (A5122)	Health tests according to section 11.3 of [SP800-90Ar1]	CAST	On demand	Manually
KAS-FFC-SSC Sp800-56Ar3 (A5122)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
KAS-ECC-SSC Sp800-56Ar3 (A5122)	KAT	CAST	On demand	Manually
ECDSA SigGen (FIPS186-5) (A5122)	KAT	CAST	On demand	Manually
ECDSA SigVer (FIPS186-5) (A5122)	KAT	CAST	On demand	Manually
KDA HKDF Sp800-56Cr1 (A5121)	KAT	CAST	On demand	Manually
HMAC-SHA-1 (A5117)	KAT	CAST	On demand	Manually
HMAC-SHA2-224 (A5117)	KAT	CAST	On demand	Manually
HMAC-SHA2-256 (A5117)	KAT	CAST	On demand	Manually
HMAC-SHA2-384 (A5117)	KAT	CAST	On demand	Manually
HMAC-SHA2-512 (A5117)	KAT	CAST	On demand	Manually
PBKDF (A5122)	KAT	CAST	On demand	Manually
RSA SigGen (FIPS186-5) (A5122)	KAT	CAST	On demand	Manually
RSA SigVer (FIPS186-5) (A5122)	KAT	CAST	On demand	Manually
SHA3-224 (A5118)	KAT	CAST	On demand	Manually
SHA3-256 (A5118)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
SHA3-384 (A5118)	KAT	CAST	On demand	Manually
SHA3-512 (A5118)	KAT	CAST	On demand	Manually
TLS v1.2 KDF RFC7627 (A5122)	KAT	CAST	On demand	Manually
ECDSA KeyGen (FIPS186-5) (A5122)	Signature generation and verification	PCT	On demand	Manually
RSA KeyGen (FIPS186-5) (A5122)	Signature generation and verification	PCT	On demand	Manually
Safe Primes Key Generation (A5122)	PCT according to section 5.6.2.1.4 of [SP800-56Arev3]	PCT	On demand	Manually
ECDSA KeyGen (FIPS186-5) (A5122)	Signature generation and verification	PCT	On demand	Manually

Table 23: Conditional Periodic Information

This information can be found in Section 5.2.

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	The module must be restarted and perform the pre-operational self-test and the CASTs to recover from these errors.	GNUTLS_E_SELF_TEST_ERROR (-400); GNUTLS_E_RANDOM_FAILED (-206); GNUTLS_E_PK_GENERATION_ERROR (-403); GNUTLS_E_LIB_IN_ERROR_STATE (-402)

Name	Description	Conditions	Recovery Method	Indicator
		key pair fails the PCT When the module is in error state and caller requests cryptographic operations		

Table 24: Error States

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

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

10.5 Operator Initiation of Self-Tests

The module provides the Self-Test service to perform self-tests on demand which includes the pre-operational test (i.e., integrity test) and the cryptographic algorithm self-tests (CASTs). The Self-Tests service can be called on demand by invoking the gnutls_fips140_run_self_tests() function which will perform integrity tests and the cryptographic algorithms self-tests. Additionally, the Self-Test service can be invoked by powering-off and reloading the module. During the execution of the on-demand self-tests, services are not available, and no data output is possible.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

The module is distributed as a part of the GnuTLS cryptography module for AlmaLinux 9 package in the form of the gnutls-3.7.6-23.el9_2.tuxcare.3.x86_64 RPM package for x86 systems.

The binaries of the 'GnuTLS cryptography module for AlmaLinux 9 version 3.7.6-396796fe0a32b434' are contained in the RPM packages for delivery listed below, which contain the FIPS validated module:

- gnutls-3.7.6-23.el9_2.tuxcare.3.x86_64.rpm
- nettle-3.8-3.el9_2.tuxcare.1.x86_64.rpm

Before the 'GnuTLS cryptography module for AlmaLinux 9' RPM packages are installed, the system must operate in Approved mode. This can be achieved by:

- Starting the installation in Approved mode. Add the fips=1 option to the kernel command line during the system installation. During the software selection stage, do not install any third-party software. More information can be found at the vendor documentation.
- Switching the system into Approved mode after the installation. Execute the fips-mode-setup --enable command. Restart the system. More information can be found at the vendor documentation.

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

11.2 Administrator Guidance

All the functions, ports and logical interfaces described in this document are available to the Crypto Officer.

11.3 Non-Administrator Guidance

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

11.4 End of Life

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

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

12 Mitigation of Other Attacks

12.1 Attack List

RSA timing attacks.

12.2 Mitigation Effectiveness

RSA is vulnerable to timing attacks. In a setup where attackers can measure the time of RSA decryption or signature operations, blinding is always used to protect the RSA operation from that attack.

The internal API function of rsa_blind() and rsa_unblind() are called by the module for RSA signature generation and RSA decryption operations. The module generates a random blinding factor and include this random value in the RSA operations to prevent RSA timing attacks.

Appendix A. TLS Cipher Suites

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

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

Cipher Suite	ID	Reference
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
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
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter Mode
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
KAT	Known Answer Test
KW	AES Key Wrap
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
PAA	Processor Algorithm Acceleration
PAI	Processor Algorithm Implementation
PR	Prediction Resistance
PSP	Public Security Parameter
PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, Adleman
SHA	Secure Hash Algorithm
SSP	Sensitive Security Parameter
XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

Appendix C. References

FIPS140-3	**FIPS PUB 140-3 - Security Requirements For Cryptographic Modules** March 2019 https://doi.org/10.6028/NIST.FIPS.140-3
FIPS140-3_IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program** January 2024 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)** March 2012 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS186-5	**Digital Signature Standard (DSS)** February 2023 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf
FIPS197	**Advanced Encryption Standard** November 2001 https://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS198-1	**The Keyed Hash Message Authentication Code (HMAC)** July 2008 https://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
SP800-38A	**NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 https://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 https://csrc.nist.gov/publications/detail/sp/800-38b/final
SP800-38D	**NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.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 https://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf
SP800-38F	**NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP800-52rev2	**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
SP800-56Arev3	**NIST Special Publication 800-56A – Revision 3 - Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography** April 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf

SP800-56Crev2	**NIST Special Publication 800-56C – Revision 2 - Recommendation for Key-Derivation Methods in Key-Establishment Schemes** August 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf
SP800-90Arev1	**NIST Special Publication 800-90A – Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP800-90B	**NIST Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf
SP800-132	**NIST Special Publication 800-132 - Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** December 2010 https://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf
SP800-133rev2	**NIST Special Publication 800-133 – Revision 2 - Recommendation for Cryptographic Key Generation** June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf
SP800-135rev1	**NIST Special Publication 800-135 – Revision 1 – Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf
SP800-140Br1	**NIST Special Publication 800-140B – Revision 1 - CMVP Security Policy Requirements** November 2023 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140Br1.pdf