FIPS 140-3 Non-Proprietary Security Policy

Oracle Corporation

Oracle Linux 9 GnuTLS Cryptographic Module

Software Version: 3.7.6-39e2433a29b33b55

Prepared by:

atsec information security corporation

4516 Seton Center Pkwy, Suite 250

Austin, TX 78759

www.atsec.com

Title: Oracle 9 GnuTLS Cryptographic Module Security Policy

Date: August 28th, 2025

Contributing Authors:

Oracle Linux Engineering

Security Evaluations – Global Product Security

atsec information security

Oracle Corporation

World Headquarters

2300 Oracle Way

Austin, TX 78741

U.S.A.

Worldwide Inquiries:

Phone: +1.650.506.7000

Fax: +1.650.506.7200

www.oracle.com

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for version 3.7.6-39e2433a29b33b55 of the Oracle 9 GnuTLS Cryptographic Module. It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-3 (Federal Information Processing Standards Publication 140-3) for an overall Security Level 1 module.

1.2 Security Levels

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

Table 1: Security Levels

1.3 Additional Information

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

This Non-Proprietary Security Policy may be reproduced and distributed, but only whole and intact and including this notice. Other documentation is proprietary to their authors.

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 Oracle 9 GnuTLS Cryptographic Module (hereafter referred to as “the module”) is a cryptographic module that provides cryptographic services to applications running in the user space of the underlying operating system through a C language Application Program Interface (API).

Module Type: Software

Module Embodiment: MultiChipStand

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

The module components consist of the libgnutls.so.30, libnettle.so.8, libhogweed.so.6, and libgmp.so.10. The module integrity is verified for each of the component separately using HMAC at power on by comparing with the pre-computed HMAC values stored in .libgnutls.so.30.hmac file

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

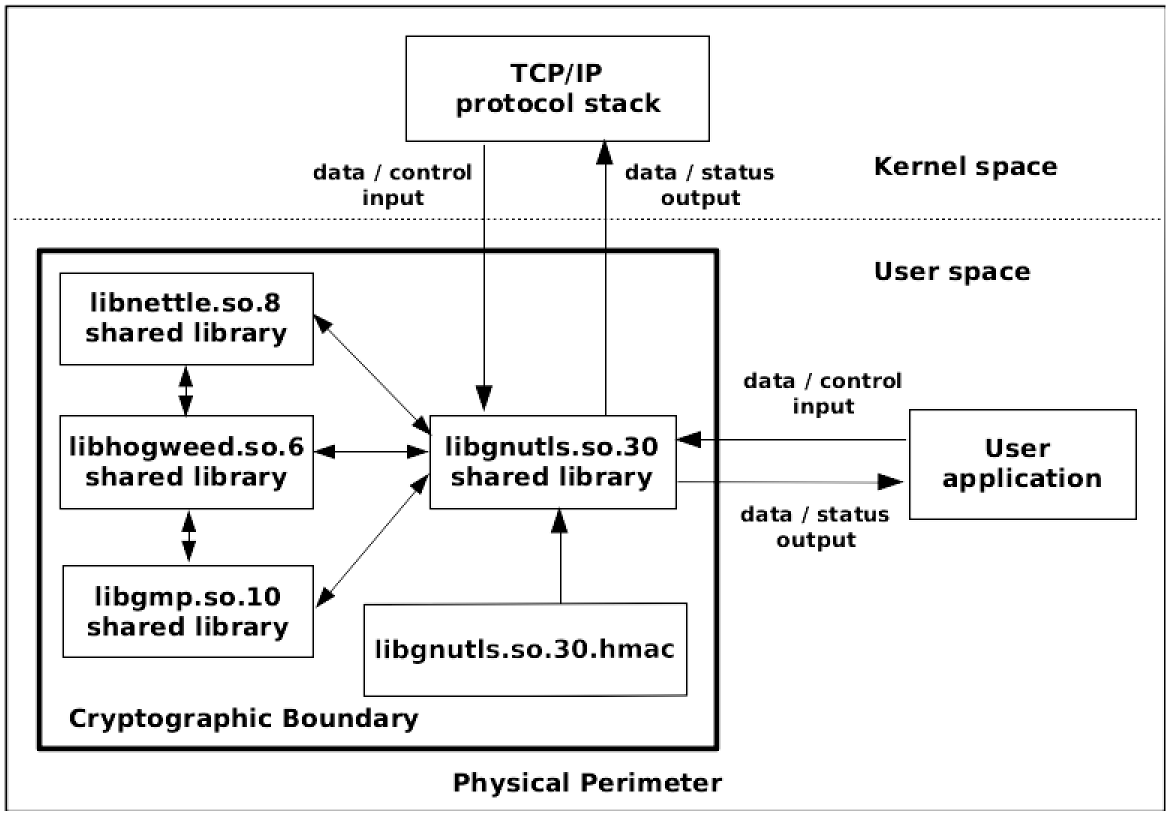

Figure 1: Cryptographic Boundary

2.2 Tested and Vendor Affirmed Module Version and Identification

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libgnutls.so.30; libnettle.so.8; libhogweed.so.6; libgmp.so.10 (statically linked to libgnutls); libgnutls.so.30.hmac on ORACLE SERVER X9-2c with Intel(R) Xeon(R) Platinum 8358	3.7.6-39e2433a29b33b55	N/A	HMAC-SHA-256
libgnutls.so.30; libnettle.so.8; libhogweed.so.6; libgmp.so.10 (statically linked to libgnutls); libgnutls.so.30.hmac on ORACLE SERVER E4-2c with AMD EPYC 7J13	3.7.6-39e2433a29b33b55	N/A	HMAC-SHA-256
libgnutls.so.30; libnettle.so.8; libhogweed.so.6; libgmp.so.10 (statically linked to libgnutls); libgnutls.so.30.hmac on ORACLE SERVER A1-2c with Ampere(R) Altra(R) Q80-30	3.7.6-39e2433a29b33b55	N/A	HMAC-SHA-256

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

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Oracle Linux 9	ORACLE SERVER X9-2c	Intel(R) Xeon(R) Platinum 8358	Yes	KVM on Oracle Linux 8	3.7.6-39e2433a29b33b55
Oracle Linux 9	ORACLE SERVER E4-2c	AMD EPYC 7J13	Yes	KVM on Oracle Linux 8	3.7.6-39e2433a29b33b55
Oracle Linux 9	ORACLE SERVER A1-2c	Ampere(R) Altra(R) Q80-30	Yes	KVM on Oracle Linux 8	3.7.6-39e2433a29b33b55

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

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

Operating System	Hardware Platform
Oracle Linux 9	Oracle X Series Servers
Oracle Linux 9	Oracle E Series Servers
Oracle Linux 9	Oracle A Series Servers
Oracle Linux 9	Marvell T93 LiquidIO III (ARM v8.x) SmartNIC
Oracle Linux 9	Pensando DSC-200-R (ARM v8.x) SmartNIC
Oracle Linux 9	Marvell Liquid IO II (MIPS64) SmartNIC
Oracle Linux 9	Nvidia Bluefield-3 (ARM v8.x) SmartNIC

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

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

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

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved mode	Automatically entered whenever an approved service is requested	Approved	Equivalent to the indicator of the requested service as defined in section 4.3
Non-approved mode	Automatically entered whenever a non-approved service is requested	Non-Approved	Equivalent to the indicator of the requested service as defined in section 4.3

Table 5: Modes List and Description

Mode Change Instructions and Status:

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

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A4743	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A4744	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A4745	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A4746	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A4751	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A4755	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A4743	Key Length - 128, 256	SP 800-38C
AES-CCM	A4755	Key Length - 128, 256	SP 800-38C
AES-CFB8	A4748	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A4749	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A4754	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A4743	Direction - Generation, Verification Key Length - 128, 256	SP 800-38B
AES-CMAC	A4746	Direction - Generation, Verification Key Length - 128, 256	SP 800-38B

Algorithm	CAVP Cert	Properties	Reference
AES-CMAC	A4751	Direction - Generation, Verification Key Length - 128, 256	SP 800-38B
AES-ECB	A4751	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A4743	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-GCM	A4744	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-GCM	A4745	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-GCM	A4746	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-GCM	A4751	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-GCM	A4755	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-GMAC	A4751	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-XTS Testing Revision 2.0	A4752	-	SP 800-38E
Counter DRBG	A4751	Prediction Resistance - No Mode - AES-256 Derivation Function Enabled - No	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-4)	A4751	Curve - P-256, P-384, P-521	FIPS 186-4
ECDSA KeyVer (FIPS186-4)	A4751	Curve - P-256, P-384, P-521	FIPS 186-4
ECDSA SigGen (FIPS186-4)	A4751	Component - No Curve - P-256, P-384, P-521	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A4751	Component - No Curve - P-256, P-384, P-521	FIPS 186-4
HMAC-SHA-1	A4746	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA-1	A4751	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA-1	A4755	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A4746	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A4751	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A4755	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A4746	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A4751	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A4755	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A4746	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1

Algorithm	CAVP Cert	Properties	Reference
HMAC-SHA2-384	A4751	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A4755	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A4746	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A4751	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A4755	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A4751	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	A4751	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	A4750	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	A4751	Iteration Count - Iteration Count: 1000-10000 Increment 1 Password Length - Password Length: 8-128 Increment 1	SP 800-132
RSA KeyGen (FIPS186-4)	A4751	Key Generation Mode - B.3.2 Modulo - 2048, 3072, 4096 Primality Tests - Table C.2 Private Key Format - Standard	FIPS 186-4
RSA SigGen (FIPS186-4)	A4751	Signature Type - PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-4)	A4751	Signature Type - PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096	FIPS 186-4
Safe Primes Key Generation	A4751	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	A4746	Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA-1	A4751	Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA-1	A4755	Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-224	A4746	Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-224	A4751	Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-224	A4755	Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A4746	Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A4751	Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A4755	Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-384	A4746	Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-384	A4751	Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-384	A4755	Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A4746	Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A4751	Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A4755	Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA3-224	A4747	-	FIPS 202
SHA3-224	A4753	-	FIPS 202
SHA3-256	A4747	-	FIPS 202
SHA3-256	A4753	-	FIPS 202
SHA3-384	A4747	-	FIPS 202
SHA3-384	A4753	-	FIPS 202
SHA3-512	A4747	-	FIPS 202
SHA3-512	A4753	-	FIPS 202

Algorithm	CAVP Cert	Properties	Reference
TLS v1.2 KDF RFC7627 (CVL)	A4751	-	SP 800-135 Rev. 1

Table 6: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG	Key Type:Symmetric and Asymmetric RSA (asymmetric):2048, 3072, 4096 bits with 112, 128, 149 bits of key strength. ECDSA (asymmetric):P-224, P-256, P 384, P-521 elliptic curves with 112-256 bits of key strength Safe Primes (asymmetric):ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 2048, 3072, 4096, 6144, 8192-bit keys (112-200 bits of key strength) CTR_DRBG (symmetric):112-256 bit keys (112-256 bits of strength)	N/A	SP 800-133r2 section 4 example 1

Table 7: Vendor-Affirmed Algorithms

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
Blowfish	Symmetric Encryption; Symmetric Decryption
Camellia	Symmetric Encryption; Symmetric Decryption
CAST	Symmetric Encryption; Symmetric Decryption
ChaCha20	Symmetric Encryption; Symmetric Decryption
Chacha20 and Poly1305	Authenticated Encryption; Authenticated Decryption
CMAC with Triple-DES	Message Authentication Code (MAC)
DES	Symmetric Encryption; Symmetric Decryption
Diffie-Hellman using keys generated with domain parameters other than safe primes	Shared Secret Computation
DSA	Key Generation; Domain Parameter Generation; Digital Signature Generation; Digital Signature Verification
ECDSA with curves not listed in the Approved Algorithms Table	Key Generation; Public Key Verification
ECDSA with curves/hash functions not listed in the Approved Algorithms Table	Digital Signature Generation; Digital Signature Verification
EC Diffie-Hellman with curves not listed in the Approved Algorithms Table	Shared Secret Computation
GMAC with keys not listed in the Approved Algorithms Table	Message Authentication Code (MAC)
GOST	Symmetric Encryption; Symmetric Decryption; Message Digest
HMAC with keys smaller than 112-bit	Message Authentication Code (MAC)
HMAC with GOST	Message Authentication Code (MAC)
MD2, MD4, MD5	Message Digest; Message Authentication Code (MAC)
PBKDF with non-approved message digest algorithms or using input parameters not meeting requirements stated in section 2.7	Key Derivation
RC2, RC4	Symmetric Encryption; Symmetric Decryption
RMD160	Message Digest; Message Authentication Code (MAC)
RSA with keys smaller than 2048 bits.	Key Generation
RSA with keys smaller than 2048 bits and/or hash functions not listed in the Approved Algorithms Table	Digital Signature Generation; Digital Signature Verification
RSA encryption and decryption with any key sizes	Key Encapsulation; Key Un-encapsulation
Salsa20	Symmetric Encryption; Symmetric Decryption
SEED	Symmetric Encryption; Symmetric Decryption

Name	Use and Function
Serpent	Symmetric Encryption; Symmetric Decryption
SRP	Key Agreement
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
DRBG generation of keys smaller than 112 bits	Random Number Generation
Non-supported cipher suites (see Appendix A for the complete list of valid cipher suites)	Transport Layer Security (TLS) network protocol
AES GCM with keys not listed in the Approved Algorithms Table	Authenticated Encryption; Authenticated Decryption

Table 8: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
KAS-ECC-SSC	KAS-SSC	Shared Secret Computation	Curves:P-224, P-256, P-384, P-521 elliptic curves with 112-256 bits of key strength Compliance: Compliant with IG D.F scenario 2(1)	KAS-ECC-SSC Sp800-56Ar3: (A4751)
KAS-FFC-SSC	KAS-SSC	Shared Secret Computation	Keys:2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of key strength Compliance:Compliant with IG D.F scenario 2(1)	KAS-FFC-SSC Sp800-56Ar3: (A4751)
AES CBC with HMAC	KTS-Wrap	Key Wrapping, Key Unwrapping	Keys:128, 192, 256 bits with 128-256 bits of key strength Compliance:Compliant with IG D.G	AES-CBC: (A4743, A4744, A4745, A4746, A4751, A4755) HMAC-SHA-1: (A4746, A4751, A4755) HMAC-SHA2-224: (A4746, A4751, A4755) HMAC-SHA2-256: (A4746, A4751, A4755) HMAC-SHA2-384: (A4746, A4751, A4755) HMAC-SHA2-512: (A4746, A4751, A4755)
AES CCM (Key Wrapping/Unwrapping)	KTS-Wrap	Key Wrapping, Key Unwrapping	Keys:128 and 256 bits with 128 and 256 bits of key strength Compliance:Compliant with IG D.G	AES-CCM: (A4743, A4755)
AES GCM (Key Wrapping/Unwrapping)	KTS-Wrap	Key Wrapping, Key Unwrapping	Keys:128 and 256 bits with 128 and 256 bits of key strength Compliance:Compliant with IG D.G	AES-GCM: (A4743, A4744, A4745, A4746, A4751, A4755)
AES CCM (Authenticated Encryption/Decryption)	BC-Auth	Authenticated Encryption/Decryption	Keys:128 and 256 bits with 128 and 256 bits of key strength	AES-CCM: (A4743, A4755)

Name	Type	Description	Properties	Algorithms
AES GCM (Authenticated Encryption/Decryption)	BC-Auth	Authenticated Encryption/Decryption	Keys:128 and 256 bits with 128 and 256 bits of key strength	AES-GCM: (A4743, A4744, A4745, A4746, A4751, A4755)
AES-CBC	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CBC: (A4743, A4744, A4745, A4746, A4751, A4755)
AES-CMAC	MAC	Message authentication code (MAC)	Keys:128 and 256 bits with 128 and 256 bits of key strength	AES-CMAC: (A4743, A4746, A4751)
HMAC	MAC	Message authentication code (MAC)	Keys:112-524288 bits with 112-256 bits of key strength	HMAC-SHA-1: (A4746, A4751, A4755) HMAC-SHA2-224: (A4746, A4751, A4755) HMAC-SHA2-256: (A4746, A4751, A4755) HMAC-SHA2-384: (A4746, A4751, A4755) HMAC-SHA2-512: (A4746, A4751, A4755)
Hashes	SHA	Hashing		SHA-1: (A4746, A4751, A4755) SHA2-224: (A4746, A4751, A4755) SHA2-256: (A4746, A4751, A4755) SHA2-384: (A4746, A4751, A4755) SHA2-512: (A4746, A4751, A4755) SHA3-224: (A4747, A4753) SHA3-256: (A4747, A4753) SHA3-384: (A4747, A4753) SHA3-512: (A4747, A4753)
AES-CFB8	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CFB8: (A4748, A4749, A4754)
AES-XTS	BC-UnAuth	Encryption/Decryption	Keys:128, 256 bits with 128 and 256 bits of key strength	AES-XTS Testing Revision 2.0: (A4752)
AES-GMAC	MAC	Message authentication code (MAC)	Keys:128 and 256 bits with 128 and 256 bits of key strength	AES-GMAC: (A4751)
Counter DRBG	DRBG	Random Number Generation	Compliance:Compliant with SP800-90ARev1	Counter DRBG: (A4751)
ECDSA Signature Generation	DigSig-SigGen	Signature Generation	Curves: P-224, P-256, P-384, P-521 Hashes:SHA2-224, SHA2-256, SHA2-384, SHA2-512	ECDSA SigGen (FIPS186-4): (A4751)
ECDSA Key Generation	CKG	Key Generation	Curves:P-224, P-256, P-384, P-521	ECDSA KeyGen (FIPS186-4): (A4751)
ECDSA Signature Verification	DigSig-SigVer	Signature Verification	Curves: P-224, P-256, P-384, P-521 Hashes:SHA2-224, SHA2-256, SHA2-384, SHA2-512	ECDSA SigVer (FIPS186-4): (A4751)
ECDSA Key Verification	AsymKeyPair-KeyVer	Key Verification	Curves:P-224, P-256, P-384, P-521	ECDSA KeyVer (FIPS186-4): (A4751)
RSA Signature Generation	DigSig-SigGen	Signature Generation	Keys:2048-16384 bits Hashes:SHA2-224, SHA2-256, SHA2-384, SHA2-512	RSA SigGen (FIPS186-4): (A4751)

Name	Type	Description	Properties	Algorithms
RSA Key Generation	CKG	Key Generation	Keys:2048-15360 bits	RSA KeyGen (FIPS186-4): (A4751)
RSA Signature Verification	DigSig-SigVer	Signature Verification	Keys:1024-16384 bits Hashes:SHA2-224, SHA2-256, SHA2-384, SHA2-512	RSA SigVer (FIPS186-4): (A4751)
Safe Primes Key Generation	CKG	Key Generation	Groups:MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	Safe Primes Key Generation: (A4751)
HKDF Key Derivation	KAS-56CKDF	Key Derivation	HKDF derived key:112-256 bits with 112-256 bits of key strength	KDA HKDF Sp800-56Cr1: (A4750)
Password-based Key Derivation	PBKDF	Key Derivation	PBKDF Derived key:112-4096 bits with 112-256 bits of key strength	PBKDF: (A4751)
TLS 1.2 Key Derivation	KAS-135KDF	Key Derivation	Derived secret::112-256 bits with112-256 bits of key strength	TLS v1.2 KDF RFC7627: (A4751)
AES-ECB	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits	AES-ECB: (A4751)
TLS Handshake	KAS-Full	Key Agreement	Curves:P-224, P-256, P-384, P-521 elliptic curves with 112-256 bits of key strength Keys:2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of key strength Compliance:Compliant with IG D.F scenario 2(2)	KAS-ECC-SSC Sp800-56Ar3: (A4751) KAS-FFC-SSC Sp800-56Ar3: (A4751) KDA HKDF Sp800-56Cr1: (A4750) TLS v1.2 KDF RFC7627: (A4751)
Symmetric Key Generation with Counter DRBG	CKG	Symmetric Key Generation	Keys:112-256 bits with 112-256 bits of key strength Compliance:SP 800-133r2 section 6.1	Counter DRBG: (A4751)

Table 9: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 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 design of the TLS protocol implicitly ensures that the counter (the nonce_explicit part of the IV) does not exhaust the maximum number of possible values for a given session key.

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

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

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.

The IV generated in both TLS 1.2 and TLS 1.3 scenarios is only used within the context of the TLS protocol implementation.

2.7.2 Key Derivation using SP 800-132 PBKDF2

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

In accordance with [SP800-132], the module ensures that the following requirements are met when running the PBKDF approved service.

- The length of the MK or DPK is 112 bits or more.
- A portion of the salt, with a length of at least 128 bits (it shall be generated randomly using the [SP800-90Ar1] DRBG).
- The minimum value of the iteration count is 1000 (the iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users).
- The 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}.

Passwords or passphrases, used as an input for the PBKDF, shall not be used as cryptographic keys. Derived keys shall only be used in storage applications. The Master Key (MK) shall not be used for other purposes. The calling application shall also observe the rest of the requirements and recommendations specified in [SP800-132].

2.7.3 AES XTS

The length of a single data unit encrypted or decrypted with AES XTS shall not exceed 220 AES blocks, that is 16MB, of data per XTS instance. An XTS instance is defined in Section 4 of SP 800-38E.

To meet the requirement stated in IG C.I, the module implements a check to ensure that the two AES keys used in AES XTS mode are not identical.

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

2.7.4 SP 800-56Ar3 Assurances

The module offers DH and ECDH shared secret computation services compliant to the SP 800-56Ar3. To meet the required assurances listed in section 5.6 of SP 800-56Ar3, the module shall be used together with an application that implements the “TLS protocol” and the following steps shall be performed.

- 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 800-56Ar3.
- As part of the module's shared secret computation (SSC) service, the module internally performs the public key validation on the peer's public key passed in as input to the SSC function. This meets the public key validity assurance required by the sections 5.6.2.2.1/5.6.2.2.2 of SP 800-56Ar3.
- 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
E99	Oracle Corporation

Table 10: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Oracle User Space CPU Time Jitter RNG Entropy Source	Non-Physical	Oracle Linux 9 on KVM on Oracle Linux 8 on AMD AMD EPYC(TM) 7001 Series AMD EPYC 7J13; Ampere Ampere(R) Altra(R) Ampere(R) Altra(R) Q80-30; Intel Ice Lake Intel(R) Xeon(R) Platinum 8358	256 bits	256 bits	AES-256 CTR DRBG (CAVP cert #A4751)

Table 11: Entropy Sources

RNG Information:

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

The DRBG supports the CTR_DRBG with AES-256, without a derivation function and without prediction resistance. The module uses an [SP800-90B]-compliant entropy source specified in the Entropy Source table. This entropy source is located within the physical perimeter, but outside of the cryptographic boundary of the module. The module obtains 384 bits to seed the DRBG, and 256 bits to reseed it, sufficient to provide a DRBG with 256 bits of security strength.

2.9 Key Generation

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

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

2.10 Key Establishment

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

2.11 Industry Protocols

The module implements KDF for the TLS protocol TLSv1.2.

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

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

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

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

Table 12: Ports and Interfaces

The logical interfaces are the APIs through which the applications request services. These logical interfaces are logically separated from each other by the API design.

The module does not implement a control output interface.

3.2 Trusted Channel Specification

The module does not implement a trusted channel.

3.3 Control Interface Not Inhibited

The module does not implement a control output interface.

4 Roles, Services, and Authentication

4.1 Authentication Methods

The module does not implement authentication for roles.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 13: Roles

The module supports the Crypto Officer role only. This sole role is implicitly and always assumed by the operator of the module. The module does not support multiple concurrent operators.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Message Digest	Compute a message digest	GNUTLS_FIPS140_OP_APPROVED	Message	Digest value	Hashes	Crypto Officer
Encryption	Encrypt a plaintext	GNUTLS_FIPS140_OP_APPROVED	AES Key, plaintext	Ciphertext	AES-CBC AES-CFB8 AES-XTS AES-ECB	Crypto Officer - AES Key: W,E
Decryption	Decrypt a ciphertext	GNUTLS_FIPS140_OP_APPROVED	AES Key, ciphertext	Plaintext	AES-CBC AES-CFB8 AES-XTS AES-ECB	Crypto Officer - AES Key: W,E
Authenticated Decryption	Authenticated Decryption	GNUTLS_FIPS140_OP_APPROVED	AES key, IV, MAC tag, ciphertext	Plaintext or failure	AES CCM (Authenticated Encryption/Decryption) AES GCM (Authenticated Encryption/Decryption)	Crypto Officer - AES Key: W,E
Authenticated Encryption	Authenticated Encryption	GNUTLS_FIPS140_OP_APPROVED	AES Key, IV, plaintext	Ciphertext, MAC tag	AES CCM (Authenticated Encryption/Decryption) AES GCM (Authenticated Encryption/Decryption)	Crypto Officer - AES Key: W,E
AES Message Authentication	Message Authentication	GNUTLS_FIPS140_OP_APPROVED	AES Key, message	MAC tag	AES-CMAC AES-GMAC	Crypto Officer - AES Key: W,E
HMAC Message Authentication	Message Authentication	GNUTLS_FIPS140_OP_APPROVED	HMAC Key, message	MAC tag	HMAC	Crypto Officer - HMAC Key: W,E
ECDH Shared Secret Computation	Compute a shared secret	GNUTLS_FIPS140_OP_APPROVED	EC Private Key, EC Public Key	Shared secret	KAS-ECC-SSC	Crypto Officer - EC Private Key: W,E - EC Public Key: W,E - Shared Secret: G,R
Key Derivation	Derive a key	GNUTLS_FIPS140_OP_APPROVED	Shared Secret	HKDF derived key	HKDF Key Derivation TLS 1.2 Key Derivation	Crypto Officer - Shared Secret: W,E - TLS Pre-Master Secret: W,E - TLS Master Secret: W,E - TLS Derived

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Secret: G,R - HKDF Derived Key: G,R
Password-Based Key Derivation	Derive a key from a password	GNUTLS_FIPS140_OP_APPROVED	Password	PBKDF derived key	Password-based Key Derivation	Crypto Officer - Password: W,E - PBKDF Derived Key: G,R
DH Key Pair Generation	Key Pair Generation	GNUTLS_FIPS140_OP_APPROVED	DH Group	Module generated DH private key, Module generated DH public key	Safe Primes Key Generation	Crypto Officer - Module Generated DH Public Key: G,R - Module Generated DH Private Key: G,R - Intermediate Key Generation Value: G
EC Key Pair Generation	Key Pair Generation	GNUTLS_FIPS140_OP_APPROVED	Curve	Module generated EC private key, Module generated EC public key	ECDSA Key Generation	Crypto Officer - Module Generated EC Public Key: G,R - Module Generated EC Private Key: G,R - Intermediate Key Generation Value: G
RSA Key Pair Generation	Key Pair Generation	GNUTLS_FIPS140_OP_APPROVED	Modulus	Module generated RSA private key, Module generated RSA public key	RSA Key Generation	Crypto Officer - Module Generated RSA Private Key: G,R - Module Generated RSA Public Key: G,R - Intermediate Key Generation Value: G,R
Public Key Verification	Verify an EC public key	GNUTLS_FIPS140_OP_APPROVED	EC public key	Return codes/log messages	ECDSA Key Verification	Crypto Officer - EC Private Key: W,E - EC Public Key: W,E
Key Wrapping	Wrap a key	GNUTLS_FIPS140_OP_APPROVED	AES Key, key to be wrapped	Wrapped key	AES CCM (Key Wrapping/Unwrapping) AES CBC with HMAC AES GCM (Key Wrapping/Unwrapping)	Crypto Officer - AES Key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Key Unwrapping	Unwrap a key	GNUTLS_FIPS140_OP_APPROVED	AES Key, key to be unwrapped	Unwrapped key	AES CCM (Key Wrapping/Unwrapping) AES CBC with HMAC AES GCM (Key Wrapping/Unwrapping)	Crypto Officer - AES Key: W,E
Random Number Generation	Generate random bytes	GNUTLS_FIPS140_OP_APPROVED	Output length	Random bytes	Counter DRBG	Crypto Officer - Entropy Input: W,E - DRBG Seed: G,E - Internal State (V, Key): G,E
Signature Verification	Verify a digital signature	GNUTLS_FIPS140_OP_APPROVED	Message, EC Public Key or RSA Public Key, signature, hash algorithm	Pass/fail	ECDSA Signature Verification RSA Signature Verification	Crypto Officer - EC Public Key: W,E - RSA Public Key: W,E
Signature Generation	Signature Generation	GNUTLS_FIPS140_OP_APPROVED	Message, EC Private Key or RSA Private Key, hash algorithm	Signature	ECDSA Signature Generation RSA Signature Generation	Crypto Officer - EC Private Key: W,E - RSA Private Key: W,E
Show Version	Return the module name and version information	None	N/A	Module name and version	None	Crypto Officer
Show Status	Return the module status	None	N/A	Module status	None	Crypto Officer
Self-Test	Perform the CASTs and integrity tests	None	N/A	Pass/Fail	KAS-ECC-SSC KAS-FFC-SSC AES CCM (Key Wrapping/Unwrapping) AES GCM (Key Wrapping/Unwrapping) AES-CBC AES-CMAC HMAC Hashes AES-CFB8 AES-XTS AES-GMAC Counter DRBG ECDSA Signature Generation ECDSA Key Generation ECDSA Signature Verification RSA Signature Generation RSA Key Generation RSA Signature Verification Safe Primes Key Generation HKDF Key Derivation Password-based Key Derivation TLS 1.2 Key Derivation AES-ECB AES CCM (Authenticated Encryption/Decryption)	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					AES GCM (Authenticated Encryption/Decryption)	
Zeroization	Zeroize all SSPs	None	Any SSP	N/A	None	Crypto Officer - Module-generated AES Key: Z - AES Key: Z - Module-generated HMAC Key: Z - HMAC Key: Z - Shared Secret: Z - Password: Z - Entropy Input: Z - DRBG Seed: Z - Internal State (V, Key): Z - DH Public Key: Z - DH Private Key: Z - Module Generated DH Public Key: Z - Module Generated DH Private Key: Z - EC Private Key: Z - EC Public Key: Z - Module Generated EC Private Key: Z - Module Generated EC Public Key: Z - RSA Private Key: Z - RSA Public Key: Z - Module Generated RSA Private Key: Z - Module Generated RSA Public Key: Z - Intermediate Key Generation Value: Z - TLS Pre-Master Secret: Z - TLS Master Secret: Z - TLS Derived Secret: Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- PBKDF Derived Key: Z - HKDF Derived Key: Z
Transport Layer Security (TLS) Network Protocol	Provide supported cipher suites in approved mode	GNUTLS_FIPS140_OP_APPROVED	Cipher-suites, Digital Certificate, Public and Private Keys, Application Data	Return codes and/or log messages, Application data	AES CBC with HMAC AES-CBC Hashes ECDSA Signature Generation ECDSA Key Generation ECDSA Signature Verification ECDSA Key Verification RSA Signature Generation RSA Signature Verification Safe Primes Key Generation TLS Handshake AES CCM (Authenticated Encryption/Decryption) AES GCM (Authenticated Encryption/Decryption)	Crypto Officer - AES Key: W,E - HMAC Key: W,E - RSA Public Key: W,E - RSA Private Key: W,E - EC Public Key: W,E - EC Private Key: W,E - Module Generated DH Public Key: G,E - Module Generated DH Private Key: G,E - Module Generated EC Private Key: G,E - Module Generated EC Public Key: G,E - TLS Master Secret: G,E - TLS Pre-Master Secret: G,E - TLS Derived Secret: G,R - HKDF Derived Key: G,R
Symmetric Key Generation	Generate a key	GNUTLS_FIPS140_OP_APPROVED	N/A	Module generated AES key, Module generated HMAC key	Symmetric Key Generation with Counter DRBG	Crypto Officer - Module-generated AES Key: G,R - Module-generated HMAC Key: G,R
DH Shared Secret Computation	Compute a shared secret	GNUTLS_FIPS140_OP_APPROVED	DH Public Key, DH Private Key	Shared secret	KAS-FFC-SSC	Crypto Officer - DH Public Key: W,E - DH Private Key: W,E - Shared Secret: G,R

Table 14: Approved Services

The following convention is used to specify access rights to SSPs:

- **Generate (G):** The module generates or derives the SSP.

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

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

4.4 Non-Approved Services

Name	Description	Algorithms	Role
Symmetric Key Generation	Generate symmetric key other than AES and HMAC keys	DRBG generation of keys smaller than 112 bits	CO
Symmetric Encryption/Decryption	Compute the cipher for encryption and decryption	Blowfish Camellia CAST ChaCha20 DES GOST RC2, RC4 Salsa20 SEED Serpent Triple-DES Twofish	CO
Asymmetric Key Generation	Generate RSA, DSA, and ECDSA key pairs	DSA ECDSA with curves not listed in the Approved Algorithms Table RSA with keys smaller than 2048 bits.	CO
Digital Signature Generation	Sign RSA, DSA, and ECDSA signatures	DSA ECDSA with curves/hash functions not listed in the Approved Algorithms Table RSA with keys smaller than 2048 bits and/or hash functions not listed in the Approved Algorithms Table	CO
Digital Signature Verification	Verify RSA, DSA, and ECDSA signatures	DSA ECDSA with curves/hash functions not listed in the Approved Algorithms Table RSA with keys smaller than 2048 bits and/or hash functions not listed in the Approved Algorithms Table	CO
Message Digest	Compute message digest	MD2, MD4, MD5 RMD160 STREEBOG	CO
Message Authentication Code (MAC)	Compute MAC	CMAC with Triple-DES GMAC with keys not listed in the Approved Algorithms Table HMAC with keys smaller than 112-bit HMAC with GOST UMAC	CO
Key Encapsulation/Un-encapsulation	Perform RSA key encapsulation/un-encapsulation	RSA encryption and decryption with any key sizes	CO
Key Derivation	Perform key derivation	PBKDF with non-approved message digest algorithms or using input parameters not meeting requirements stated in section 2.7	CO
Transport Layer Security (TLS) network protocol	Provide non-supported cipher suites	Non-supported cipher suites (see Appendix A for the complete list of valid cipher suites)	CO
Random Number Generation	Generate random numbers	Yarrow DRBG generation of keys smaller than 112 bits	CO
Key Agreement	Perform key agreement	SRP	CO

Name	Description	Algorithms	Role
Authenticated Encryption/Decryption	Perform authenticated encryption or decryption	Chacha20 and Poly1305 AES GCM with keys not listed in the Approved Algorithms Table	CO
Shared Secret Computation	Perform shared secret computation	Diffie-Hellman using keys generated with domain parameters other than safe primes EC Diffie-Hellman with curves not listed in the Approved Algorithms Table	CO
Public Key Verification	Verify ECDSA public keys	ECDSA with curves not listed in the Approved Algorithms Table	CO

Table 15: Non-Approved Services

4.5 External Software/Firmware Loaded

The module does not load external software or firmware.

5 Software/Firmware Security

5.1 Integrity Techniques

The integrity of the module is verified by comparing 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 the software components of the module listed in section 2. If the HMAC values do not match, the test fails, and the module enters the error state.

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

6.2 Configuration Settings and Restrictions

The module shall be installed as stated in Section 11.1. There are no concurrent operators.

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

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

7 Physical Security

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

8 Non-Invasive Security

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

9 Sensitive Security Parameters Management

9.1 Storage Areas

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

Table 16: Storage Areas

9.2 SSP Input-Output Methods

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

Table 17: SSP Input-Output Methods

The module does not support entry and output of SSPs beyond the physical perimeter of the operational environment. The SSPs are provided to the module via API input parameters in the plaintext form and output via API output parameters in the plaintext form within the physical perimeter of the operational environment. This is allowed by [FIPS140-3_IG] IG 9.5.A.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Free Cipher Handle	Zeroizes the SSPs referenced. In the function name	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable.	By calling the appropriate zeroization functions: AES Key: gnutls_cipher_deinit(); AES Key: gnutls_aead_cipher_deinit(); HMAC Key: gnutls_hmac_deinit(); RSA Public and Private Keys: gnutls_privkey_deinit(), gnutls_x509_privkey_deinit(), gnutls_rsa_params_deinit(); ECDSA Public and Private Keys: gnutls_privkey_deinit(), gnutls_x509_privkey_deinit(), gnutls_rsa_params_deinit(); Diffie-Hellman Public and Private Keys: gnutls_dh_params_deinit(); TLS Pre-master Secret: gnutls_deinit(); TLS Master Secret: gnutls_deinit(); TLS Derived Secret: gnutls_deinit(); Diffie-Hellman Public and Private Keys: gnutls_pk_params_clear(); EC Diffie-Hellman Public and Private Keys: gnutls_pk_params_clear(); Diffie-Hellman Shared Secret: zeroize key(); EC Diffie-Hellman Shared Secret: zeroize key(); All SSPs: gnutls_global_deinit()
Module Reset	De-allocates the volatile memory used to store SSPs	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable	By unloading and reloading the module.
Automatic	Automatically zeroized by the module when no longer needed	Automatically zeroized by the module when no longer needed	N/A

Table 18: SSP Zeroization Methods

All data output is inhibited during zeroization. Once the zeroization is started, all data output via the data output interface is inhibited until the zeroization is completed successfully.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Module-generated AES Key	AES key generated during Symmetric Key Generation	128, 192, 256 bits - 128, 192, 256 bits	Symmetric key - CSP	Symmetric Key Generation with Counter DRBG		
AES Key	AES key used for encryption, decryption, authenticated encryption, authenticated decryption and computing MAC tags	128, 192, 256 bits - 128, 192, 256 bits	Symmetric key - CSP			AES CCM (Key Wrapping/Unwrapping) AES CBC with HMAC AES GCM (Key Wrapping/Unwrapping) AES-CBC AES-CMAC AES-CFB8 AES-XTS AES-GMAC AES CCM (Authenticated Encryption/Decryption) AES GCM (Authenticated Encryption/Decryption)
Module-generated HMAC Key	HMAC key generated during Symmetric Key Generation	HMAC key generated during Symmetric Key Generation - 112-256 bits	Authentication key - CSP	Symmetric Key Generation with Counter DRBG		
HMAC Key	HMAC Key	112-524288 bits - 112-256 bits	Authentication key - CSP			HMAC
Shared Secret	Shared secret generated by DH/ECDH	224-8192 bits - 112-256 bits	Shared secret - CSP		KAS-ECC-SSC KAS-FFC-SSC	HKDF Key Derivation TLS 1.2 Key Derivation KDA HKDF Sp800-56Cr1 (A4750)
Password	PBKDF password	112-256 bits - N/A	Password - CSP			Password-based Key Derivation
PBKDF Derived Key	PBKDF2 derived key	112-4096 bits - 112-256 bits	Derived Key - CSP	Password-based Key Derivation		
Entropy Input	Entropy input used to seed the DRBGs	128-448 bits - 128-256 bits	Entropy - CSP			Counter DRBG
DRBG Seed	DRBG seed derived from entropy input as defined in SP 800-90Ar1	128, 192, 256 bits - 128-256 bits	SEED - CSP	Counter DRBG		Counter DRBG
Internal State (V, Key)	Internal state of CTR_DRBG	128, 192, 256 bits - 128-256 bits	DRBG Internal state - CSP	Counter DRBG		Counter DRBG
DH Public Key	Public key used for DH	2048, 3072, 4096, 6144, 8192 bits - 112-200 bits	Public key - PSP			KAS-FFC-SSC
DH Private Key	Private key used for DH	2048, 3072, 4096, 6144, 8192 bits - 112-200 bits	Private key - CSP			KAS-FFC-SSC
Module Generated DH Public Key	DH public key generated by the module	2048, 3072, 4096, 6144, 8192 bits - 112-200 bits	Public key - PSP	Safe Primes Key Generation		TLS Handshake
Module Generated DH Private Key	DH private key generated by the module	2048, 3072, 4096, 6144, 8192 bits - 112-200 bits	Private key - CSP	Safe Primes Key Generation		TLS Handshake
EC Private Key	Private key used for ECDSA signature generation and Shared Secret Computation	P-224, P-256, P-384, P-521 - 128-256 bits	Private key - CSP			KAS-ECC-SSC

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
EC Public Key	Public key used for ECDSA signature verification primitive and Shared Secret Computation	P-224, P-256, P-384, P-521 - 128-256 bits	Public key - PSP			KAS-ECC-SSC
Module Generated EC Private Key	EC private key generated by the module	P-224, P-256, P-384, P-521 - 128-256 bits	Private key - CSP	ECDSA Key Generation		TLS Handshake
Module Generated EC Public Key	EC public key generated by the module	P-224, P-256, P-384, P-521 - 128-256 bits	Public key - PSP	ECDSA Key Generation		TLS Handshake
RSA Private Key	Private key used for RSA signature generation	2048, 3072, 4096 bits - 112, 128, 150 bits	Private key - CSP			RSA Signature Generation
RSA Public Key	Public key used for RSA signature verification	1024, 2048, 3072, 4096 bits - 80, 112, 128, 150 bits	Public key - PSP			RSA Signature Verification
Module Generated RSA Private Key	RSA private key generated by the module	2048, 3072, 4096 bits - 112, 128, 150 bits	Private key - CSP	RSA Key Generation		
Module Generated RSA Public Key	RSA public key generated by the module	2048, 3072, 4096 bits - 112, 128, 150 bits	Public key - PSP	RSA Key Generation		
TLS Pre-Master Secret	TLS pre-master secret used for deriving the TLS master secret	112-256 bits - N/A	TLS Pre-master secret - CSP		KAS-ECC-SSC KAS-FFC-SSC	TLS Handshake
TLS Master Secret	TLS master secret used for deriving the TLS derived secret	112-256 bits - N/A	TLS Master secret - CSP	TLS 1.2 Key Derivation		TLS Handshake
TLS Derived Secret	TLS derived secret, derived from TLS master secret	112-256 bits - 112-256 bits	Symmetric key - CSP	TLS 1.2 Key Derivation		TLS Handshake
Intermediate Key Generation Value	Intermediate key generation value	224-4096 bits - 112-256 bits	Intermediate value - CSP	ECDSA Key Generation RSA Key Generation Safe Primes Key Generation		ECDSA Key Generation RSA Key Generation Safe Primes Key Generation
HKDF Derived Key	HKDF derived key	112-256 - 112-256 bits	Derived key - CSP	KDA HKDF Sp800-56Cr1 (A4750)		

Table 19: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Module-generated AES Key	API output parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	Internal State (V, Key):Generated from
AES Key	API input parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	
Module-generated HMAC Key	API output parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	Internal State (V, Key):Generated from
HMAC Key	API input parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Shared Secret	API output parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	DH Public Key:Used With DH Private Key:Used With EC Private Key:Used With EC Public Key:Used With
Password	API input parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	PBKDF Derived Key:Derivation of
PBKDF Derived Key	API output parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	Password:Derived From
Entropy Input		RAM:Plaintext	From generation until DRBG seed is created	Automatic	DRBG Seed:Generation of
DRBG Seed		RAM:Plaintext	While the DRBG is being instantiated	Automatic	Entropy Input:Derived From Internal State (V, Key):Generation of
Internal State (V, Key)		RAM:Plaintext	From DRBG instantiation until DRBG termination	Automatic	DRBG Seed:Generated From Module-generated AES Key:Generation Of Module-generated HMAC Key:Generation Of
DH Public Key	API input parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	DH Private Key:Paired With Shared Secret:Generation Of
DH Private Key	API input parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	DH Public Key:Paired With Shared Secret:Generation Of
Module Generated DH Public Key	API output parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	Module Generated DH Private Key:Paired With Intermediate Key Generation Value:Generated From
Module Generated DH Private Key	API output parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	Module Generated DH Public Key:Paired With Intermediate Key Generation Value:Generated From
EC Private Key	API input parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	EC Public Key:Paired With Shared Secret:Generation Of
EC Public Key	API input parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	EC Private Key:Paired With Shared Secret:Generation Of
Module Generated EC Private Key	API output parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	Module Generated EC Public Key:Paired With Intermediate Key Generation Value:Generated From
Module Generated EC Public Key	API output parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	Module Generated EC Private Key:Paired With Intermediate Key Generation Value:Generated From
RSA Private Key	API input parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	RSA Public Key:Paired With
RSA Public Key	API input parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	RSA Private Key:Paired With
Module Generated RSA Private Key	API output parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	Module Generated RSA Public Key:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Module Generated RSA Public Key	API output parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	Module Generated RSA Private Key:Paired With
TLS Pre-Master Secret		RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	TLS Master Secret:Generation Of DH Public Key:Used With DH Private Key:Used With EC Private Key:Used With EC Public Key:Used With
TLS Master Secret		RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	TLS Pre-Master Secret:Derived From TLS Derived Secret:Derivation Of
TLS Derived Secret	API output parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle Module Reset	TLS Master Secret:Derived From
Intermediate Key Generation Value		RAM:Plaintext	For the duration of the service	Automatic	Module Generated DH Public Key:Generation Of Module Generated DH Private Key:Generation Of Module Generated EC Private Key:Generation Of Module Generated EC Public Key:Generation Of Module Generated RSA Private Key:Generation Of Module Generated RSA Public Key:Generation Of
HKDF Derived Key	API output parameters	RAM:Plaintext	For the duration of the service	Free Cipher Handle	Shared Secret:Derived From

Table 20: SSP Table 2

9.5 Transitions

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

The RSA, ECDSA algorithm as implemented by the module conforms to FIPS 186-4, which has been superseded by FIPS 186-5. FIPS 186-4 will be withdrawn on February 3, 2024.

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A4746)	256-bit key	Message Authentication	SW/FW Integrity	Module becomes operational and services are available for use	Integrity test for libgnutls.so.30, libnettle.so.8, libhogweed.so.6, libgmp.so.10
HMAC-SHA2-256 (A4751)	256-bit key	Message Authentication	SW/FW Integrity	Module becomes operational and services are available for use	Integrity test for libgnutls.so.30, libnettle.so.8, libhogweed.so.6, libgmp.so.10
HMAC-SHA2-256 (A4755)	256-bit key	Message Authentication	SW/FW Integrity	Module becomes operational and services are available for use	Integrity test for libgnutls.so.30, libnettle.so.8, libhogweed.so.6, libgmp.so.10

Table 21: Pre-Operational Self-Tests

The pre-operational software integrity test is performed automatically (after the CASTs) when the module is powered on, before the module transitions into the operational state. While the module is executing the self-tests, services are not available, and data output (via the data output interface) is inhibited until the tests are successfully completed. The module transitions to the operational state only after the pre-operational self-test has passed successfully. If the pre-operational self-test fails, the module transitions to the error state.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
ECDSA KeyGen (FIPS186-4) (A4751)	SHA2-256	Signature generation and Signature verification	PCT	Successful key generation	Signature generation and verification	Key pair generation
RSA KeyGen (FIPS186-4) (A4751)	SHA2-256	Signature generation and Signature verification	PCT	Successful key generation	Signature generation and verification	Key pair generation
Safe Primes Key Generation (A4751)	N/A	Diffie-Hellman key generation	PCT	Successful key generation	PCT according to section 5.6.2.1.4 of [SP800-56Ar3]	Key pair generation
SHA3-224 (A4747)	SHA3-224	KAT SHA3-224	CAST	Module is operational and services are available for use	Message Digest	Module initialization
SHA3-224 (A4753)	SHA3-224	KAT SHA3-224	CAST	Module is operational and services are available for use	Message Digest	Module initialization
AES-CBC (A4743)	256-bit keys	Encrypt/Decrypt KAT for CBC	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization
AES-CBC (A4744)	256-bit keys	Encrypt/Decrypt KAT for CBC	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization
AES-CBC (A4745)	256-bit keys	Encrypt/Decrypt KAT for CBC	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC (A4746)	256-bit keys	Encrypt/Decrypt KAT for CBC	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization
AES-CBC (A4751)	256-bit keys	Encrypt/Decrypt KAT for CBC	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization
AES-CBC (A4755)	256-bit keys	Encrypt/Decrypt KAT for CBC	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization
AES-CFB8 (A4748)	256-bit keys	Encrypt/Decrypt KAT for CFB8	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization
AES-CFB8 (A4749)	256-bit keys	Encrypt/Decrypt KAT for CFB8	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization
AES-CFB8 (A4754)	256-bit keys	Encrypt/Decrypt KAT for CFB8	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization
AES-GCM (A4743)	256-bit keys	Encrypt/Decrypt KAT for GCM	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization
AES-GCM (A4744)	256-bit keys	Encrypt/Decrypt KAT for GCM	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization
AES-GCM (A4745)	256-bit keys	Encrypt/Decrypt KAT for GCM	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization
AES-GCM (A4746)	256-bit keys	Encrypt/Decrypt KAT for GCM	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization
AES-GCM (A4751)	256-bit keys	Encrypt/Decrypt KAT for GCM	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization
AES-GCM (A4755)	256-bit keys	Encrypt/Decrypt KAT for GCM	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization
AES-XTS Testing Revision 2.0 (A4752)	256-bit keys	Encrypt/Decrypt KAT for XTS	CAST	Module is operational and services are available for use	Encryption/Decryption	Module initialization
KDA HKDF Sp800-56Cr1 (A4750)	HMAC-SHA2-256	KAT with HMAC-SHA2-256 for HKDF	CAST	Module is operational and services are available for use	Key Derivation	Module initialization
TLS v1.2 KDF RFC7627 (A4751)	HMAC-SHA2-256	KAT with HMAC-SHA2-256 for TLS 1.2 KDF	CAST	Module is operational and	Key Derivation	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		
PBKDF (A4751)	HMAC-SHA2-256	KAT with HMAC-SHA2-256 for PBKDF	CAST	Module is operational and services are available for use	Key Derivation	Module initialization
Counter DRBG (A4751)	256-bit key; Health tests	CTR_DRBG with AES without DF, without PR KAT; SP800-90Ar1 Section 11.3 Health Test	CAST	Module is operational and services are available for use	KAT CTR_DRBG with AES with 256-bit keys without DF, without PR; Health tests	Module initialization
KAS-FFC-SSC Sp800-56Ar3 (A4751)	3072-bit key	Primitive "Z" computation KAT	CAST	Module is operational and services are available for use	Shared Secret Computation	Module initialization
KAS-ECC-SSC Sp800-56Ar3 (A4751)	Curve P-256	Primitive "Z" computation KAT	CAST	Module is operational and services are available for use	Shared Secret Computation	Module initialization
ECDSA SigGen (FIPS186-4) (A4751)	Curve P-256 and SHA2-256	KAT ECDSA with P-256 using SHA2-256	CAST	Module is operational and services are available for use	Signature Generation	Module initialization
ECDSA SigVer (FIPS186-4) (A4751)	Curve P-256 and SHA2-256	KAT ECDSA with P-256 using SHA2-256	CAST	Module is operational and services are available for use	Signature Verification	Module initialization
HMAC-SHA-1 (A4746)	128-bit key	HMAC KAT with 128-bit key	CAST	Module is operational and services are available for use	MAC	Module initialization
HMAC-SHA-1 (A4751)	128-bit key	HMAC KAT with 128-bit key	CAST	Module is operational and services are available for use	MAC	Module initialization
HMAC-SHA-1 (A4755)	128-bit key	HMAC KAT with 128-bit key	CAST	Module is operational and services are available for use	MAC	Module initialization
RSA SigGen (FIPS186-4) (A4751)	SHA2-256	RSA KAT with PKCS#1 v1.5 with SHA-256 and 2048-bit key	CAST	Module is operational and services are available for use	Signature generation	Module initialization
RSA SigVer (FIPS186-4) (A4751)	SHA2-256	RSA KAT with PKCS#1 v1.5 with SHA-256 and 2048-bit key	CAST	Module is operational and services are available for use	Signature verification	Module initialization
HMAC-SHA2-224 (A4746)	160-bit key	HMAC KAT with 160-bit key	CAST	Module is operational and services are available for use	MAC	Module initialization
HMAC-SHA2-224 (A4751)	160-bit key	HMAC KAT with 160-bit key	CAST	Module is operational and services are available for use	MAC	Module initialization
HMAC-SHA2-224 (A4755)	160-bit key	HMAC KAT with 160-bit key	CAST	Module is operational and services are available for use	MAC	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-256 (A4746)	160-bit key	HMAC KAT with 160-bit key	CAST	Module is operational and services are available for use	MAC	Module initialization
HMAC-SHA2-256 (A4751)	160-bit key	HMAC KAT with 160-bit key	CAST	Module is operational and services are available for use	MAC	Module initialization
HMAC-SHA2-256 (A4755)	160-bit key	HMAC KAT with 160-bit key	CAST	Module is operational and services are available for use	MAC	Module initialization
HMAC-SHA2-384 (A4746)	160-bit key	HMAC KAT with 160-bit key	CAST	Module is operational and services are available for use	MAC	Module initialization
HMAC-SHA2-384 (A4751)	160-bit key	HMAC KAT with 160-bit key	CAST	Module is operational and services are available for use	MAC	Module initialization
HMAC-SHA2-384 (A4755)	160-bit key	HMAC KAT with 160-bit key	CAST	Module is operational and services are available for use	MAC	Module initialization
HMAC-SHA2-512 (A4746)	160-bit key	HMAC KAT with 160-bit key	CAST	Module is operational and services are available for use	MAC	Module initialization
HMAC-SHA2-512 (A4751)	160-bit key	HMAC KAT with 160-bit key	CAST	Module is operational and services are available for use	MAC	Module initialization
HMAC-SHA2-512 (A4755)	160-bit key	HMAC KAT with 160-bit key	CAST	Module is operational and services are available for use	MAC	Module initialization
SHA3-256 (A4747)	SHA3-256	KAT SHA3-256	CAST	Module is operational and services are available for use	Message Digest	Module initialization
SHA3-256 (A4753)	SHA3-256	KAT SHA3-256	CAST	Module is operational and services are available for use	Message Digest	Module initialization
SHA3-384 (A4747)	SHA3-384	KAT SHA3-384	CAST	Module is operational and services are available for use	Message Digest	Module initialization
SHA3-384 (A4753)	SHA3-384	KAT SHA3-384	CAST	Module is operational and services are available for use	Message Digest	Module initialization
SHA3-512 (A4747)	SHA3-512	KAT SHA3-512	CAST	Module is operational and services are available for use	Message Digest	Module initialization
SHA3-512 (A4753)	SHA3-512	KAT SHA3-512	CAST	Module is operational and	Message Digest	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				services are available for use		

Table 22: Conditional Self-Tests

If any conditional self-test fails, the module transitions to the error state.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A4746)	Message Authentication	SW/FW Integrity	On demand	Manually
HMAC-SHA2-256 (A4751)	Message Authentication	SW/FW Integrity	On demand	Manually
HMAC-SHA2-256 (A4755)	Message Authentication	SW/FW Integrity	On demand	Manually

Table 23: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
ECDSA KeyGen (FIPS186-4) (A4751)	Signature generation and Signature verification	PCT	On demand	Manually
RSA KeyGen (FIPS186-4) (A4751)	Signature generation and Signature verification	PCT	On demand	Manually
Safe Primes Key Generation (A4751)	Diffie-Hellman key generation	PCT	On demand	Manually
SHA3-224 (A4747)	KAT SHA3-224	CAST	On demand	Manually
SHA3-224 (A4753)	KAT SHA3-224	CAST	On demand	Manually
AES-CBC (A4743)	Encrypt/Decrypt KAT for CBC	CAST	On demand	Manually
AES-CBC (A4744)	Encrypt/Decrypt KAT for CBC	CAST	On demand	Manually
AES-CBC (A4745)	Encrypt/Decrypt KAT for CBC	CAST	On demand	Manually
AES-CBC (A4746)	Encrypt/Decrypt KAT for CBC	CAST	On demand	Manually
AES-CBC (A4751)	Encrypt/Decrypt KAT for CBC	CAST	On demand	Manually
AES-CBC (A4755)	Encrypt/Decrypt KAT for CBC	CAST	On demand	Manually
AES-CFB8 (A4748)	Encrypt/Decrypt KAT for CFB8	CAST	On demand	Manually
AES-CFB8 (A4749)	Encrypt/Decrypt KAT for CFB8	CAST	On demand	Manually
AES-CFB8 (A4754)	Encrypt/Decrypt KAT for CFB8	CAST	On demand	Manually
AES-GCM (A4743)	Encrypt/Decrypt KAT for GCM	CAST	On demand	Manually
AES-GCM (A4744)	Encrypt/Decrypt KAT for GCM	CAST	On demand	Manually
AES-GCM (A4745)	Encrypt/Decrypt KAT for GCM	CAST	On demand	Manually
AES-GCM (A4746)	Encrypt/Decrypt KAT for GCM	CAST	On demand	Manually
AES-GCM (A4751)	Encrypt/Decrypt KAT for GCM	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-GCM (A4755)	Encrypt/Decrypt KAT for GCM	CAST	On demand	Manually
AES-XTS Testing Revision 2.0 (A4752)	Encrypt/Decrypt KAT for XTS	CAST	On demand	Manually
KDA HKDF Sp800-56Cr1 (A4750)	KAT with HMAC-SHA2-256 for HKDF	CAST	On demand	Manually
TLS v1.2 KDF RFC7627 (A4751)	KAT with HMAC-SHA2-256 for TLS 1.2 KDF	CAST	On demand	Manually
PBKDF (A4751)	KAT with HMAC-SHA2-256 for PBKDF	CAST	On demand	Manually
Counter DRBG (A4751)	CTR_DRBG with AES without DF, without PR KAT; SP800-90Ar1 Section 11.3 Health Test	CAST	On demand	Manually
KAS-FFC-SSC Sp800-56Ar3 (A4751)	Primitive "Z" computation KAT	CAST	On demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A4751)	Primitive "Z" computation KAT	CAST	On demand	Manually
ECDSA SigGen (FIPS186-4) (A4751)	KAT ECDSA with P-256 using SHA2-256	CAST	On demand	Manually
ECDSA SigVer (FIPS186-4) (A4751)	KAT ECDSA with P-256 using SHA2-256	CAST	On demand	Manually
HMAC-SHA-1 (A4746)	HMAC KAT with 128-bit key	CAST	On demand	Manually
HMAC-SHA-1 (A4751)	HMAC KAT with 128-bit key	CAST	On demand	Manually
HMAC-SHA-1 (A4755)	HMAC KAT with 128-bit key	CAST	On demand	Manually
RSA SigGen (FIPS186-4) (A4751)	RSA KAT with PKCS#1 v1.5 with SHA-256 and 2048-bit key	CAST	On demand	Manually
RSA SigVer (FIPS186-4) (A4751)	RSA KAT with PKCS#1 v1.5 with SHA-256 and 2048-bit key	CAST	On demand	Manually
HMAC-SHA2-224 (A4746)	HMAC KAT with 160-bit key	CAST	On demand	Manually
HMAC-SHA2-224 (A4751)	HMAC KAT with 160-bit key	CAST	On demand	Manually
HMAC-SHA2-224 (A4755)	HMAC KAT with 160-bit key	CAST	On demand	Manually
HMAC-SHA2-256 (A4746)	HMAC KAT with 160-bit key	CAST	On demand	Manually
HMAC-SHA2-256 (A4751)	HMAC KAT with 160-bit key	CAST	On demand	Manually
HMAC-SHA2-256 (A4755)	HMAC KAT with 160-bit key	CAST	On demand	Manually
HMAC-SHA2-384 (A4746)	HMAC KAT with 160-bit key	CAST	On demand	Manually
HMAC-SHA2-384 (A4751)	HMAC KAT with 160-bit key	CAST	On demand	Manually
HMAC-SHA2-384 (A4755)	HMAC KAT with 160-bit key	CAST	On demand	Manually
HMAC-SHA2-512 (A4746)	HMAC KAT with 160-bit key	CAST	On demand	Manually
HMAC-SHA2-512 (A4751)	HMAC KAT with 160-bit key	CAST	On demand	Manually
HMAC-SHA2-512 (A4755)	HMAC KAT with 160-bit key	CAST	On demand	Manually
SHA3-256 (A4747)	KAT SHA3-256	CAST	On demand	Manually
SHA3-256 (A4753)	KAT SHA3-256	CAST	On demand	Manually
SHA3-384 (A4747)	KAT SHA3-384	CAST	On demand	Manually
SHA3-384 (A4753)	KAT SHA3-384	CAST	On demand	Manually
SHA3-512 (A4747)	KAT SHA3-512	CAST	On demand	Manually
SHA3-512 (A4753)	KAT SHA3-512	CAST	On demand	Manually

Table 24: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Error	The module stops functioning and ends the Application process	When the Pre-Operational Self-Test or a CAST fails When a PCT fails Crypto operations requested in the Error state	The module must be restarted and successfully 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)

Table 25: Error States

In the error state, the data output interface is inhibited, and the module accepts no more inputs or requests (as the module is no longer running). The calling application can obtain the module state by calling the gnutls_fips140_get_operation_state() API function.

10.5 Operator Initiation of Self-Tests

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

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

The module is distributed as a part of the Oracle Linux 9 (OL9) RPM in the form of gnutls-3.7.6-21.0.1.el9_2_fips, nettle-3.8-3.el9_0, gmp-6.2.0-10.el9 RPM packages that are in the “Oracle Linux 9 Security Validation (Update 3)” yum repository (ol9_u3_security_validation). Note: libhogweed is provided by nettle-3.8-3.el9_0.

The Oracle Linux 9 system FIPS validated configuration can be achieved by:

- For installation 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.
- For switching the system into the FIPS validated configuration after the installation execute the fips-mode-setup --enable command. Restart the system.

In both cases, the Crypto Officer must verify that the Oracle Linux 9 system operates in a FIPS validated configuration by executing the fips-mode-setup --check command, which shall output “FIPS mode is enabled.”

For more information on Oracle Linux 9 system FIPS validated configuration, please see Oracle Linux 9 product documentation.

After installation of the gnutls-3.7.6-21.0.1.el9_2_fips, nettle-3.8-3.el9_0, gmp-6.2.0-10.el9 RPM packages, the Crypto Officer must execute the “Show module name and version” service using the gnutls_get_library_config API. The service output must be as follows:

```
fips-module-name: Oracle Linux 9 GnuTLS Cryptographic Module

fips-module-version: 3.7.6-39e2433a29b33b55

libgnutls-soname: libgnutls.so.30

libnettle-soname: libnettle.so.8

libhogweed-soname: libhogweed.so.6
```

11.2 Administrator Guidance

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

11.3 Non-Administrator Guidance

The module does not have any guidance for non-administrator.

11.4 End of Life

As the module does not persistently store SSPs, secure sanitization of the module consists of unloading the module. This will zeroize all SSPs in volatile memory. Then, if desired, the gnutls-3.7.6-21.0.1.el9_2_fips, nettle-3.8-3.el9_0, gmp-6.2.0-10.el9 RPM packages can be uninstalled from the Oracle Linux 9 system.

12 Mitigation of Other Attacks

The module does not offer mitigation of other attacks.

Appendix A. TLS Cipher Suites

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

Cipher Suite	ID	Reference
TLS_DH_RSA_WITH_AES_128_CBC_SHA	{ 0x00, 0x31 }	RFC3268
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	{ 0x00, 0x33 }	RFC3268
TLS_DH_RSA_WITH_AES_256_CBC_SHA	{ 0x00, 0x37 }	RFC3268
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	{ 0x00, 0x39 }	RFC3268
TLS_DH_RSA_WITH_AES_128_CBC_SHA256	{ 0x00,0x3F }	RFC5246
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	{ 0x00,0x67 }	RFC5246
TLS_DH_RSA_WITH_AES_256_CBC_SHA256	{ 0x00,0x69 }	RFC5246
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	{ 0x00,0x6B }	RFC5246
TLS_PSK_WITH_AES_128_CBC_SHA	{ 0x00, 0x8C }	RFC4279
TLS_PSK_WITH_AES_256_CBC_SHA	{ 0x00, 0x8D }	RFC4279
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	{ 0x00, 0x9E }	RFC5288
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	{ 0x00, 0x9F }	RFC5288
TLS_DH_RSA_WITH_AES_128_GCM_SHA256	{ 0x00, 0xA0 }	RFC5288
TLS_DH_RSA_WITH_AES_256_GCM_SHA384	{ 0x00, 0xA1 }	RFC5288
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x04 }	RFC4492
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x05 }	RFC4492
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x09 }	RFC4492
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x0A }	RFC4492
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x0E }	RFC4492
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x0F }	RFC4492
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x13 }	RFC4492
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x14 }	RFC4492
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x23 }	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x24 }	RFC5289
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x25 }	RFC5289
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x26 }	RFC5289
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x27 }	RFC5289
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x28 }	RFC5289
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x29 }	RFC5289
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x2A }	RFC5289

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

Appendix B. Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
API	Application Programming Interface
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
GMAC	Galois Counter Mode Message Authentication Code
HKDF	HMAC-based Key Derivation Function
HMAC	Keyed-Hash Message Authentication Code
KAT	Known Answer Test
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
PAA	Processor Algorithm Acceleration
PBKDF2	Password-based Key Derivation Function v2
PKCS	Public-Key Cryptography Standards
RSA	Rivest, Shamir, Adleman
SHA	Secure Hash Algorithm
SSC	Shared Secret Computation
SSP	Sensitive Security Parameter
TOEPP	Tested Operational Environment's Physical Perimeter
XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

Appendix C. References

FIPS 140-3	FIPS PUB 140-3 - Security Requirements For Cryptographic Modules March 2019 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf
FIPS 140-3 IG	Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements
FIPS 180-4	Secure Hash Standard (SHS) March 2012 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS 186-4	Digital Signature Standard (DSS) July 2013 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
FIPS 186-5	Digital Signature Standard (DSS) February 2023 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf
FIPS 197	Advanced Encryption Standard November 2001 https://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS 198-1	The Keyed Hash Message Authentication Code (HMAC) July 2008 https://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
FIPS 202	SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions August 2015 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
PKCS#1	Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1 February 2003 https://www.ietf.org/rfc/rfc3447.txt
RFC 3526	More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE) May 2003 https://www.ietf.org/rfc/rfc3526.txt
RFC 5288	AES Galois Counter Mode (GCM) Cipher Suites for TLS August 2008 https://www.ietf.org/rfc/rfc5288.txt
RFC 7919	Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS) August 2016 https://www.ietf.org/rfc/rfc7919.txt
RFC 8446	The Transport Layer Security (TLS) Protocol Version 1.3 August 2018 https://www.ietf.org/rfc/rfc8446.txt
SP 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

SP 800-38B	Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication May 2005 https://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf
SP 800-38C	Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality May 2004 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf
SP 800-38D	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC November 2007 https://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf
SP 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
SP 800-52r2	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
SP 800-56Ar3	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography April 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf
SP 800-56Cr1	Recommendation for Key-Derivation Methods in Key-Establishment Schemes August 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr1.pdf
SP 800-56Cr2	Recommendation for Key-Derivation Methods in Key-Establishment Schemes August 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf
SP 800-90Ar1	Recommendation for Random Number Generation Using Deterministic Random Bit Generators June 2015 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP 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
SP 800-108r1	NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions August 2022 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-108r1.pdf
SP 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
SP 800-133r2	Recommendation for Cryptographic Key Generation June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf
SP 800-135r1	Recommendation for Existing Application-Specific Key Derivation Functions December 2011 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf

SP 800-140B	CMVP Security Policy Requirements March 2020 **https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140B.pdf**