SUSE, LLC

SUSE Linux Enterprise GnuTLS Cryptographic Module

FIPS 140-3 Non-Proprietary Security Policy

Version 1.3

Last Update: 2026-07-09

Prepared by:

atsec information security corporation

4516 Seton Center Parkway, Suite 250

Austin, TX 78759

www.atsec.com

1 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 3.1 of the SUSE Linux Enterprise NSS 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. It has a one-to-one mapping to SP 800-140B starting with section B.2.1 named "General" which maps to Section 1 in this document and ending with section B.2.12 named "Mitigation of other attacks" which maps to Section 12 in this document.This Non-Proprietary Security Policy may be reproduced and distributed, but only whole and intact and including this notice. Other documentation is proprietary to their authors.

1.2 Security Levels

Table 1 describes the individual security areas of FIPS 140-3, as well as the security levels of those individual areas.

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

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The SUSE Linux Enterprise GnuTLS Cryptographic Module (hereafter referred to as "the module") is a software library that provides a C language application program interface (API) for use by other applications that require cryptographic functionality. The module operates on a general-purpose computer.

Module Type: Software

Module Embodiment: Multi-Chip Standalone

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

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 shows a block diagram that represents the design of the module when the module is operational and providing services to other user space applications. In this diagram, the physical perimeter of the operational environment is the general-purpose computer on which the module is installed. The cryptographic boundary is represented by the libgnutls, libnettle, libhogweed, and libgmp shared libraries, as well as their respective integrity check files.

The Data/Control Input and Data/Status Output arrows indicate the flow of data between the cryptographic module and its operator application through the logical interfaces defined in Section 3.

Other components are only included in the diagram for informational purposes. They are not included in the cryptographic boundary and therefore are not part of the module's validation.

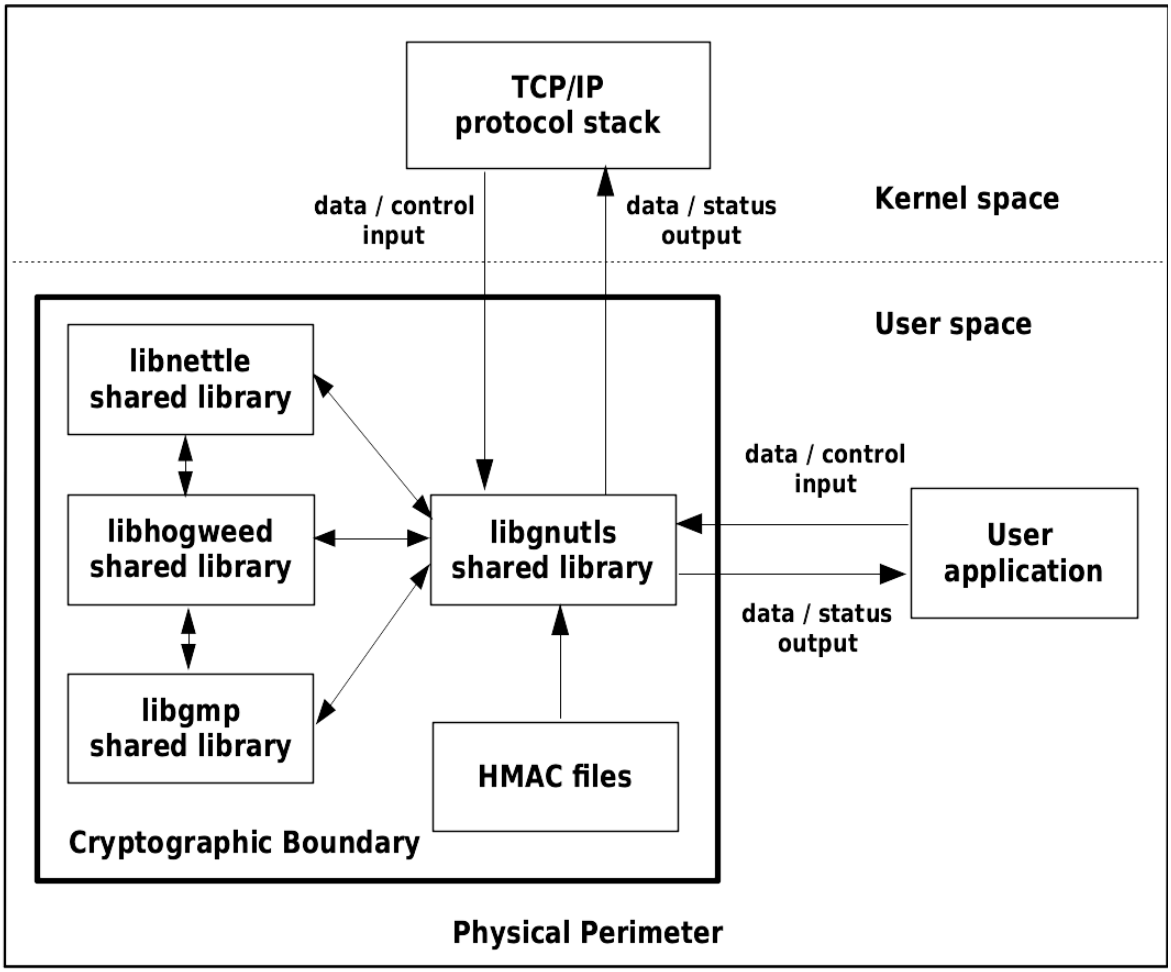

Figure 1 - Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

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

The following table lists the software components of the cryptographic module, which defines its cryptographic boundary.

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libgnutls.so.30, libnettle.so.8, libhogweed.so.6, libgmp.so.10 on SUSE Linux Enterprise Server 15 SP4 and Intel®	1.1	N/A	HMAC-SHA2-256

Package or File Name	Software/ Firmware Version	Features	Integrity Test
Xeon® Silver 4215R or AMD EPYCTM 7371			
libgnutls.so.30, libnettle.so.8, libhogweed.so.6, libgmp.so.10 on SUSE Linux Enterprise Server 15 SP4 and ARM Ampere® Altra® Q80-30	1.1	N/A	HMAC-SHA2-256
libgnutls.so.30, libnettle.so.8, libhogweed.so.6, libgmp.so.10 on SUSE Linux Enterprise Server 15 SP4 and IBM z15	1.1	N/A	HMAC-SHA2-256
libgnutls.so.30, libnettle.so.8, libhogweed.so.6, libgmp.so.10 on SUSE Linux Enterprise Server 15 SP4 on PowerVM (VIOS 3.1.4.00) and IBM Power10	1.1	N/A	HMAC-SHA2-256

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

Tested Operational Environments - Software, Firmware, Hybrid:

The module has been tested on the following platforms with the corresponding module variants and configuration options:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
SUSE Linux Enterprise Server 15 SP4	Supermicro Super Server SYS-6019P-WTR	Intel® Xeon® Silver 4215R	Yes	N/A	1.1

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
SUSE Linux Enterprise Server 15 SP4	Supermicro Super Server SYS-6019P-WTR	Intel® Xeon® Silver 4215R	No	N/A	1.1
SUSE Linux Enterprise Server 15 SP4	GIGABYTE R181-Z90-00	AMD EPYCTM 7371	Yes	N/A	1.1
SUSE Linux Enterprise Server 15 SP4	GIGABYTE R181-Z90-00	AMD EPYCTM 7371	No	N/A	1.1
SUSE Linux Enterprise Server 15 SP4	GIGABYTE G242-P32-QZ	ARM Ampere® Altra® Q80-30	Yes	N/A	1.1
SUSE Linux Enterprise Server 15 SP4	GIGABYTE G242-P32-QZ	ARM Ampere® Altra® Q80-30	No	N/A	1.1
SUSE Linux Enterprise Server 15 SP4	IBM z/15	z15	Yes	N/A	1.1
SUSE Linux Enterprise Server 15 SP4	IBM z/15	z15	No	N/A	1.1
SUSE Linux Enterprise Server 15 SP4	IBM Power E1080 (9080-HEX)	Power10	Yes	PowerVM (VIOS 3.1.4.00)	1.1
SUSE Linux Enterprise Server 15 SP4	IBM Power E1080 (9080-HEX)	Power10	No	PowerVM (VIOS 3.1.4.00)	1.1

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

The module implements Processor Algorithm Implementation (PAI) for the IBM z/15 platform and Processor Algorithm Acceleration (PAA) for all other tested platforms listed above.

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

Operating System	Hardware Platform
SUSE Linux Enterprise Server 15SP4	IBM LinuxONE III LT1 [z15]
SUSE Linux Enterprise Micro 5.3	Supermicro Super Server SYS-6019P-WTR [Intel® Xeon® Silver 4215R]
SUSE Linux Enterprise Micro 5.3	GIGABYTE R181-Z90-00 [AMD EPYCTM 7371]
SUSE Linux Enterprise Micro 5.3	GIGABYTE G242-P32-QZ [ARM Ampere® Altra® Q80-30]
SUSE Linux Enterprise Micro 5.3	IBM z/15 [z15]
SUSE Linux Enterprise Micro 5.3	IBM LinuxONE III LT1 [z15]
SUSE Linux Enterprise Server for SAP 15SP4	Supermicro Super Server SYS-6019P-WTR [Intel® Xeon® Silver 4215R]
SUSE Linux Enterprise Server for SAP 15SP4	GIGABYTE R181-Z90-00 [AMD EPYCTM 7371]
SUSE Linux Enterprise Server for SAP 15SP4	IBM Power E1080 (9080-HEX) [Power10]
SUSE Linux Enterprise Base Container Image 15SP4	Supermicro Super Server SYS-6019P-WTR [Intel® Xeon® Silver 4215R]
SUSE Linux Enterprise Base Container Image 15SP4	GIGABYTE R181-Z90-00 [AMD EPYCTM 7371]
SUSE Linux Enterprise Base Container Image 15SP4	GIGABYTE G242-P32-QZ [ARM Ampere® Altra® Q80-30]
SUSE Linux Enterprise Base Container Image 15SP4	IBM z/15 [z15]
SUSE Linux Enterprise Base Container Image 15SP4	IBM LinuxONE III LT1 [z15]
SUSE Linux Enterprise Base Container Image 15SP4	IBM Power E1080 (9080-HEX) [Power10]
SUSE Linux Enterprise Desktop 15SP4	Supermicro Super Server SYS-6019P-WTR [Intel® Xeon® Silver 4215R]
SUSE Linux Enterprise Desktop 15SP4	GIGABYTE R181-Z90-00 [AMD EPYCTM 7371]

Operating System	Hardware Platform
SUSE Linux Enterprise Real Time 15SP4	Supermicro Super Server SYS-6019P-WTR [Intel® Xeon® Silver 4215R]
SUSE Linux Enterprise Real Time 15SP4	GIGABYTE R181-Z90-00 [AMD EPYCTM 7371]

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.

The SUSE Linux Enterprise Server operating system is used as the basis of other products. Compliance is maintained for SUSE products whenever the binary is found unchanged per the vendor affirmation from SUSE based on the allowance provided by FIPS 140-3 Management Manual section 7.9.1 bullet 1 a i).

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
Non-approved mode	Automatically entered whenever a non-approved service is requested	Non-Approved	Equivalent to the indicator of the requested service

Table 5: Modes List and Description

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

Mode change instructions and Status:

If the module is in the approved mode, it can only be transitioned into the non-Approved mode by calling one of the non-Approved services listed in the Non-Approved Services table in Section 4.4. The module switches between approved and non-approved mode based on the service requested. Please see Section 4.3 for the details on the service indicator provided by the module that identifies when an approved service is called.

2.5 Algorithms

Approved Algorithms:

The following table lists all approved algorithms and their corresponding CAVP certificates.

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A2984	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A2985	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A2986	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A2987	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A2992	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A2996	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A2997	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A3004	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A3007	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A2984	Key Length - 128, 256 Tag Length - 112, 128, 32, 48, 64, 80, 96 IV Length - IV Length: 56, 64, 72, 80, 88, 96, 104 Payload Length - Payload Length: 0-256 Increment 8 AAD Length - AAD Length: 0, 256, 65536	SP 800-38C
AES-CCM	A2996	Key Length - 128, 256 Tag Length - 112, 128, 32, 48, 64, 80, 96 IV Length - IV Length: 56, 64, 72, 80, 88, 96, 104 Payload Length - Payload Length: 0-256 Increment 8 AAD Length - AAD Length: 0, 256, 65536	SP 800-38C
AES-CCM	A3004	Key Length - 128, 256 Tag Length - 112, 128, 32, 48, 64, 80, 96 IV Length - IV Length: 56, 64, 72, 80, 88, 96, 104	SP 800-38C

Algorithm	CAVP Cert	Properties	Reference
		Payload Length - Payload Length: 0-256 Increment 8 AAD Length - AAD Length: 0, 256, 65536	
AES-CCM	A3007	Key Length - 128, 256 Tag Length - 112, 128, 32, 48, 64, 80, 96 IV Length - IV Length: 56, 64, 72, 80, 88, 96, 104 Payload Length - Payload Length: 0-256 Increment 8 AAD Length - AAD Length: 0, 256, 65536	SP 800-38C
AES-CFB8	A2989	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A2990	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A2995	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A2984	Direction - Generation, Verification Key Length - 128, 256 MAC Length - MAC Length: 128 Message Length - Message Length: 8-524288 Increment 8	SP 800-38B
AES-CMAC	A2987	Direction - Generation, Verification Key Length - 128, 256 MAC Length - MAC Length: 128 Message Length - Message Length: 8-524288 Increment 8	SP 800-38B
AES-CMAC	A2992	Direction - Generation, Verification Key Length - 128, 256 MAC Length - MAC Length: 128 Message Length - Message Length: 8-524288 Increment 8	SP 800-38B
AES-CMAC	A2996	Direction - Generation, Verification Key Length - 128, 256 MAC Length - MAC Length: 128 Message Length - Message Length: 8-524288 Increment 8	SP 800-38B
AES-CMAC	A3004	Direction - Generation, Verification Key Length - 128, 256 MAC Length - MAC Length: 128 Message Length - Message Length: 8-524288 Increment 8	SP 800-38B
AES-GCM	A2984	Direction - Decrypt, Encrypt IV Generation - External	SP 800-38D

Algorithm	CAVP Cert	Properties	Reference
		IV Generation Mode - 8.2.1 Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 Payload Length - Payload Length: 128, 256, 120, 248 AAD Length - AAD Length: 128, 256, 120, 0	
AES-GCM	A2985	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 Payload Length - Payload Length: 128, 256, 120, 248 AAD Length - AAD Length: 128, 256, 120, 0	SP 800-38D
AES-GCM	A2986	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 Payload Length - Payload Length: 128, 256, 120, 248 AAD Length - AAD Length: 128, 256, 120, 0	SP 800-38D
AES-GCM	A2987	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 Payload Length - Payload Length: 128, 256, 120, 248 AAD Length - AAD Length: 128, 256, 120, 0	SP 800-38D
AES-GCM	A2992	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 Payload Length - Payload Length: 128, 256, 120, 248 AAD Length - AAD Length: 128, 256, 120, 0	SP 800-38D
AES-GCM	A2996	Direction - Decrypt, Encrypt IV Generation - External	SP 800-38D

Algorithm	CAVP Cert	Properties	Reference
		IV Generation Mode - 8.2.1 Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 Payload Length - Payload Length: 128, 256, 120, 248 AAD Length - AAD Length: 128, 256, 120, 0	
AES-GCM	A2997	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 Payload Length - Payload Length: 128, 256, 120, 248 AAD Length - AAD Length: 128, 256, 120, 0	SP 800-38D
AES-GCM	A3004	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 Payload Length - Payload Length: 128, 256, 120, 248 AAD Length - AAD Length: 128, 256, 120, 0	SP 800-38D
AES-GCM	A3007	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 Payload Length - Payload Length: 128, 256, 120, 248 AAD Length - AAD Length: 128, 256, 120, 0	SP 800-38D
AES-GMAC	A2992	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 AAD Length - AAD Length: 128, 256, 120, 0	SP 800-38D

Algorithm	CAVP Cert	Properties	Reference
AES-XTS Testing Revision 2.0	A2993	Direction - Decrypt, Encrypt Key Length - 128, 256 Payload Length - Payload Length: 128-65536 Increment 128 Tweak Mode - Hex Data Unit Length Matches Payload Length - Yes	SP 800-38E
Counter DRBG	A2992	Prediction Resistance - No Supports Reseed - Yes Mode - AES-256 Derivation Function Enabled - No Additional Input - Additional Input: 0, 256 Entropy Input - Entropy Input: 384 Nonce - Nonce: 128 Personalization String Length - Personalization String Length: 0, 256 Returned Bits - 512	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-4)	A2992	Curve - P-256, P-384, P-521 Secret Generation Mode - Testing Candidates	FIPS 186-4
ECDSA KeyVer (FIPS186-4)	A2992	Curve - P-256, P-384, P-521	FIPS 186-4
ECDSA SigGen (FIPS186-4)	A2992	Component - No Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A2992	Component - No Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
HMAC-SHA-1	A2987	MAC - MAC: 160 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA-1	A2992	MAC - MAC: 160 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA-1	A2998	MAC - MAC: 160 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA-1	A3007	MAC - MAC: 160 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1

Algorithm	CAVP Cert	Properties	Reference
HMAC-SHA2-224	A2987	MAC - MAC: 224 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A2992	MAC - MAC: 224 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A2998	MAC - MAC: 224 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A3007	MAC - MAC: 224 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A2987	MAC - MAC: 256 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A2992	MAC - MAC: 256 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A2998	MAC - MAC: 256 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A3007	MAC - MAC: 256 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A2987	MAC - MAC: 384 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A2992	MAC - MAC: 384 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A2998	MAC - MAC: 384 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A3007	MAC - MAC: 384 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A2987	MAC - MAC: 512 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A2992	MAC - MAC: 512 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A2998	MAC - MAC: 512 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1

Algorithm	CAVP Cert	Properties	Reference
HMAC-SHA2-512	A3007	MAC - MAC: 512 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A2992	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	A2992	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	A2991	Fixed Info Pattern - uPartyInfo||vPartyInfo Fixed Info Encoding - concatenation Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-65336 Increment 8 HMAC Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	SP 800-56C Rev. 2
KDF TLS (CVL)	A2992	TLS Version - v1.0/1.1	SP 800-135 Rev. 1
PBKDF	A2992	Iteration Count - Iteration Count: 10-1000 Increment 1 HMAC Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512 Password Length - Password Length: 8-128 Increment 1 Salt Length - Salt Length: 128-4096 Increment 8 Key Data Length - Key Data Length: 128-4096 Increment 8	SP 800-132
RSA KeyGen (FIPS186-4)	A2992	Key Generation Mode - B.3.2 Modulo - 2048, 3072, 4096 Hash Algorithm - SHA2-384 Primality Tests - Table C.2 Info Generated By Server - No Public Exponent Mode - Random Private Key Format - Standard	FIPS 186-4
RSA SigGen (FIPS186-4)	A2992	Signature Type - PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096	FIPS 186-4

Algorithm	CAVP Cert	Properties	Reference
		Hash Pair - Hash Algorithm - SHA2-224	
RSA SigVer (FIPS186-4)	A2992	Signature Type - PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA2-224 Public Exponent Mode - Random	FIPS 186-4
Safe Primes Key Generation	A2992	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	A2987	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA-1	A2992	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA-1	A2998	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA-1	A3007	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-224	A2987	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-224	A2992	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-224	A2998	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-224	A3007	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A2987	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A2992	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A2998	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A3007	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A2987	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A2992	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A2998	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A3007	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4

Algorithm	CAVP Cert	Properties	Reference
SHA2-512	A2987	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A2992	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A2998	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A3007	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA3-224	A2988	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-224	A2994	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-256	A2988	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-256	A2994	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-384	A2988	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-384	A2994	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-512	A2988	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-512	A2994	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
TLS v1.2 KDF RFC7627 (CVL)	A2992	Hash Algorithm - SHA2-256, SHA2-384	SP 800-135 Rev. 1

Table 6: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG		N/A	SP 800-133 Rev. 2 Section 4 Example 1 with V=0

Table 7: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

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

Non-Approved, Allowed Algorithms with No Security Claimed:

The following table lists the non-approved algorithms that are allowed in the approved mode of operation with no security claimed. These algorithms are used by the approved services listed in Section 4.3.

Name	Caveat	Use and Function
MD5	Only allowed as part of the PRF in TLSv1.0 and v1.1 per IG 2.4.A	Message Digest used in TLS v1.0/1.1 KDF only

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

Non-Approved, Not Allowed Algorithms:

The following table lists non-approved algorithms that are not allowed in the approved mode of operation. These algorithms are used by the non-approved services listed in Section 4.4.

Name	Use and Function
AES GCM when not used in the context of the TLS protocol.	Authenticated symmetric encryption; Authenticated symmetric decryption
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 with keys generated with domain parameters other than safe primes	Key agreement; Diffie-Hellman shared secret computation
DSA	Key pair generation; Domain parameter generation; Digital signature generation; Digital signature verification
ECDSA with curves not listed in CAVP certificates found in Table 6.	Key pair generation; Public key verification; Digital signature generation; Digital signature verification
EC Diffie-Hellman with curves not listed in CAVP certificates found in Table 6	Key agreement; EC Diffie-Hellman shared secret computation
GMAC	Message authentication code (MAC)
GOST	Symmetric encryption; Symmetric decryption; Message digest

Name	Use and Function
HMAC with keys smaller than 112-bits	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	Password-based 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 pair 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 un-encapsulation
Salsa20	Symmetric encryption; Symmetric decryption
SEED	Symmetric encryption; Symmetric decryption
Serpent	Symmetric encryption; Symmetric decryption
SHA-1	Digital signature generation
SRP	Key agreement
STREEBOG	Message digest; Message authentication code (MAC)
Non-supported cipher suites (not listed in Appendix A)	Transport Layer Security (TLS) Network Protocol
Triple-DES	Symmetric encryption; Symmetric decryption
Twofish	Symmetric encryption; Symmetric decryption
UMAC	Message authentication code (MAC)
Yarrow	Random number generation

Table 9: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

The following table lists all security functions of the module, including specific key strengths employed for approved services, and implemented modes of operation.

Name	Type	Description	Properties	Algorithms
Symmetric encryption	BC-UnAuth	Symmetric encryption		AES-CBC: (A3007, A2992, A2985, A2987, A2996, A2997, A3004, A2984, A2986) AES-CFB8: (A2995, A2989, A2990) AES-XTS Testing Revision 2.0: (A2993)
Symmetric decryption	BC-UnAuth	Symmetric decryption		AES-CBC: (A3007, A2992, A2985, A2987, A2996, A2997, A3004, A2984, A2986) AES-CFB8: (A2995, A2989, A2990) AES-XTS Testing Revision 2.0: (A2993)
Authenticated symmetric encryption	BC-Auth	Authenticated symmetric encryption		AES-CCM: (A3007, A2996, A3004, A2984)
Authenticated symmetric decryption	BC-Auth	Authenticated symmetric decryption		AES-CCM: (A3007, A2996, A3004, A2984)
Message authentication code (MAC)	MAC	Message authentication code (MAC)		AES-CMAC: (A2992, A2987, A2996, A3004, A2984) HMAC-SHA-1: (A3007, A2992, A2998, A2987) HMAC-SHA2-224: (A3007, A2992, A2998, A2987)

Name	Type	Description	Properties	Algorithms
				HMAC-SHA2-256: (A3007, A2992, A2998, A2987) HMAC-SHA2-384: (A3007, A2992, A2998, A2987) HMAC-SHA2-512: (A3007, A2992, A2998, A2987) AES-GMAC: (A2992)
Authenticated symmetric encryption for TLS	BC-Auth	Symmetric encryption in the context of the Transport Layer Security (TLS) network protocol		AES-GCM: (A3007, A2992, A2985, A2987, A2996, A2997, A3004, A2984, A2986) AES-CCM: (A2984, A2996, A3004, A3007) AES-CBC: (A2984, A2985, A2986, A2987, A2992, A2996, A2997, A3004, A3007) HMAC-SHA2-256: (A2987, A2992, A2998, A3007) HMAC-SHA2-512: (A2987, A2992, A2998, A3007)
Authenticated symmetric decryption for TLS	BC-Auth	Symmetric decryption in the context of the Transport Layer Security (TLS) network protocol		AES-GCM: (A3007, A2992, A2985, A2987, A2996, A2997, A3004, A2984, A2986) AES-CCM: (A2984, A2996, A3004, A3007) AES-CBC: (A2984, A2985, A2986, A2987, A2992, A2996, A2997, A3004, A3007) HMAC-SHA2-256:

Name	Type	Description	Properties	Algorithms
				(A2987, A2992, A2998, A3007) HMAC-SHA2-512: (A2987, A2992, A2998, A3007)
Random number generation	DRBG	Random number generation		Counter DRBG: (A2992)
Key pair generation	AsymKeyPair-KeyGen CKG	Key pair generation	RSA:Only key lengths between 2048 and 4096 bits are approved.	ECDSA KeyGen (FIPS186-4): (A2992) RSA KeyGen (FIPS186-4): (A2992) Safe Primes Key Generation: (A2992) CKG: ()
Public key verification	AsymKeyPair-KeyVer	Public key verification		ECDSA KeyVer (FIPS186-4): (A2992)
Digital signature generation	DigSig-SigGen	Digital signature generation	RSA:Only key lengths between 2048 and 4096 bits are approved.	ECDSA SigGen (FIPS186-4): (A2992) RSA SigGen (FIPS186-4): (A2992)
Digital signature verification	DigSig-SigVer	Digital signature verification	RSA:Only key lengths between 1024 and 4096 bits are approved.	ECDSA SigVer (FIPS186-4): (A2992) RSA SigVer (FIPS186-4): (A2992)
ECDH SSC for TLS	KAS-SSC	EC Diffie-Hellman shared secret computation; Transport Layer Security (TLS) network protocol		KAS-ECC-SSC Sp800-56Ar3: (A2992)

Name	Type	Description	Properties	Algorithms
DH SSC for TLS	KAS-SSC	Diffie-Hellman shared secret computation; Transport Layer Security (TLS) network protocol		KAS-FFC-SSC Sp800-56Ar3: (A2992)
HKDF key derivation for TLS	KAS-56CKDF	HKDF key derivation; Transport Layer Security (TLS) network protocol		KDA HKDF Sp800-56Cr1: (A2991)
TLS key derivation	KAS-135KDF	TLS key derivation		KDF TLS: (A2992) TLS v1.2 KDF RFC7627: (A2992)
Password-based key derivation	PBKDF	Password-based key derivation		PBKDF: (A2992)
Message digest	SHA	Message digest		SHA3-224: (A2988, A2994) SHA3-256: (A2988, A2994) SHA3-384: (A2988, A2994) SHA3-512: (A2988, A2994) SHA-1: (A3007, A2992, A2998, A2987) SHA2-224: (A3007, A2992, A2998, A2987) SHA2-256: (A3007, A2992, A2998, A2987) SHA2-384: (A3007, A2992, A2998, A2987) SHA2-512: (A3007, A2992, A2998, A2987)

Name	Type	Description	Properties	Algorithms
DH KAS	KAS-Full	Diffie-Hellman KAS in the context of TLS	IG:IG D.F Scenario 2, path (2), split Key Confirmation:no Key Derivation:IG 2.4.B SP 800-135rev1 CVL and KDA (separately tested) Caveat:Key establishment methodology provides between 112 and 200 bits of security strength	Safe Primes Key Generation: (A2992) KAS-FFC-SSC Sp800-56Ar3: (A2992) KDA HKDF Sp800-56Cr1: (A2991) KDF TLS: (A2992) TLS v1.2 KDF RFC7627: (A2992)
ECDH KAS	KAS-Full	Elliptic Curve Diffie-Hellman KAS in the context of TLS	IG:IG D.F Scenario 2, path (2), split Key Confirmation:no Key Derivation:IG 2.4.B SP 800-135r1 CVL and KDA (separately tested) Caveat:Key establishment methodology provides between 112 and 200 bits of security strength	ECDSA KeyGen (FIPS186-4): (A2992) KAS-ECC-SSC Sp800-56Ar3: (A2992) KDA HKDF Sp800-56Cr1: (A2991) KDF TLS: (A2992) TLS v1.2 KDF RFC7627: (A2992)

Table 10: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 AES XTS

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

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 (in compliance with IG C.I).

AES-XTS keys (i.e., Key_1 and Key_2) entered into the module shall be generated and/or established independently according to NIST SP 800-133rev2, Section 6.3. for an approved use of AES-XTS.

2.7.2 AES GCM IV

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

For TLS v1.2, IV generation is in compliance with scenario 1.a of IG C.H and [RFC 5288]. The module supports acceptable AES-GCM cipher suites from section 3.3.1 of [SP 800-52 Rev. 2].

For TLS v1.3, IV generation is in compliance with scenario 5 of IG C.H and [RFC 8446]. The module supports acceptable AES-GCM cipher suites from section 3.3.1 of [SP 800-52 Rev. 2].

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

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

2.7.3 Key derivation using SP 800-132 PBKDF

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

In accordance with SP 800-132, the following requirements shall be met.

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

2.7.4 SP 800-56A Rev. 3 Assurances

To comply with the assurances found in Section 5.6.2 of [SP 800-56A Rev. 3], the operator must use the module together with an application that implements the TLS protocol. Additionally, the module's approved "Key pair generation" service must be used to generate ephemeral Diffie- Hellman or EC Diffie-Hellman key pairs, or the key pairs must be obtained from another FIPS-validated module.

As part of this service, the module will internally perform the full public key validation of the generated public key. The module's shared secret computation service will internally perform the full public key validation of the peer public key, complying with Sections 5.6.2.2.1 and 5.6.2.2.2 of SP 800-56A Rev. 3.

2.7.5 RSA Moduli Sizes

The module supports RSA key generation, signature generation, and signature verification with any even moduli size between 2048 and 15360 bits. Only moduli lengths between 2048 and 4096 bits may be used in the approved mode of operation for RSA key generation and signature generation, while only moduli lengths between 1024 and 4096 bits may be used in the approved mode of operation for RSA signature verification.

Moduli lengths other than 2048, 3072, and 4096 bits cannot be tested by CAVP but are approved for usage in RSA key generation, signature generation, and signature verification as per IG C.F.

2.7.6 Key Transport

The module does not establish SSPs using an approved key transport scheme (KTS). However, it does offer approved authentication algorithms that can be used by an external operator/application as part of an approved KTS.

2.8 RBG and Entropy

Cert Number	Vendor Name
E29	SUSE, LLC

Table 11: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Userspace Standalone CPU Time Jitter RNG version 3.4.0	Non-Physical	See Operating Environment Table	256 bits	256 bits	SHA3-256 (A3034)

Table 12: Entropy Sources

The module employs a Deterministic Random Bit Generator (DRBG) based on SP 800-90A Rev. 1 for the generation of random values used in asymmetric keys, and for providing an RNG service to calling applications. The approved DRBG provided by the module is CTR_DRBG with AES-256. The DRBG does not employ

prediction resistance or a derivation function. The module uses an SP 800-90B-compliant entropy source specified in the table above to seed the DRBG.

The DRBG is instantiated with a 384-bit entropy input (corresponding to 384 bits of entropy). Additionally, the DRBG is reseeded with a 256-bit 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 sections 5.1 and 5.2 of SP 800-133 Rev. 2.

- For generating RSA and ECDSA keys, the module implements asymmetric cryptographic key generation (CKG) services compliant with FIPS 186-4.
- 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 FIPS 186-4 and SP 800-56A Rev. 3.
- The public and private keys used in the Diffie-Hellman key agreement scheme are also compliant with SP 800-56A Rev. 3. The module generates keys using safe primes defined in RFC 7919 and RFC 3526, as described in the next section.

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

Key Derivation

Additionally, the module supports the following key derivation methods:

- KDF TLS (CVL), compliant with SP 800-135 Rev. 1: derivation of secret keys in the context of TLS 1.0/1.1,
- TLS v1.2 KDF RFC7627 (CVL), compliant with SP 800-135 Rev. 1: derivation of secret keys in the context of TLS 1.2,
- KDA HKDF SP 800-56C Rev. 1, compliant with SP 800-56C Rev 1: derivation of secret keys in the context of SP 800-56A Rev. 3 key agreement schemes,
- Password-based key derivation (PBKDF) compliant with option 1a of SP 800-132. Keys derived from passwords or passphrases using this method can only be used in storage applications.

2.10 Key Establishment

The module provides Diffie-Hellman and EC Diffie-Hellman shared secret computation compliant with SP 800-56A Rev. 3 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 the derivation of the keying material using SP 800-135 Rev. 1 KDF.

For Diffie-Hellman, the module supports the use of safe primes from RFC 7919 for domain parameters and key generation, which are used in the TLS key agreement implemented by the module.

- TLS (RFC 7919)
- 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 RFC 3526, 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 (RFC 3526)
- MODP-2048 (ID=14)
- MODP-3072 (ID=15)
- MODP-4096 (ID=16)
- MODP-6144 (ID=17)
- MODP-8192 (ID=18)

According to Table 2: Comparable strengths in SP 800-57 Rev. 5, the key sizes of AES, Diffie-Hellman and EC Diffie-Hellman provides the following security strength in approved mode of operation:

- Diffie-Hellman key agreement provides between 112 and 200 bits of encryption strength.
- EC Diffie-Hellman key agreement provides between 128 and 256 bits of encryption strength.

2.11 Industry Protocols

The TLS protocol implementation provides both server and client sides. In order to operate in the approved mode, digital certificates used for server and client authentication shall comply with the restrictions of key size and message digest algorithms imposed by [SP 800-131A Rev. 2]. In addition, as also required by [SP 800-131A Rev. 2], Diffie-Hellman with keys smaller than 2048 bits must not be used.

The TLS protocol lacks the support to negotiate the used Diffie-Hellman key sizes. To ensure full support for all TLS protocol versions, the TLS client implementation of the module accepts Diffie-Hellman key sizes smaller than 2048 bits offered by the TLS server.

For complying with the requirement to not allow Diffie-Hellman key sizes smaller than 2048 bits, the Crypto Officer must ensure that:

- in case the module is used as a TLS server, the Diffie-Hellman parameters must be 2048 bits or larger;
- in case the module is used as a TLS client, the TLS server must be configured to only offer Diffie-Hellman keys of 2048 bits or larger.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

As a software-only module, the module does not have physical ports. The operator can only interact with the module through the API provided by the module. Thus, the physical ports are interpreted to be the physical ports of the hardware platform on which the module runs. The following table shows the logical interfaces implemented in the module. Note that the control output interface is omitted as the module does not implement it.

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 the error state.

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 13: Ports and Interfaces

4 Roles, Services, and Authentication

4.1 Authentication Methods

The module does not support authentication.

4.2 Roles

The Crypto Officer role is implicitly and always assumed by the operator of the module.

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	Crypto Officer	None

Table 14: Roles

4.3 Approved Services

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

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

The details of the approved cryptographic algorithms including the CAVP certificate numbers can be found in Section 2.5.

The "Indicator" column shows the service indicator API function that must be used to verify the service indicator after executing a service. The gnutls_fips140_get_operation_state() function indicates GNUTLS_FIPS140_OP_APPROVED whether the API invoked corresponds to an approved algorithm.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Symmetric encryption	Perform AES encryption	GNUTLS_FIPS140_OP_APPROVED	Key, Plaintext	Ciphertext	Symmetric encryption	Crypto Officer - AES key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Symmetric decryption	Perform AES decryption	GNUTLS_FIPS140_OP_APPROVED	Key, Ciphertext	Plaintext	Symmetric decryption	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	Crypto Officer - AES key: W,E
Authenticated symmetric decryption	Decrypt a ciphertext	GNUTLS_FIPS140_OP_APPROVED	Key, Ciphertext, MAC tag, IV	Plaintext or Fail	Authenticated symmetric decryption	Crypto Officer - AES key: W,E
Key pair generation	Generate RSA, DH, ECDH and ECDSA key pairs	GNUTLS_FIPS140_OP_APPROVED	RSA key size, Diffie-Hellman safe prime group or Elliptic curve	RSA keypair, Diffie-Hellman keypair or Elliptic curve keypair	Key pair generation	Crypto Officer - Module-generated RSA public key: G,R - Module-generated RSA private key: G,R - Module-generated ECDSA public key: G,R - Module-generated ECDSA private key: G,R - Module-generated Diffie-Hellman public key: G,R - Module-

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						generated Diffie-Hellman private key: G,R - Module-generated EC Diffie-Hellman public key: G,R - Module-generated EC Diffie-Hellman private key: G,R - Intermediate Key Generation Value: G,E,Z
Digital signature generation	Generate RSA and ECDSA signatures	GNUTLS_FIPS140_OP_APPROVED	Message, private key, hash algorithm	Digital signature	Digital signature generation	Crypto Officer - RSA private key: W,E - ECDSA private key: W,E
Digital signature verification	Verify RSA, and ECDSA signature	GNUTLS_FIPS140_OP_APPROVED	Message, signature, public key, hash algorithm	Pass or Fail	Digital signature verification	Crypto Officer - RSA public key: W,E - ECDSA public key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Public key verification	Verify ECDSA public key	GNUTLS_FIPS140_OP_APPROVED	ECDSA public key	Pass or Fail	Public key verification	Crypto Officer - ECDSA public key: W,E
Random number generation	Generate random bitstrings	GNUTLS_FIPS140_OP_APPROVED	Number of random bits	Random bitstring	Random number generation	Crypto Officer - Entropy input: W,E - DRBG seed: G,E - DRBG internal state: G,E
Message digest	Compute SHA hashes	GNUTLS_FIPS140_OP_APPROVED	Message	Message digest	Message digest	Crypto Officer
Message authentication code (MAC)	Compute HMAC or AES CMAC	GNUTLS_FIPS140_OP_APPROVED	HMAC or AES key, message	MAC	Message authentication code (MAC)	Crypto Officer - AES key: W,E - HMAC key: W,E
Diffie-Hellman shared secret computation	Perform shared secret computation	GNUTLS_FIPS140_OP_APPROVED	DH private key, DH public key from peer	Shared secret	DH SSC for TLS	Crypto Officer - Diffie-Hellman private key: W,E - Diffie-Hellman public key: W,E - Diffie-Hellman shared secret: G,R

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
EC Diffie-Hellman shared secret computation	Perform shared secret computation	GNUTLS_FIPS140_OP_APPROVED	ECDH private key, ECDH public key from peer	Shared secret	ECDH SSC for TLS	Crypto Officer - EC Diffie-Hellman private key: W,E - EC Diffie-Hellman public key: W,E - EC Diffie-Hellman shared secret: G,R
HKDF key derivation	Perform key derivation using HKDF (in the context of TLS 1.3)	GNUTLS_FIPS140_OP_APPROVED	Shared secret	HKDF derived key	HKDF key derivation for TLS	Crypto Officer - Diffie-Hellman shared secret: W,E - EC Diffie-Hellman shared secret: W,E - HKDF derived key: G,R
Password-based key derivation	Perform password-based key derivation	GNUTLS_FIPS140_OP_APPROVED	Password, salt, iteration count	PBKDF derived key	Password-based key derivation	Crypto Officer - PBKDF derived key: G,R - PBKDF password or

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						passphrase: W,E
TLS key derivation	Perform key derivation using TLS 1.2 KDF (RFC7627)	GNUTLS_FIPS140_OP_APPROVED	TLS pre-master secret	TLS derived key	TLS key derivation	Crypto Officer - TLS Derived key: G,R - TLS pre-master secret: W,E - TLS master secret: G,W,E
Transport Layer Security (TLS) network protocol	Establish TLS channel	GNUTLS_FIPS140_OP_APPROVED	Cipher suite, Digital certificate, Public and private keys, Application data	Application data, log messages, return codes	Authenticated symmetric encryption for TLS Authenticated symmetric decryption for TLS Digital signature verification HKDF key derivation for TLS TLS key derivation DH KAS ECDH KAS	Crypto Officer - RSA public key: W,E - RSA private key: W,E - Diffie-Hellman public key: W,E - EC Diffie-Hellman public key: W,E - TLS pre-master secret: G,E - TLS master secret: G,E - TLS Derived key: G,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- HKDF derived key: G,E - KAS Domain Parameters: W,E
Show status	Show module status	Implicit (always approved)	None	Module status	None	Crypto Officer
Zeroization	Zeroize SSPs	Implicit (always approved)	Context containing SSPs	N/A	None	Crypto Officer - AES key: Z - HMAC key: Z - Module-generated RSA public key: Z - Module-generated RSA private key: Z - RSA public key: Z - RSA private key: Z - Module-generated ECDSA public key: Z - Module-generated ECDSA private key: Z - ECDSA

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

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						private key: Z - Diffie-Hellman shared secret: Z - EC Diffie-Hellman shared secret: Z - PBKDF password or passphrase: Z - PBKDF derived key: Z - Entropy input: Z - DRBG seed: Z - DRBG internal state: Z - TLS pre-master secret: Z - TLS master secret: Z - TLS Derived key: Z - HKDF derived key: Z - KAS Domain Parameters: Z - Intermediate Key

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Generation Value: Z
Self-tests	Perform self-tests	Implicit (always approved)	Module reset or API call	Pass or Fail	Symmetric encryption Symmetric decryption Authenticated symmetric encryption Authenticated symmetric decryption Message authentication code (MAC) Random number generation Key pair generation Public key verification Digital signature generation Digital signature verification ECDH SSC for TLS DH SSC for TLS HKDF key derivation for TLS TLS key derivation Password-based key derivation Message	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					digest DH KAS ECDH KAS	
Show module name and version	Show module name and version	Implicit (always approved)	None	Module name and version	None	Crypto Officer

Table 15: Approved Services

4.4 Non-Approved Services

The following table lists the non-approved services. The details of the non-approved cryptographic algorithms available in non-Approved mode can be found in Section 2.5.

Name	Description	Algorithms	Role
AES GCM when not used in the context of the TLS protocol.	Symmetric encryption; Symmetric decryption	AES GCM when not used in the context of the TLS protocol.	CO
Blowfish	Symmetric encryption; Symmetric decryption	Blowfish	CO
Camellia	Symmetric encryption; Symmetric decryption	Camellia	CO
CAST	Symmetric encryption; Symmetric decryption	CAST	CO
ChaCha20	Symmetric encryption; Symmetric decryption	ChaCha20	CO
DES	Symmetric encryption; Symmetric decryption	DES	CO
RC2, RC4	Symmetric encryption; Symmetric decryption	RC2, RC4	CO
Salsa20	Symmetric encryption; Symmetric decryption	Salsa20	CO

Name	Description	Algorithms	Role
Triple-DES	Symmetric encryption; Symmetric decryption	Triple-DES	CO
Serpent	Symmetric encryption; Symmetric decryption	Serpent	CO
SEED	Symmetric encryption; Symmetric decryption	SEED	CO
Twofish	Symmetric encryption; Symmetric decryption	Twofish	CO
Chacha20 and Poly1305	Authenticated encryption; Authenticated decryption	Chacha20 and Poly1305	CO
GOST	Symmetric encryption; Symmetric decryption; Message digest	GOST	CO
CMAC with Triple-DES	Message authentication code (MAC)	CMAC with Triple-DES	CO
GMAC	Message authentication code (MAC)	GMAC	CO
HMAC with keys smaller than 112-bits	Message authentication code (MAC)	HMAC with keys smaller than 112-bits	CO
HMAC with GOST	Message authentication code (MAC)	HMAC with GOST	CO
UMAC	Message authentication code (MAC)	UMAC	CO
MD2, MD4, MD5	Message digest; Message authentication code (MAC)	MD2, MD4, MD5	CO
RMD160	Message digest; Message authentication code (MAC)	RMD160	CO
STREEBOG	Message digest; Message authentication code (MAC)	STREEBOG	CO
Diffie-Hellman with keys generated with domain parameters other than safe primes	Key agreement; Diffie-Hellman shared secret computation	Diffie-Hellman with keys generated with domain parameters other than safe primes	CO
DSA	Key pair generation; Domain parameter generation; Digital	DSA	CO

Name	Description	Algorithms	Role
	signature generation; Digital signature verification		
ECDSA with curves not listed in CAVP certificates found in Table 6.	Key pair generation; Public key verification; Digital signature generation; Digital signature verification	ECDSA with curves not listed in CAVP certificates found in Table 6.	CO
EC Diffie-Hellman with curves not listed in CAVP certificates found in Table 6	Key agreement; EC Diffie-Hellman shared secret computation	EC Diffie-Hellman with curves not listed in CAVP certificates found in Table 6	CO
PBKDF with non-approved message digest algorithms	Password-based key derivation	PBKDF with non-approved message digest algorithms	CO
RSA with keys smaller than 2048 bits or greater than 4096 bits.	Key pair generation; Digital signature generation	RSA with keys smaller than 2048 bits or greater than 4096 bits.	CO
RSA with keys smaller than 1024 bits or greater than 4096 bits.	Digital signature verification	RSA with keys smaller than 1024 bits or greater than 4096 bits.	CO
RSA encryption and decryption with any key sizes.	Key encapsulation; Key un-encapsulation	RSA encryption and decryption with any key sizes.	CO
SHA-1	Digital signature generation	SHA-1	CO
SRP	Key agreement	SRP	CO
Non-supported cipher suites (not listed in Appendix A)	Transport Layer Security (TLS) Network Protocol	Non-supported cipher suites (not listed in Appendix A)	CO
Yarrow	Random number generation	Yarrow	CO

Table 16: 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

Each software component of the module has an associated HMAC-SHA2-256 integrity check value. The integrity of the module is verified by comparing the HMAC-SHA2-256 value calculated at run time for each software component of the module with their corresponding HMAC values calculated at build time and stored in .hmac files. A list of the software components and their corresponding .hmac files can be found in Section 2.2. The HMAC key used for the integrity test is embedded in the libgnutls shared library. 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:

This module operates in a modifiable operational environment per the FIPS 140-3 level 1 specifications. The SUSE Linux Enterprise Server operating system is used as the basis of other products. Compliance is maintained for SUSE products whenever the binary is found unchanged per the vendor affirmation from SUSE based on the allowance FIPS 140-3 management manual section 7.9.1 bullet 1 a i).

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 process is prevented.

SSPs contained within the module are protected by the process isolation and memory separation mechanisms provided by the SUSE Linux Enterprise Server operating system, and only the module has control over these SSPs.

6.2 Configuration Settings and Restrictions

The module shall be installed as stated in Section 11.1

Instrumentation tools like the ptrace system call, gdb and strace utilities, 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-tested operational environment.

7 Physical Security

This 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

All SSPs not generated by the module are provided by the calling application.

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.

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

Table 17: Storage Areas

9.2 SSP Input-Output Methods

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

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

Table 18: SSP Input-Output Methods

9.3 SSP Zeroization Methods

The memory occupied by SSPs is allocated by regular memory allocation operating system calls. The application that is acting as the CO is responsible for calling the appropriate zeroization functions provided in the module's API and listed in the table below. Calling the gnutls_global_deinit() will zeroize the SSPs stored in the TLS protocol internal state and invoke the corresponding API functions 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.

Zeroization Method	Description	Rationale	Operator Initiation
Wipe and Free memory block allocated	Zeroizes the SSPs contained within the cipher handle.	Memory occupied by SSPs is overwritten with zeroes and then it is released, which renders the SSP values irretrievable. The completion of the zeroization routine indicates that the zeroization procedure succeeded.	By calling the cipher related zeroization API function: gnutls_cipher_deinit() and gnutls_aead_cipher_deinit() for AES keys; gnutls_hmac_deinit() for HMAC keys; gnutls_privkey_deinit(), gnutls_x509_privkey_deinit(), gnutls_rsa_params_deinit() for RSA keys; gnutls_pk_params_clear() for ECDSA and ECDH keys; gnutls_dh_params_deinit() and gnutls_pk_params_clear() for DH keys; zeroize_key() for shared secrets and PBKDF derived keys; gnutls_global_deinit() for DRBG SSPs; gnutls_deinit() for TLS secrets and derived keys
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
Module Reset	De-allocates the volatile memory used to store SSPs	Volatile memory used by the module is overwritten within nanoseconds when power is removed.	By unloading and reloading the module

Table 19: SSP Zeroization Methods

9.4 SSPs

The two tables below summarize the SSPs that are used by the cryptographic services implemented in the module.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES key	AES key	AES-XTS: 128, 256 bits; Other modes: 128, 192, 256 bits - AES-XTS: 128, 256	Symmetric key - CSP			Symmetric encryption Symmetric decryption Authenticated

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		bits; Other modes: 128, 192, 256 bits				symmetric encryption Authenticated symmetric decryption Message authentication code (MAC) Authenticated symmetric encryption for TLS Authenticated symmetric decryption for TLS
HMAC key	HMAC key	112-524288 bits - 112-256 bits	Symmetric key - CSP			Message authentication code (MAC)
Module-generated RSA public key	Module-generated RSA public key	2048-4096 bits - 112-150 bits	Public key - PSP	Key pair generation		
Module-generated RSA private key	Module-generated RSA private key	2048-4096 bits - 112-150 bits	Private key - CSP	Key pair generation		
RSA public key	RSA public key	1024-4096 bits - 80-150 bits	Public key - PSP			Digital signature verification
RSA private key	RSA private key	2048-4096 bits - 112-150 bits	Private key - CSP			Digital signature generation
Module-generated ECDSA public key	Module-generated ECDSA public key	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP	Key pair generation		

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Module-generated ECDSA private key	Module-generated ECDSA private key used for: Use: Key pair generation	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP	Key pair generation		
ECDSA private key	ECDSA private key	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP			Digital signature generation Public key verification
ECDSA public key	ECDSA public key	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP			Digital signature verification Public key verification
Module-generated Diffie-Hellman public key	Module-generated Diffie-Hellman public key	MODP-2048, ffdhe2048, MODP-3072, ffdhe3072, MODP-4096, ffdhe4096, MODP-6144, ffdhe6144, MODP-8192, ffdhe8192, - 112, 128, 152, 176, 200 bits	Public key - PSP	Key pair generation		
Module-generated Diffie-Hellman private key	Module-generated Diffie-Hellman private key	MODP-2048, ffdhe2048, MODP-3072, ffdhe3072, MODP-4096, ffdhe4096, MODP-6144, ffdhe6144, MODP-8192, ffdhe8192, - 112,	Private key - CSP	Key pair generation		

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		128, 152, 176, 200 bits				
Diffie-Hellman public key	Diffie-Hellman public key	MODP-2048, ffdhe2048, MODP-3072, ffdhe3072, MODP-4096, ffdhe4096, MODP-6144, ffdhe6144, MODP-8192, ffdhe8192, - 112, 128, 152, 176, 200 bits	Public key - PSP			DH SSC for TLS
Diffie-Hellman private key	Diffie-Hellman private key	MODP-2048, ffdhe2048, MODP-3072, ffdhe3072, MODP-4096, ffdhe4096, MODP-6144, ffdhe6144, MODP-8192, ffdhe8192, - 112, 128, 152, 176, 200 bits	Private key - CSP			DH SSC for TLS
Module-generated EC Diffie-Hellman public key	Module-generated EC Diffie-Hellman public key	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP	Key pair generation		
Module-generated EC Diffie-Hellman private key	Module-generated EC Diffie-Hellman private key	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP	Key pair generation		
EC Diffie-Hellman public key	EC Diffie-Hellman public key	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP			ECDH SSC for TLS

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
EC Diffie-Hellman private key	EC Diffie-Hellman private key	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP			ECDH SSC for TLS
Diffie-Hellman shared secret	Diffie-Hellman shared secret	MODP-2048, ffdhe2048, MODP-3072, ffdhe3072, MODP-4096, ffdhe4096, MODP-6144, ffdhe6144, MODP-8192, ffdhe8192, - 112, 128, 152, 176, 200 bits	Shared Secret - CSP		DH SSC for TLS	
EC Diffie-Hellman shared secret	EC Diffie-Hellman shared secret	P-256, P-384, P-521 - 128, 192, 256 bits	Shared Secret - CSP		ECDH SSC for TLS	
PBKDF password or passphrase	PBKDF password or passphrase	8-128 bytes - N/A	Password - CSP			Password-based key derivation
PBKDF derived key	PBKDF derived key	128-4096 bits - 128-256 bits	Symmetric key - CSP			Password-based key derivation
Entropy input	Entropy input (IG D.L compliant)	192-384 bits - 192 to 384 bits	Entropy Input - CSP			Random number generation
DRBG seed	DRBG seed (IG D.L compliant)	256, 320, 384 bits - 128, 192, 256 bits	Seed - CSP	Random number generation		Random number generation
DRBG internal state	DRBG internal state consisting of V (value) and key. Compliant with IG D.L.	256, 320, 384 bits - 128, 192, 256 bits	Internal state - CSP	Random number generation		Random number generation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
TLS pre-master secret	TLS pre-master secret	DH: MODP-2048/ffdhe2048 to MODP-8192/ffdhe8192; ECDH: P-256, P-384, P-521 - DH: 112 to 200 bits; ECDH 128 to 256 bits	Shared Secret - CSP		DH SSC for TLS ECDH SSC for TLS	TLS key derivation
TLS master secret	TLS master secret	384 bits - 112 to 256 bits	Shared Secret - CSP	TLS key derivation		TLS key derivation
TLS Derived key	TLS Derived key	112 to 256 bits - 112 to 256 bits	Symmetric key - CSP	TLS key derivation		
HKDF derived key	HKDF derived key	112 to 256 bits - 112 to 256 bits	Symmetric key - CSP	HKDF key derivation for TLS		
KAS Domain Parameters	SP 800-56Arev3 Domain Parameters used for KAS	DH: MODP-2048/ffdhe2048 to MODP-8192/ffdhe8192; ECDH: P-224, P-256, P-384, P-521 - N/A	Domain parameters - PSP			DH KAS ECDH KAS
Intermediate Key Generation Value	Intermediate key pair generation value generated during key generation (SP 800-133 Rev. 2 Section 4, 5.1, and 5.2)	112-16384 bits - 112-256 bits	Intermediate value - CSP	Key pair generation		Key pair generation

Table 20: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES key	API input parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	
HMAC key	API input parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	
Module-generated RSA public key	API output parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	Module-generated RSA private key:Paired With
Module-generated RSA private key	API output parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	Module-generated RSA public key:Paired With
RSA public key	API input parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	RSA private key:Paired With
RSA private key	API input parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	RSA public key:Paired With
Module-generated ECDSA public key	API output parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	Module-generated ECDSA private key:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Module-generated ECDSA private key	API output parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	Module-generated ECDSA public key:Paired With
ECDSA private key	API input parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	ECDSA public key:Paired With
ECDSA public key	API input parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	ECDSA private key:Paired With
Module-generated Diffie-Hellman public key	API output parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	Module-generated Diffie-Hellman private key:Paired With
Module-generated Diffie-Hellman private key	API output parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	Module-generated Diffie-Hellman public key:Paired With
Diffie-Hellman public key	API input parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	Diffie-Hellman private key:Paired With Diffie-Hellman shared secret:Establishes TLS pre-master secret:Establishes
Diffie-Hellman private key	API input parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	Diffie-Hellman public key:Paired With Diffie-Hellman shared secret:Establishes TLS pre-master secret:Establishes

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Module-generated EC Diffie-Hellman public key	API output parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	Module-generated EC Diffie-Hellman private key:Paired With
Module-generated EC Diffie-Hellman private key	API output parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	Module-generated EC Diffie-Hellman public key:Paired With
EC Diffie-Hellman public key	API input parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	EC Diffie-Hellman private key:Paired With EC Diffie-Hellman shared secret:Establishes TLS pre-master secret:Establishes
EC Diffie-Hellman private key	API input parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	EC Diffie-Hellman public key:Paired With EC Diffie-Hellman shared secret:Establishes TLS pre-master secret:Establishes
Diffie-Hellman shared secret	API input parameters API output parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	Diffie-Hellman public key:Established By Diffie-Hellman private key:Established By
EC Diffie-Hellman shared secret	API input parameters API output parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	EC Diffie-Hellman public key:Established By EC Diffie-Hellman private key:Established By

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
PBKDF password or passphrase	API input parameters	RAM:Plaintext	From service invocation to service completion	Automatic	PBKDF derived key:Derives
PBKDF derived key	API output parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	PBKDF password or passphrase:Derived From
Entropy input		RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset Automatic	DRBG seed:Derives
DRBG seed		RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset Automatic	Entropy input:Derived From DRBG internal state:Derives
DRBG internal state		RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset Automatic	DRBG seed:Derived From Module-generated RSA public key:Derives Module-generated RSA private key:Derives Module-generated ECDSA public key:Derives Module-generated ECDSA private key:Derives Module-generated Diffie-Hellman public key:Derives Module-generated Diffie-Hellman private key:Derives Module-generated EC

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
					Diffie-Hellman public key:Derives Module-generated EC Diffie-Hellman private key:Derives
TLS pre-master secret	API input parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated	Diffie-Hellman private key:Established By Diffie-Hellman public key:Established By EC Diffie-Hellman private key:Established By EC Diffie-Hellman public key:Established By TLS master secret:Derives
TLS master secret		RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	TLS pre-master secret:Derived From TLS Derived key:Derives
TLS Derived key	API output parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	TLS master secret:Derived From
HKDF derived key	API output parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated	Diffie-Hellman shared secret:Derived From EC Diffie-Hellman shared secret:Derived From
KAS Domain Parameters		RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Intermediate Key Generation Value		RAM:Plaintext	From service invocation to service completion	Automatic	Module-generated RSA public key:Derived From Module-generated RSA private key:Derived From Module-generated ECDSA public key:Derived From Module-generated ECDSA private key:Derived From Module-generated Diffie-Hellman public key:Derived From Module-generated Diffie-Hellman private key:Derived From Module-generated EC Diffie-Hellman public key:Derived From Module-generated EC Diffie-Hellman private key:Derived From

Table 21: SSP Table 2

10 Self-Tests

The module performs the pre-operational self-test and CASTs automatically when the module is loaded into memory. The pre-operational self-test ensure that the module is not corrupted, and the CASTs ensure that the cryptographic algorithms work as expected. While the module is executing the self-tests, services are not available, and input and output are inhibited. The module is not available for use by the calling application until the pre-operational tests and 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.

10.1 Pre-Operational Self-Tests

The module performs the integrity test using HMAC-SHA2-256. The HMAC-SHA2-256 CAST is performed before the integrity test. The details of the integrity test are provided in Section 5.1.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A2987)	SHA2-256	MAC Tag Verification	SW/FW Integrity	Module becomes operational	N/A
HMAC-SHA2-256 (A2992)	SHA2-256	MAC Tag Verification	SW/FW Integrity	Module becomes operational	N/A
HMAC-SHA2-256 (A2998)	SHA2-256	MAC Tag Verification	SW/FW Integrity	Module becomes operational	N/A
HMAC-SHA2-256 (A3007)	SHA2-256	MAC Tag Verification	SW/FW Integrity	Module becomes operational	N/A

Table 22: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

The table below specifies all the conditional self-tests performed by the module. These include both Cryptographic Algorithm Self-Tests (CASTs) and Pair-wise Consistency Tests (PCTs).

The CASTs are performed in the form of the Known Answer Tests (KATs) and are run prior to performing the integrity test. A KAT includes the comparison of a calculated output with an expected known answer, hard coded as part of the test vectors used in the test. If the values do not match, the KAT fails.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC (A3007) - Encrypt	128/256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A2992) - Encrypt	128/256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A2985) - Encrypt	128/256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A2987) - Encrypt	128/256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A2996) - Encrypt	128/256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A2997) - Encrypt	128/256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A3004) - Encrypt	128/256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A2984) - Encrypt	128/256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A2986) - Encrypt	128/256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC (A3007) - Decrypt	128/256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A2992) - Decrypt	128/256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A2985) - Decrypt	128/256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A2987) - Decrypt	128/256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A2996) - Decrypt	128/256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A2997) - Decrypt	128/256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A3004) - Decrypt	128/256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A2984) - Decrypt	128/256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CBC (A2986) - Decrypt	128/256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-GCM (A3007) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A2992) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A2985) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A2987) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A2996) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A2997) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A3004) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A2984) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A2986) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-GCM (A3007) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A2992) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A2985) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A2987) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A2996) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A2997) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A3004) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A2984) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-GCM (A2986) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CFB8 (A2995) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CFB8 (A2989) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CFB8 (A2990) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CFB8 (A2995) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CFB8 (A2989) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CFB8 (A2990) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-CMAC (A2992) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
AES-CMAC (A2987) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
AES-CMAC (A2996) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CMAC (A3004) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
AES-CMAC (A2984) - Encrypt	256-bit key, encrypt	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
AES-CMAC (A2992) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
AES-CMAC (A2987) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
AES-CMAC (A2996) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
AES-CMAC (A3004) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
AES-CMAC (A2984) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
AES-XTS Testing Revision 2.0 (A2993) - Decrypt	256-bit key, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before the integrity test
AES-XTS Testing Revision 2.0	256-bit key, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
(A2993) - Encrypt						before the integrity test
KAS-FFC-SSC Sp800-56Ar3 (A2992)	ffdhe3072	KAT	CAST	Module becomes operational	Shared secret computation	Test runs at power-on before the integrity test
KAS-ECC-SSC Sp800-56Ar3 (A2992)	P-256 curve	KAT	CAST	Module becomes operational	Shared secret computation	Test runs at power-on before the integrity test
Counter DRBG (A2992)	256-bit key, no DF, no PR	KAT	CAST	Module becomes operational	Compliant with SP 800-90Ar1	Test runs at power-on before the integrity test
ECDSA SigGen (FIPS186-4) (A2992)	P-256 with SHA2-256, P-384 with SHA2-384, P-521 with SHA2-512	KAT	CAST	Module becomes operational	Digital signature generation	Test runs at power-on before the integrity test
ECDSA SigVer (FIPS186-4) (A2992)	P-256 with SHA2-256, P-384 with SHA2-384, P-521 with SHA2-512	KAT	CAST	Module becomes operational	Digital signature verification	Test runs at power-on before the integrity test
KDA HKDF Sp800-56Cr1 (A2991)	SHA2-256	KAT	CAST	Module becomes operational	Shared secret key derivation	Test runs at power-on before the integrity test
HMAC-SHA-1 (A3007)	SHA-1	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA-1 (A2992)	SHA-1	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA-1 (A2998)	SHA-1	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA-1 (A2987)	SHA-1	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-224 (A3007)	SHA2-224	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-224 (A2992)	SHA2-224	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-224 (A2998)	SHA2-224	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-224 (A2987)	SHA2-224	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-256 (A3007)	SHA2-256	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-256 (A2992)	SHA2-256	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-256 (A2998)	SHA2-256	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-256 (A2987)	SHA2-256	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-384 (A3007)	SHA2-384	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-384 (A2992)	SHA2-384	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-384 (A2998)	SHA2-384	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-384 (A2987)	SHA2-384	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-512 (A3007)	SHA2-512	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-512 (A2992)	SHA2-512	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-512 (A2998)	SHA2-512	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test
HMAC-SHA2-512 (A2987)	SHA2-512	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
PBKDF (A2992)	HMAC-SHA2-256	KAT	CAST	Module becomes operational	Password-based key derivation	Test runs at power-on before the integrity test
RSA SigGen (FIPS186-4) (A2992)	2048-bit key and SHA2-256	KAT	CAST	Module becomes operational	Digital signature generation	Test runs at power-on before the integrity test
RSA SigVer (FIPS186-4) (A2992)	2048-bit key and SHA2-256	KAT	CAST	Module becomes operational	Digital signature verification	Test runs at power-on before the integrity test
SHA3-224 (A2988)		KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA3-224 (A2994)		KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA3-256 (A2988)		KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA3-256 (A2994)		KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA3-384 (A2988)		KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA3-384 (A2994)		KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA3-512 (A2988)		KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
SHA3-512 (A2994)		KAT	CAST	Module becomes operational	Message digest	Test runs at power-on before the integrity test
TLS v1.2 KDF RFC7627 (A2992)	SHA2-256	KAT	CAST	Module becomes operational	Industry-based TLS v1.2 KDF key derivation	Test runs at power-on before the integrity test
ECDSA KeyGen (FIPS186-4) (A2992)	SHA2-256	PCT	PCT	Successful key pair generation	Signature generation & verification	Key pair generation
RSA KeyGen (FIPS186-4) (A2992)	SHA2-256	PCT	PCT	Successful key pair generation	Signature generation & verification	Key pair generation
Safe Primes Key Generation	SHA2-256	PCT	PCT	Successful key pair generation	Signature generation & verification	Key pair generation

Table 23: Conditional Self-Tests

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A2987)	MAC Tag Verification	SW/FW Integrity	On Demand	Manually
HMAC-SHA2-256 (A2992)	MAC Tag Verification	SW/FW Integrity	On Demand	Manually
HMAC-SHA2-256 (A2998)	MAC Tag Verification	SW/FW Integrity	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A3007)	MAC Tag Verification	SW/FW Integrity	On Demand	Manually

Table 24: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC (A3007) - Encrypt	KAT	CAST	On Demand	Manually
AES-CBC (A2992) - Encrypt	KAT	CAST	On Demand	Manually
AES-CBC (A2985) - Encrypt	KAT	CAST	On Demand	Manually
AES-CBC (A2987) - Encrypt	KAT	CAST	On Demand	Manually
AES-CBC (A2996) - Encrypt	KAT	CAST	On Demand	Manually
AES-CBC (A2997) - Encrypt	KAT	CAST	On Demand	Manually
AES-CBC (A3004) - Encrypt	KAT	CAST	On Demand	Manually
AES-CBC (A2984) - Encrypt	KAT	CAST	On Demand	Manually
AES-CBC (A2986) - Encrypt	KAT	CAST	On Demand	Manually
AES-CBC (A3007) - Decrypt	KAT	CAST	On Demand	Manually
AES-CBC (A2992) - Decrypt	KAT	CAST	On Demand	Manually
AES-CBC (A2985) - Decrypt	KAT	CAST	On Demand	Manually
AES-CBC (A2987) - Decrypt	KAT	CAST	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC (A2996) - Decrypt	KAT	CAST	On Demand	Manually
AES-CBC (A2997) - Decrypt	KAT	CAST	On Demand	Manually
AES-CBC (A3004) - Decrypt	KAT	CAST	On Demand	Manually
AES-CBC (A2984) - Decrypt	KAT	CAST	On Demand	Manually
AES-CBC (A2986) - Decrypt	KAT	CAST	On Demand	Manually
AES-GCM (A3007) - Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A2992) - Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A2985) - Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A2987) - Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A2996) - Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A2997) - Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A3004) - Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A2984) - Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A2986) - Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A3007) - Decrypt	KAT	CAST	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-GCM (A2992) - Decrypt	KAT	CAST	On Demand	Manually
AES-GCM (A2985) - Decrypt	KAT	CAST	On Demand	Manually
AES-GCM (A2987) - Decrypt	KAT	CAST	On Demand	Manually
AES-GCM (A2996) - Decrypt	KAT	CAST	On Demand	Manually
AES-GCM (A2997) - Decrypt	KAT	CAST	On Demand	Manually
AES-GCM (A3004) - Decrypt	KAT	CAST	On Demand	Manually
AES-GCM (A2984) - Decrypt	KAT	CAST	On Demand	Manually
AES-GCM (A2986) - Decrypt	KAT	CAST	On Demand	Manually
AES-CFB8 (A2995) - Encrypt	KAT	CAST	On Demand	Manually
AES-CFB8 (A2989) - Encrypt	KAT	CAST	On Demand	Manually
AES-CFB8 (A2990) - Encrypt	KAT	CAST	On Demand	Manually
AES-CFB8 (A2995) - Decrypt	KAT	CAST	On Demand	Manually
AES-CFB8 (A2989) - Decrypt	KAT	CAST	On Demand	Manually
AES-CFB8 (A2990) - Decrypt	KAT	CAST	On Demand	Manually
AES-CMAC (A2992) - Encrypt	KAT	CAST	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CMAC (A2987) - Encrypt	KAT	CAST	On Demand	Manually
AES-CMAC (A2996) - Encrypt	KAT	CAST	On Demand	Manually
AES-CMAC (A3004) - Encrypt	KAT	CAST	On Demand	Manually
AES-CMAC (A2984) - Encrypt	KAT	CAST	On Demand	Manually
AES-CMAC (A2992) - Decrypt	KAT	CAST	On Demand	Manually
AES-CMAC (A2987) - Decrypt	KAT	CAST	On Demand	Manually
AES-CMAC (A2996) - Decrypt	KAT	CAST	On Demand	Manually
AES-CMAC (A3004) - Decrypt	KAT	CAST	On Demand	Manually
AES-CMAC (A2984) - Decrypt	KAT	CAST	On Demand	Manually
AES-XTS Testing Revision 2.0 (A2993) - Decrypt	KAT	CAST	On Demand	Manually
AES-XTS Testing Revision 2.0 (A2993) - Encrypt	KAT	CAST	On Demand	Manually
KAS-FFC-SSC Sp800-56Ar3 (A2992)	KAT	CAST	On Demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A2992)	KAT	CAST	On Demand	Manually
Counter DRBG (A2992)	KAT	CAST	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
ECDSA SigGen (FIPS186-4) (A2992)	KAT	CAST	On Demand	Manually
ECDSA SigVer (FIPS186-4) (A2992)	KAT	CAST	On Demand	Manually
KDA HKDF Sp800-56Cr1 (A2991)	KAT	CAST	On Demand	Manually
HMAC-SHA-1 (A3007)	KAT	CAST	On Demand	Manually
HMAC-SHA-1 (A2992)	KAT	CAST	On Demand	Manually
HMAC-SHA-1 (A2998)	KAT	CAST	On Demand	Manually
HMAC-SHA-1 (A2987)	KAT	CAST	On Demand	Manually
HMAC-SHA2-224 (A3007)	KAT	CAST	On Demand	Manually
HMAC-SHA2-224 (A2992)	KAT	CAST	On Demand	Manually
HMAC-SHA2-224 (A2998)	KAT	CAST	On Demand	Manually
HMAC-SHA2-224 (A2987)	KAT	CAST	On Demand	Manually
HMAC-SHA2-256 (A3007)	KAT	CAST	On Demand	Manually
HMAC-SHA2-256 (A2992)	KAT	CAST	On Demand	Manually
HMAC-SHA2-256 (A2998)	KAT	CAST	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A2987)	KAT	CAST	On Demand	Manually
HMAC-SHA2-384 (A3007)	KAT	CAST	On Demand	Manually
HMAC-SHA2-384 (A2992)	KAT	CAST	On Demand	Manually
HMAC-SHA2-384 (A2998)	KAT	CAST	On Demand	Manually
HMAC-SHA2-384 (A2987)	KAT	CAST	On Demand	Manually
HMAC-SHA2-512 (A3007)	KAT	CAST	On Demand	Manually
HMAC-SHA2-512 (A2992)	KAT	CAST	On Demand	Manually
HMAC-SHA2-512 (A2998)	KAT	CAST	On Demand	Manually
HMAC-SHA2-512 (A2987)	KAT	CAST	On Demand	Manually
PBKDF (A2992)	KAT	CAST	On Demand	Manually
RSA SigGen (FIPS186-4) (A2992)	KAT	CAST	On Demand	Manually
RSA SigVer (FIPS186-4) (A2992)	KAT	CAST	On Demand	Manually
SHA3-224 (A2988)	KAT	CAST	On Demand	Manually
SHA3-224 (A2994)	KAT	CAST	On Demand	Manually
SHA3-256 (A2988)	KAT	CAST	On Demand	Manually
SHA3-256 (A2994)	KAT	CAST	On Demand	Manually
SHA3-384 (A2988)	KAT	CAST	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
SHA3-384 (A2994)	KAT	CAST	On Demand	Manually
SHA3-512 (A2988)	KAT	CAST	On Demand	Manually
SHA3-512 (A2994)	KAT	CAST	On Demand	Manually
TLS v1.2 KDF RFC7627 (A2992)	KAT	CAST	On Demand	Manually
ECDSA KeyGen (FIPS186-4) (A2992)	PCT	PCT	On Demand	Manually
RSA KeyGen (FIPS186-4) (A2992)	PCT	PCT	On Demand	Manually
Safe Primes Key Generation	PCT	PCT	On Demand	Manually

Table 25: Conditional Periodic Information

10.4 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.

The following table shows the error codes and the corresponding condition:

Name	Description	Conditions	Recovery Method	Indicator
Error	General-purpose error state.	Any integrity test or pre-operational self-test fails at start-up Any newly generated RSA, ECDSA, DH, or ECDH key pair fails a PCT Module is in error state and a caller requests	Restart	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_ERROR, module does not function

Name	Description	Conditions	Recovery Method	Indicator
		cryptographic operations		

Table 26: Error States

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 per-operational self-test and the CASTs to recover from these errors. If failures persist, the module must be re-installed.

A completed list of the error codes can be found in Appendix C “Error Codes and Descriptions” in the gnutls.pdf provided with the module's code.

10.5 Operator Initiation of Self-Tests

The module provides the Self-Test service to perform self-tests on demand which includes the pre-operational test (i.e., integrity test) and 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

11.1.1 Module Installation

The Crypto Officer can install the RPM packages containing the module as listed in Table 28 using the zypper tool. The integrity of the RPM package is automatically verified during the installation, and the Crypto Officer shall not install the RPM package if there is any integrity error.

11.1.2 Operating Environment Configuration

The operating environment needs to be configured to support FIPS, so the following steps shall be performed with the root privilege:

1. Install the dracut-fips RPM package:

```
# zypper install dracut-fips
```

2. Recreate the INITRAMFS image:

```
# dracut -f
```

3. After regenerating the initrd, the Crypto Officer has to append the following parameter in the /etc/default/grub configuration file in the GRUB_CMDLINE_LINUX_DEFAULT line:

```
fips=1
```

4. After editing the configuration file, please run the following command to change the setting in the boot loader:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

If /boot or /boot/efi resides on a separate partition, the kernel parameter boot=<partition of /boot or /boot/efi> must be supplied. The partition can be identified with the command "df /boot" or "df /boot/efi" respectively. For example:

```
# df /boot
Filesystem     1K-blocks     Used     Available     Use%     Mounted on
/dev/sda1      233191        30454    190296        14%              /boot
```

The partition of /boot is located on /dev/sda1 in this example. Therefore, the following string needs to be appended in the aforementioned grub file:

```
"boot=/dev/sda1"
```

5. Reboot to apply these settings.

Now, the operating environment is configured to support FIPS operation. The Crypto Officer should check the existence of the file /proc/sys/crypto/fips_enabled, and verify it contains a numeric value "1". If the file does not exist or does not contain "1", the operating environment is not configured to support FIPS and the module will not operate as a FIPS-validated module properly.

11.1.3 Module Installation for Vendor Affirmed Platforms

The following table includes the information on the module installation process for the vendor affirmed platforms that are listed in Section 2.2.

Product	Link
SUSE Linux Enterprise Micro 5.3	https://documentation.suse.com/sle-micro/5.3/single-html/SLE-Micro-security/#sec-fips-slemicro-install
SUSE Linux Enterprise Server for SAP 15SP4	https://documentation.suse.com/sles/15-SP4/html/SLES-all/book-security.html
SUSE Linux Enterprise Base Container Image 15SP4	https://documentation.suse.com/smart/linux/html/concept-bci/index.html
SUSE Linux Enterprise Desktop 15SP4	https://documentation.suse.com/sled/15-SP4/html/SLED-all/book-security.html
SUSE Linux Enterprise Real Time 15SP4	https://documentation.suse.com/sle-rt/15-SP4/

Table 27 - Installation for Vendor Affirmed Platforms

Note: Per section 7.9 in the FIPS 140-3 Management Manual, the Cryptographic Module Validation Program (CMVP) makes no statement as to the correct operation of the module or the security strengths of the generated keys when this module is ported and executed in an operational environment not listed on the validation certificate.

11.2 Administrator Guidance

The binaries of the module are contained in the RPM packages for delivery. The Crypto Officer shall follow section 11.1.1 and 11.1.2 to configure the operational environment and install the module to be operated as a FIPS 140-3 validated module.

Table 28 lists the RPM packages that contain the FIPS validated module. The "Show module name and version" service returns the value "GnuTLS version 3.7.3-150400.4.35.1", which matches the version included in the RPM package filenames, and map to version 1.1 of the cryptographic module.

Processor Architecture	RPM Packages
Intel 64-bit	libgnutls30-3.7.3-150400.4.35.1.x86_64.rpm libnettle8-3.7.3-150400.2.21.x86_64.rpm libhogweed6-3.7.3-150400.2.21.x86_64.rpm libgmp10-6.1.2-4.9.1.x86_64.rpm
AMD 64-bit	libgnutls30-3.7.3-150400.4.35.1.x86_64.rpm libnettle8-3.7.3-150400.2.21.x86_64.rpm libhogweed6-3.7.3-150400.2.21.x86_64.rpm libgmp10-6.1.2-4.9.1.x86_64.rpm
IBM z15	libgnutls30-3.7.3-150400.4.35.1.s390x.rpm libnettle8-3.7.3-150400.2.21.s390x.rpm libhogweed6-3.7.3-150400.2.21.s390x.rpm libgmp10-6.1.2-4.9.1.s390x.rpm
ARMv8 64-bit	libgnutls30-3.7.3-150400.4.35.1.aarch64.rpm libnettle8-3.7.3-150400.2.21.aarch64.rpm libhogweed6-3.7.3-150400.2.21.aarch64.rpm libgmp10-6.1.2-4.9.1.aarch64.rpm
IBM Power10 64-bit	libgnutls30-3.7.3-150400.4.35.1.ppc64le.rpm libnettle8-3.7.3-150400.2.21.ppc64le.rpm libhogweed6-3.7.3-150400.2.21.ppc64le.rpm libgmp10-6.1.2-4.9.1.ppc64le.rpm

Table 28 - RPM packages

11.3 Non-Administrator Guidance

There is no 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 needed.

11.5 Additional Information

The module cannot use the following environment variables:

- GNUTLS_NO_EXPLICIT_INIT
- GNUTLS_SKIP_FIPS_INTEGRITY_CHECKS

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

- gnutls_crypto_register_cipher
- gnutls_crypto_register_aead_cipher
- gnutls_crypto_register_mac
- gnutls_crypto_register_digest
- gnutls_privkey_import_ext4

12 Mitigation of Other Attacks

This module is not designed to mitigate any other attacks.

Appendix A. TLS Cipher Suites

The module supports the following cipher suites for the TLS protocol version 1.0, 1.1, 1.2 and 1.3, compliant with section 3.3.1 of [SP 800-52 Rev. 2]. 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

Cipher Suite	ID	Reference
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x23 }	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x24 }	RFC5289
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x25 }	RFC5289
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x26 }	RFC5289
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x27 }	RFC5289
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x28 }	RFC5289
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x29 }	RFC5289
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x2A }	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	{ 0xC0, 0x2B }	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	{ 0xC0, 0x2C }	RFC5289
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

Table 29 - TLS Cipher Suites

Appendix B. Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CPACF	Central Processor Assist for Cryptographic Function
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DF	Derivation Function
DSA	Digital Signature Algorithm
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards Publication
FSM	Finite State Model
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
KAS	Key Agreement Schema
KAT	Known Answer Test
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
OFB	Output Feedback
O/S	Operating System
PAA	Processor Algorithm Acceleration
PAI	Processor Algorithm Implementation
PR	Prediction Resistance

PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, Adleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SSH	Secure Shell
SSP	Sensitive Security Parameter
TDES	Triple-DES
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://doi.org/10.6028/NIST.FIPS.140-3
FIPS 140-3 IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program** March 2024 https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements
SP 800-38A	**Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 https://doi.org/10.6028/NIST.SP.800-38A
SP 800-38B	**Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 https://doi.org/10.6028/NIST.SP.800-38B
SP 800-38C	**Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality** May 2004 https://doi.org/10.6028/NIST.SP.800-38C
SP 800-38D	**Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 https://doi.org/10.6028/NIST.SP.800-38D
SP 800-38E	**Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality of Storage Devices** January 2010 https://doi.org/10.6028/NIST.SP.800-38E
FIPS 180-4	**Secure Hash Standard (SHS)** August 2015 https://doi.org/10.6028/NIST.FIPS.180-4
FIPS 202	**SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions** August 2015 https://doi.org/10.6028/NIST.FIPS.202
FIPS 198-1	**The Keyed-Hash Message Authentication Code (HMAC)** July 2008 https://doi.org/10.6028/NIST.FIPS.198-1
FIPS 186-4	**Digital Signature Standard (DSS)** July 2013 https://doi.org/10.6028/NIST.FIPS.186-4

SP 800-132	**Recommendation for Password-Based Key Derivation Part 1: Storage Applications** December 2010 https://doi.org/10.6028/NIST.SP.800-132
SP 800-56A Rev. 3	**Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography** April 2018 https://doi.org/10.6028/NIST.SP.800-56Ar3
SP 800-56C Rev. 2	**Recommendation for Key-Derivation Methods in Key-Establishment Schemes** August 2020 https://doi.org/10.6028/NIST.SP.800-56Cr2
SP 800-135 Rev. 1	**Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 https://doi.org/10.6028/NIST.SP.800-135r1
SP 800-90A Rev. 1	**Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 https://doi.org/10.6028/NIST.SP.800-90Ar1
SP 800-90B	**Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://doi.org/10.6028/NIST.SP.800-90B
SP 800-133 Rev. 2	**Recommendation for Cryptographic Key Generation** June 2020 https://doi.org/10.6028/NIST.SP.800-133r2
SP 800-52 Rev. 2	**Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations** August 2019 https://doi.org/10.6028/NIST.SP.800-52r2
SP 800-131A Rev. 2	**Transitioning the Use of Cryptographic Algorithms and Key Lengths** March 2019 https://doi.org/10.6028/NIST.SP.800-131Ar2
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 7627	**Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension** September 2015 https://www.ietf.org/rfc/rfc7627.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