Arista Networks, Inc.

Arista Crypto Module Lvl2 [Software, Software IPsec]

FIPS 140-3 Non-Proprietary Security Policy

Document Version: v1.4
Date: January 26, 2026

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 v1.0 of the Arista Networks Inc. Arista Crypto Module Lvl2 [Software, Software IPsec]. 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 2 module.

1.2 Security Levels

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

Table 1: Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The Arista Crypto Module Lvl2 v1.0 [Software, Software IPsec] (hereafter referred to as “the module”) is a Software Multichip standalone cryptographic module. The module provides cryptographic services to applications running in the user space of the underlying operating system through a C language Application Program Interface (API).

Module Type: Software

Module Embodiment: MultiChipStand

Cryptographic Boundary:

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

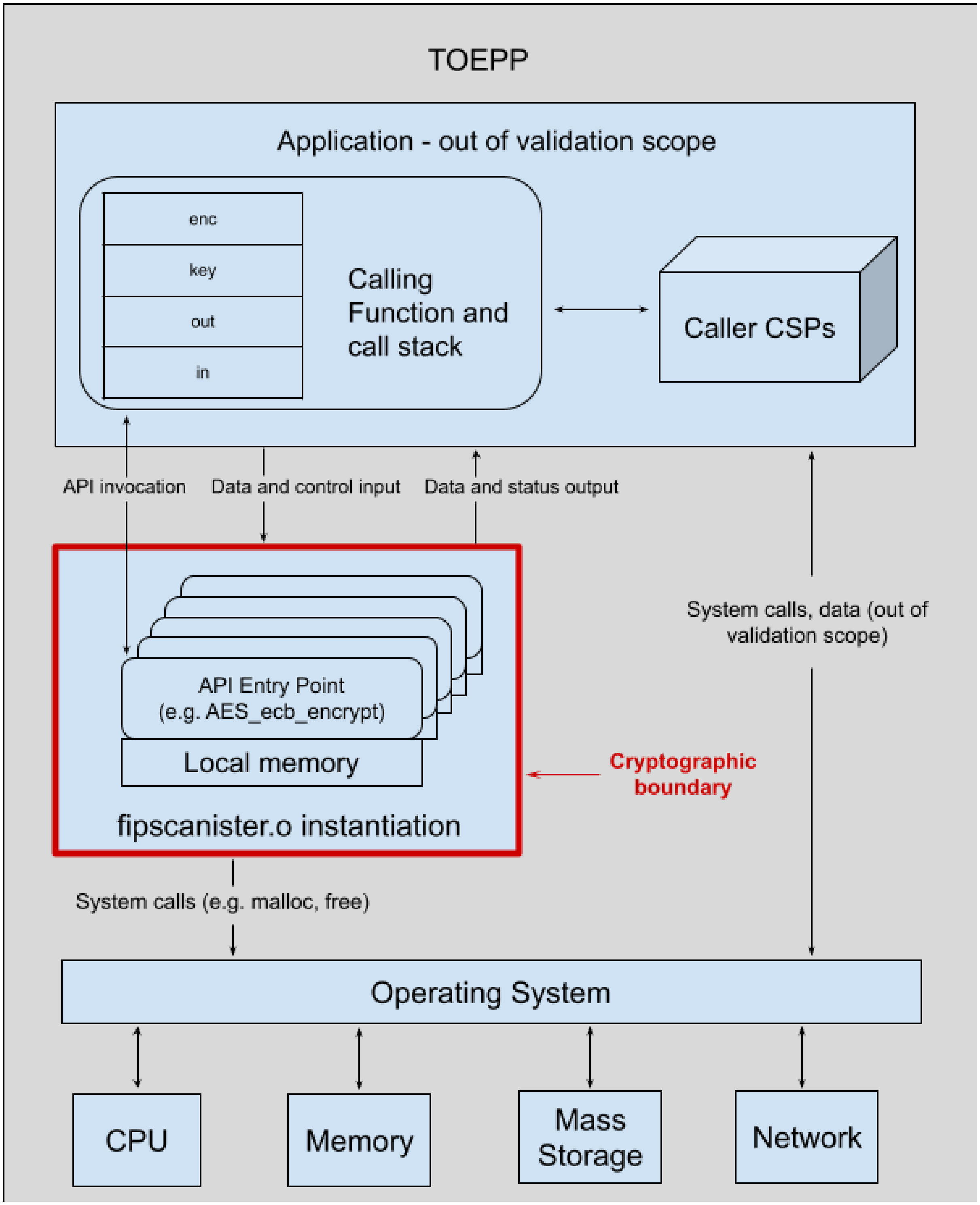

Figure 1: Block Diagram depicting the cryptographic boundary (in red rectangle) and data flow between the module interfaces and operator. The boundary also includes the instantiation of the cryptographic module in memory.

The module components consist of the fipscanister.o file in executable form. The fipscanister.o is delivered in the product by statically linking to libcrypto.so. The Module performs no communications other than with the calling application (the process that invokes the Module services) and the OS syslog. The boundary also includes the instantiation of the module saved in memory.

2.2 Tested and Vendor Affirmed Module Version and Identification

The module operates in a modifiable operational environment. The module runs on an EOS, based on a general-purpose operating system. The module executes on the hardware platform listed in Table 3. The module does not support concurrent operators.

Tested Module Identification – Hardware:

N/A for this module.

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

The module has been tested on the platforms indicated in the following table, with the corresponding module variants and configuration options with and without PAA.

Package or File Name	Software/ Firmware Version	Features	Integrity Test
fipscanister.o	v1.0	None	Message authentication with HMAC-SHA2-256

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

Code Sets:

The module consists of executable code in the form of fipscanister.o. The compiler used to generate the executable code is gcc.

Tested Module Identification – Hybrid Disjoint Hardware:

N/A for this module.

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
EOSv4	Arista AWE-5510	Intel Xeon D-2798NX	Yes	None	v1.0
EOSv4	Arista AWE-5510	Intel Xeon D-2798NX	No	None	v1.0

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

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

The vendor claims the following platforms to be vendor affirmed - that is, the module functions the same way and provides the same services on the following systems:

Operating System	Hardware Platform
EOSv4	Arista AWE-5310
EOSv4	Arista 7300-SUP
EOSv4	Arista 7300-SUP-D
EOSv4	Arista 7368-SUP
EOSv4	Arista 7368-SUP-D

Operating System	Hardware Platform
EOSv4	Arista 7388-SUP
EOSv4	Arista 7388-SUP-D
EOSv4	Arista CCS-710P-12
EOSv4	Arista CCS-710P-16P
EOSv4	Arista CCS-720DF-48Y
EOSv4	Arista CCS-720DF-48Y-2
EOSv4	Arista CCS-720DP-24S
EOSv4	Arista CCS-720DP-24S-2
EOSv4	Arista CCS-720DP-24ZS
EOSv4	Arista CCS-720DP-24ZS-2
EOSv4	Arista CCS-720DP-48S
EOSv4	Arista CCS-720DP-48S-2
EOSv4	Arista CCS-720DT-24S
EOSv4	Arista CCS-720DT-24S-2
EOSv4	Arista CCS-720DT-48S
EOSv4	Arista CCS-720DT-48S-2
EOSv4	Arista CCS-720XP-24Y6
EOSv4	Arista CCS-720XP-24ZY4
EOSv4	Arista CCS-720XP-48TXH-2C-S
EOSv4	Arista CCS-720XP-48Y6
EOSv4	Arista CCS-720XP-48ZC2
EOSv4	Arista CCS-720XP-48ZXC2
EOSv4	Arista CCS-720XP-96ZC2
EOSv4	Arista CCS-722XPM-48Y4
EOSv4	Arista CCS-722XPM-48ZY8
EOSv4	Arista CCS-750-Sup100
EOSv4	Arista CCS-750-SUP100
EOSv4	Arista CCS-750-SUP25
EOSv4	Arista CCS-750-Sup25
EOSv4	Arista DCS-7010T-48
EOSv4	Arista DCS-7010T-48-DC
EOSv4	Arista DCS-7010TX-48
EOSv4	Arista DCS-7010TX-48-DC
EOSv4	Arista DCS-7010TX-48C
EOSv4	Arista DCS-7020SR-24C2
EOSv4	Arista DCS-7020SR-32C2
EOSv4	Arista DCS-7020SRG-24C2
EOSv4	Arista DCS-7020TR-48
EOSv4	Arista DCS-7020TRA-48
EOSv4	Arista DCS-7050CX3-32C
EOSv4	Arista DCS-7050CX3-32S
EOSv4	Arista DCS-7050CX3-32S-SSD
EOSv4	Arista DCS-7050CX3M-32S
EOSv4	Arista DCS-7050CX4-24D8
EOSv4	Arista DCS-7050CX4-40D
EOSv4	Arista DCS-7050CX4-48D8
EOSv4	Arista DCS-7050CX4M-48D8
EOSv4	Arista DCS-7050DX4-32S
EOSv4	Arista DCS-7050DX4M-32S
EOSv4	Arista DCS-7050PX4-32S
EOSv4	Arista DCS-7050QX-32
EOSv4	Arista DCS-7050QX-32S

Operating System	Hardware Platform
EOSv4	Arista DCS-7050QX2-32S
EOSv4	Arista DCS-7050SDX4-48D8
EOSv4	Arista DCS-7050SPX4-48D8
EOSv4	Arista DCS-7050SX-128
EOSv4	Arista DCS-7050SX-64
EOSv4	Arista DCS-7050SX-72
EOSv4	Arista DCS-7050SX-72Q
EOSv4	Arista DCS-7050SX-96
EOSv4	Arista DCS-7050SX2-128
EOSv4	Arista DCS-7050SX2-72Q
EOSv4	Arista DCS-7050SX3-48C8
EOSv4	Arista DCS-7050SX3-48C8C
EOSv4	Arista DCS-7050SX3-48YC12
EOSv4	Arista DCS-7050SX3-48YC8
EOSv4	Arista DCS-7050SX3-48YC8C
EOSv4	Arista DCS-7050SX3-96YC8
EOSv4	Arista DCS-7050TX-128
EOSv4	Arista DCS-7050TX-48
EOSv4	Arista DCS-7050TX-64
EOSv4	Arista DCS-7050TX-72
EOSv4	Arista DCS-7050TX-72Q
EOSv4	Arista DCS-7050TX-96
EOSv4	Arista DCS-7050TX2-128
EOSv4	Arista DCS-7050TX3-48C8
EOSv4	Arista DCS-7060CX-32C
EOSv4	Arista DCS-7060CX-32S
EOSv4	Arista DCS-7060CX2-32S
EOSv4	Arista DCS-7060DX4-32
EOSv4	Arista DCS-7060DX5-64
EOSv4	Arista DCS-7060DX5-64E
EOSv4	Arista DCS-7060DX5-64S
EOSv4	Arista DCS-7060PX4-32
EOSv4	Arista DCS-7060PX5-64
EOSv4	Arista DCS-7060PX5-64E
EOSv4	Arista DCS-7060PX5-64S
EOSv4	Arista DCS-7060SX2-48YC6
EOSv4	Arista DCS-7130-16G3S
EOSv4	Arista DCS-7130-48EHS
EOSv4	Arista DCS-7130-48G3S
EOSv4	Arista DCS-7130-48LAS
EOSv4	Arista DCS-7130-48LBAS
EOSv4	Arista DCS-7130-48LBS
EOSv4	Arista DCS-7130-96LAS
EOSv4	Arista DCS-7130-96LBAS
EOSv4	Arista DCS-7130-96LBS
EOSv4	Arista DCS-7130-96LS
EOSv4	Arista DCS-7130-96SS
EOSv4	Arista DCS-7130LBR-48S6QD
EOSv4	Arista DCS-7132LB-48Y4C
EOSv4	Arista DCS-7132LB-48Y4CDC
EOSv4	Arista DCS-7132LN-48Y4C
EOSv4	Arista DCS-7135LB-48Y4C

Operating System	Hardware Platform
EOSv4	Arista DCS-7148SX
EOSv4	Arista DCS-7150S-24-CL
EOSv4	Arista DCS-7150SC-24-CLD
EOSv4	Arista DCS-7150SC-64-CLD
EOSv4	Arista DCS-7160-32CQ
EOSv4	Arista DCS-7160-48TC6
EOSv4	Arista DCS-7160-48YC6
EOSv4	Arista DCS-7170-32C
EOSv4	Arista DCS-7170-32CD
EOSv4	Arista DCS-7170-64C
EOSv4	Arista DCS-7170B-64C
EOSv4	Arista DCS-7260CX-64
EOSv4	Arista DCS-7260CX3-64
EOSv4	Arista DCS-7260CX3-64E
EOSv4	Arista DCS-7260CX3-64LQ
EOSv4	Arista DCS-7260QX-64
EOSv4	Arista DCS-7280CR-48
EOSv4	Arista DCS-7280CR2-60
EOSv4	Arista DCS-7280CR2A-30
EOSv4	Arista DCS-7280CR2A-60
EOSv4	Arista DCS-7280CR2K-30
EOSv4	Arista DCS-7280CR2K-60
EOSv4	Arista DCS-7280CR2M-30
EOSv4	Arista DCS-7280CR3-32D4
EOSv4	Arista DCS-7280CR3-32P4
EOSv4	Arista DCS-7280CR3-36S
EOSv4	Arista DCS-7280CR3-96
EOSv4	Arista DCS-7280CR3A-24D12
EOSv4	Arista DCS-7280CR3A-48D6
EOSv4	Arista DCS-7280CR3A-72
EOSv4	Arista DCS-7280CR3AK-24D12
EOSv4	Arista DCS-7280CR3AK-48D6
EOSv4	Arista DCS-7280CR3AK-72
EOSv4	Arista DCS-7280CR3AM-24D12
EOSv4	Arista DCS-7280CR3AM-48D6
EOSv4	Arista DCS-7280CR3AM-72
EOSv4	Arista DCS-7280CR3E-36S
EOSv4	Arista DCS-7280CR3K-32D4
EOSv4	Arista DCS-7280CR3K-32D4A
EOSv4	Arista DCS-7280CR3K-32P4
EOSv4	Arista DCS-7280CR3K-32P4A
EOSv4	Arista DCS-7280CR3K-36A
EOSv4	Arista DCS-7280CR3K-36S
EOSv4	Arista DCS-7280CR3K-96
EOSv4	Arista DCS-7280CR3MK-32D4
EOSv4	Arista DCS-7280CR3MK-32D4S
EOSv4	Arista DCS-7280CR3MK-32P4
EOSv4	Arista DCS-7280CR3MK-32P4S
EOSv4	Arista DCS-7280DR3-24
EOSv4	Arista DCS-7280DR3A-36
EOSv4	Arista DCS-7280DR3A-54
EOSv4	Arista DCS-7280DR3AK-36

Operating System	Hardware Platform
EOSv4	Arista DCS-7280DR3AK-36S
EOSv4	Arista DCS-7280DR3AK-54
EOSv4	Arista DCS-7280DR3AM-36
EOSv4	Arista DCS-7280DR3AM-54
EOSv4	Arista DCS-7280DR3K-24
EOSv4	Arista DCS-7280PR3-24
EOSv4	Arista DCS-7280PR3K-24
EOSv4	Arista DCS-7280QR-C36
EOSv4	Arista DCS-7280QR-C72
EOSv4	Arista DCS-7280QRA-C36S
EOSv4	Arista DCS-7280SE-64
EOSv4	Arista DCS-7280SE-68
EOSv4	Arista DCS-7280SE-72
EOSv4	Arista DCS-7280SR-48C6
EOSv4	Arista DCS-7280SR2-48YC6
EOSv4	Arista DCS-7280SR2A-48YC6
EOSv4	Arista DCS-7280SR2K-48C6
EOSv4	Arista DCS-7280SR3-40YC6
EOSv4	Arista DCS-7280SR3-48YC8
EOSv4	Arista DCS-7280SR3E-40YC6
EOSv4	Arista DCS-7280SR3E-48YC8
EOSv4	Arista DCS-7280SR3K-48YC8
EOSv4	Arista DCS-7280SR3K-48YC8A
EOSv4	Arista DCS-7280SR3M-48YC8
EOSv4	Arista DCS-7280SR3MK-48YC8A-S
EOSv4	Arista DCS-7280SRA-48C6
EOSv4	Arista DCS-7280SRAM-48C6
EOSv4	Arista DCS-7280SRM-40CX2
EOSv4	Arista DCS-7280TR-48C6
EOSv4	Arista DCS-7280TR3-40C6
EOSv4	Arista DCS-7280TRA-48C6
EOSv4	Arista DCS-7289-SUP
EOSv4	Arista DCS-7289-SUP-S
EOSv4	Arista DCS-7300-SUP2-D
EOSv4	Arista DCS-7500-SUP2
EOSv4	Arista DCS-7500-SUP2-D
EOSv4	Arista DCS-7516-SUP2
EOSv4	Arista DCS-7800-SUP
EOSv4	Arista DCS-7800-SUP1A
EOSv4	Arista DCS-7800-SUP1S
EOSv4	Arista DCS-7800A-SUP1A
EOSv4	Arista DCS-7816-SUP
EOSv4	Arista DCS-7816-SUP1S
EOSv4	Arista SKN-7280CR3-4C2
EOSv4	Arista SKN-7280CR3-4C2G
EOSv4	Arista SKN-7280CR3-4C6
EOSv4	Arista AWE-7250R-16S-FLX
EOSv4	Arista AWE-7230R-4TX-4S-FLX
EOSv4	Arista AWE-5310-2
EOSv4	Arista AWE-5510-2
EOSv4	Arista AWE-7220RP-5TH-2S
EOSv4	Arista AWE-7230R-4TX-4S

Operating System	Hardware Platform
EOSv4	Arista AWE-7250R-16S

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

The module installation procedure for the above platforms is the same as mentioned in Section 11.1, Startup Procedures.

Per the FIPS 140-3 Cryptographic Module Validation Program Management Manual, Section 7.9, Arista affirms that the module remains compliant with the FIPS 140-3 validation when operating on any general-purpose computer (GPC) provided that the GPC uses the specified operating system/mode specified on the validation certificate, or another compatible operating system (including Linux distros such as CentOS 6.x,7.x,8.x). The CMVP allows vendor porting and re-compilation of a validated cryptographic module from the operational environment specified on the validation certificate to an operational environment which was not included as part of the validation testing as long as the porting rules are followed.

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

2.3 Excluded Components

There are no excluded components for the module.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved Mode	Single Approved Mode - Set by calling the FIPS_mode_set(1) function, and call only Approved services.	Approved	Per service indication
Non-Approved Mode	Single Non-Approved Mode - Set by calling the FIPS_mode_set(0) function.	Non-Approved	Per service indication

Table 5: Modes List and Description

When the module starts up successfully, after passing all the pre-operational self-tests, the module is set to use Approved Mode by calling FIPS_mode_set with an argument of 1. Only use Approved services after setting the Approved Mode. When any Non-Approved service is used, the module is in the non-approved mode. Check the syslog for approved service and non-approved service message as specified in Sections 4.3 and 4.4 that provides details on the service indicator implemented by the module.

Mode Change Instructions and Status:

To change to Approved mode, call FIPS_mode_set(1) and verify the return value is equal to “1”, indicating the function executed successfully. To validate that Approved mode is active, call FIPS_mode() and verify the return value is equal to “1”.

To change to Non-Approved mode, call FIPS_mode_set(0) and verify the return value is equal to “1”. To validate that Non-Approved mode is active, call FIPS_mode() and verify the return value is equal to “0”. Check the syslog for non-approved service message as specified in Section 4.4 which provides details on the non-approved service indicator implemented by the module.

2.5 Algorithms

Approved Algorithms:

The table below lists the approved security functions (or cryptographic algorithms) of the module, including specific key lengths employed for approved services, and implemented modes or methods of operation of the algorithms.

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A4984	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A4984	Key Length - 128, 192, 256	SP 800-38C
AES-CFB1	A4984	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB128	A4984	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A4984	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A4984	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B
AES-CTR	A4984	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A4984	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A4984	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-XTS Testing Revision 2.0	A4984	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E
Counter DRBG	A4984	Prediction Resistance - No, Yes Mode - AES-128, AES-192, AES-256 Derivation Function Enabled - No, Yes	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-4)	A4984	Curve - P-256, P-384, P-521	FIPS 186-4
ECDSA KeyVer (FIPS186-4)	A4984	Curve - P-256, P-384, P-521	FIPS 186-4
ECDSA SigGen (FIPS186-4)	A4984	Component - No Curve - P-256, P-384, P-521	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A4984	Component - No Curve - P-256, P-384, P-521	FIPS 186-4
Hash DRBG	A4984	Prediction Resistance - No, Yes Mode - SHA-1, SHA2-256, SHA2-512	SP 800-90A Rev. 1
HMAC DRBG	A4984	Prediction Resistance - No, Yes Mode - SHA-1, SHA2-256, SHA2-512	SP 800-90A Rev. 1
HMAC-SHA-1	A4984	Key Length - Key Length: 112-2048 Increment 8	FIPS 198-1
HMAC-SHA2-224	A4984	Key Length - Key Length: 112-2048 Increment 8	FIPS 198-1
HMAC-SHA2-256	A4984	Key Length - Key Length: 112-2048 Increment 8	FIPS 198-1
HMAC-SHA2-384	A4984	Key Length - Key Length: 112-2048 Increment 8	FIPS 198-1
HMAC-SHA2-512	A4984	Key Length - Key Length: 112-2048 Increment 8	FIPS 198-1

Algorithm	CAVP Cert	Properties	Reference
KAS-ECC-SSC Sp800-56Ar3	A4984	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	A4984	Domain Parameter Generation Methods - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Scheme - dhEphem - KAS Role - initiator, responder	SP 800-56A Rev. 3
KDF IKEv1 (CVL)	A4984	Authentication Method - Pre-shared Key Preshared Key Length - Preshared Key Length: 8-8192 Increment 8 Diffie-Hellman Shared Secret Length - Diffie-Hellman Shared Secret Length: 1024-8192 Increment 1024	SP 800-135 Rev. 1
KDF IKEv2 (CVL)	A4984	Diffie-Hellman Shared Secret Length - Diffie-Hellman Shared Secret Length: 1024-8192 Increment 1024 Derived Keying Material Length - Derived Keying Material Length: 256-2048 Increment 128	SP 800-135 Rev. 1
KDF SP800-108	A4984	KDF Mode - Counter	SP 800-108 Rev. 1
KDF SSH (CVL)	A4984	Cipher - AES-128, AES-192, AES-256	SP 800-135 Rev. 1
KDF TLS (CVL)	A4984	TLS Version - v1.0/1.1 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
KTS-IFC	A4984	Modulo - 2048, 3072, 4096 Key Generation Methods - rsakpg2-basic Scheme - KTS-OAEP-basic - KAS Role - initiator, responder Key Length - 512	SP 800-56B Rev. 2
RSA KeyGen (FIPS186-4)	A4984	Key Generation Mode - B.3.3, B.3.6 Modulo - 2048, 3072, 4096 Primality Tests - Table C.2 Private Key Format - Standard	FIPS 186-4
RSA SigGen (FIPS186-4)	A4984	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-4)	A4984	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096	FIPS 186-4
SHA-1	A4984	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-224	A4984	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A4984	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A4984	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A4984	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
TLS v1.2 KDF RFC7627 (CVL)	A4984	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1

Table 6: Approved Algorithms

Vendor-Affirmed Algorithms:

The table below lists the vendor-affirmed algorithms that are allowed in the approved mode of operation.

Name	Properties	Implementation	Reference
CKG Section 4	Key Type:Symmetric and Asymmetric	N/A	SP 800-133r2 Section 4, example 1

Table 7: Vendor-Affirmed Algorithms

The DRBG output is approved for generating keys or SSPs. The calling application must following the DRBG specification in Section 2.7 for entropy and DRBG seeding for the target security strength.

Non-Approved, Allowed Algorithms:

N/A for this module.

The module does not implement any Non-Approved Algorithms Allowed in the Approved Mode of Operation. (SP 800-140B table 7: *Non-Approved Algorithms Allowed in the Approved Mode of Operation* has been omitted).

Non-Approved, Allowed Algorithms with No Security Claimed:

The table below 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 the Approved Services table in Section 4.3.

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

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

Non-Approved, Not Allowed Algorithms:

The table below lists non-approved algorithms that are not allowed in the approved mode of operation.

Name	Use and Function
DSA (disallowed)	KeyGen, PQGGen, PQGVer, SigGen, and SigVer
RSA (disallowed)	Key Encryption and Decryption using PKCS#1 v1.5
Counter DRBG (non-compliant)	Random number generation using Counter DRBG without a DF [algorithm disabled by module in approved mode]
Triple-DES KW (non-compliant)	Key wrapping [algorithm disabled by module in Approved mode]
Blowfish	Encryption and Decryption [algorithm disabled by module in approved mode]
Camellia 128/192/256	Encryption and Decryption [algorithm disabled by module in approved mode]
CAST5	Encryption and Decryption [algorithm disabled by module in approved mode]
DES	Encryption and Decryption [algorithm disabled by module in approved mode]
DES-X	Encryption and Decryption [algorithm disabled by module in approved mode]
IDEA	Encryption and Decryption [algorithm disabled by module in approved mode]
RC2	Encryption and Decryption [algorithm disabled by module in approved mode]
RC5	Encryption and Decryption [algorithm disabled by module in approved mode]
SEED	Encryption and Decryption [algorithm disabled by module in approved mode]
Triple-DES	Encryption and Decryption [algorithm disabled by module in approved mode]
MD4	Message Digest [algorithm disabled by module in approved mode]
MD5	Message Digest [algorithm disabled by module in approved mode]
RIPEMD-160	Message Digest [algorithm disabled by module in approved mode]
Whirlpool	Message Digest [algorithm disabled by module in approved mode]
Triple-DES MAC	Message Digest [algorithm disabled by module in approved mode]
HMAC-MD5	Keyed Hash [algorithm disabled by module in approved mode]

Table 9: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
KAS-ECC-SSC	KAS-SSC	SP 800-56Arev3. KAS_ECC_SSC per IG D.F Scenario 2, path (1)	Caveat:Key establishment methodology provides between 128 and 256 bits of encryption strength	KAS-ECC-SSC Sp800-56Ar3: (A4984)
KAS-FFC-SSC	KAS-SSC	SP 800-56Arev3. KAS_FFC_SSC per IG D.F Scenario 2, path (1)	Caveat:Key establishment methodology provides between 112 and 200 bits of encryption strength	KAS-FFC-SSC Sp800-56Ar3: (A4984)
KTS-IFC	KTS-Decap KTS-Encap	SP 800-56Brev2. KTS-IFC key encapsulation and un-encapsulation per IG D.G.	Caveat:Key transport provides between 112 and 152 bits of encryption strength.	KTS-IFC: (A4984)
Asymmetric Key Pair Generation	AsymKeyPair-KeyGen CKG	Asymmetric Key Pair Generation performed by ECDSA or RSA		ECDSA KeyGen (FIPS186-4): (A4984) RSA KeyGen (FIPS186-4): (A4984) Counter DRBG: (A4984) Hash DRBG: (A4984) HMAC DRBG: (A4984) CKG Section 4: () Key Type: Symmetric and Asymmetric
Asymmetric Key Verification	AsymKeyPair-KeyVer	Asymmetric Key Pair Verification for ECDSA		ECDSA KeyVer (FIPS186-4): (A4984)
Digital Signature Generation	DigSig-SigGen	Digital Signature Generation using ECDSA or RSA		ECDSA SigGen (FIPS186-4): (A4984) RSA SigGen (FIPS186-4): (A4984) SHA2-224: (A4984) SHA2-256: (A4984) SHA2-384: (A4984) SHA2-512: (A4984)
Digital Signature Verification	DigSig-SigVer	Digital Signature Verification using ECDSA or RSA		ECDSA SigVer (FIPS186-4): (A4984) RSA SigVer (FIPS186-4):

Name	Type	Description	Properties	Algorithms
				(A4984) SHA-1: (A4984) SHA2-224: (A4984) SHA2-256: (A4984) SHA2-384: (A4984) SHA2-512: (A4984)
Encryption	BC-UnAuthEncrypt	Block Cipher Symmetric Encryption Non-Authenticated		AES-CBC: (A4984) AES-CFB1: (A4984) AES-CFB128: (A4984) AES-CFB8: (A4984) AES-CTR: (A4984) AES-ECB: (A4984) AES-XTS Testing Revision 2.0: (A4984)
Authenticated Encryption	BC-AuthEncrypt	Block Cipher Symmetric Encryption Authenticated		AES-CCM: (A4984) AES-GCM: (A4984)
MAC1	MAC	Message Authentication Computation with HMAC-SHA-1		HMAC-SHA-1: (A4984) Key Length: 112-2048
MAC2	MAC	Message Authentication Computation with AES CMAC and HMAC		AES-CMAC: (A4984) Key Length: 128, 192, 256 HMAC-SHA-1: (A4984) Key Length: 112-2048 HMAC-SHA2-224: (A4984) Key Length: 112-2048 HMAC-SHA2-256: (A4984) Key Length: 112-2048 HMAC-SHA2-384: (A4984) Key Length: 112-2048 HMAC-SHA2-512: (A4984) Key Length: 112-2048
Message Digest	SHA	Message Digest		SHA-1: (A4984) SHA2-224: (A4984) SHA2-256: (A4984) SHA2-384: (A4984) SHA2-512: (A4984)

Name	Type	Description	Properties	Algorithms
DRBG	DRBG	Generation of random numbers		Counter DRBG: (A4984) Hash DRBG: (A4984) HMAC DRBG: (A4984)
KDF-TLS	KAS-135KDF	Key derivation for TLS		KDF TLS: (A4984) TLS v1.2 KDF RFC7627: (A4984)
KDF-SSH	KAS-135KDF	Key derivation for SSH		KDF SSH: (A4984)
KDF-IKE	KAS-135KDF	Key derivation for IKE		KDF IKEv1: (A4984) KDF IKEv2: (A4984)
Authenticated Decryption	BC-AuthDecrypt	Block Cipher Symmetric Decryption Authenticated		AES-CCM: (A4984) AES-GCM: (A4984)
Decryption	BC-UnAuthDecrypt	Block Cipher Symmetric Decryption		AES-CBC: (A4984) AES-CFB1: (A4984) AES-CFB128: (A4984) AES-CFB8: (A4984) AES-CTR: (A4984) AES-ECB: (A4984) AES-XTS Testing Revision 2.0: (A4984)
MAC3	MAC	MAC computation with HMAC-SHA2-256 used for the Integrity test		HMAC-SHA2-256: (A4984)
KDF-SP800-108	KBKDF	Key Derivation with SP 800-108r1		KDF SP800-108: (A4984)

Table 10: Security Function Implementations

2.7 Algorithm Specific Information

Industry Protocols

The Module does not implement the TLS, SSH, and IPSec protocols itself. However, the module provides the cryptographic functions required for implementing the protocols. The calling application is allowed to construct or use IV for these protocols.

Note: no parts of the TLS v1.0/1.1, v1.2, SSHv2, or IPsec-v3 protocols, other than the approved cryptographic algorithms and the KDFs, have been tested by the CAVP and CMVP.

AES-GCM IV Generation

The GCM IV generation for these implementations complies respectively with IG C.H under Scenario 1 and Scenario 2. The GCM shall only be used in the context of the AES-GCM encryption executing under each scenario, and using the referenced APIs explained next.

Scenario 1, TLS 1.2

The module provides the cryptographic functions to support the AES-GCM ciphersuites from Section 3.3.1 of SP800-52rev2 and the mechanism for IV generation per RFC 5288.

The module explicitly ensures that the counter (the nonce_explicit part of the IV) does not exhaust the maximum number of possible values of 2^{64}-1 for a given session key. If this exhaustion condition is observed, the module returns an error indication to the calling application, which will then need to either abort the connection, or trigger a handshake to establish a new encryption key.

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

Scenario 1, SSHv2

The module provides the cryptographic functions to support the calling application for compliant with RFCs 4252, 4253, and 5647.

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

Scenario 1, IPsec-v3

The module provides the cryptographic functions to support the calling application for compliant with RFCs 4106 and 5282.

The module's implementation of AES-GCM is used together with an application that runs outside the module's cryptographic boundary. This application negotiates the protocol session's keys and the value in the first 32 bits of the nonce. The construction of the last 64 bits of the nonce is deterministic and uses a counter.

The module explicitly ensures that the counter (the nonce_explicit part of the IV) does not exhaust the maximum number of possible values of 2^{64}-1 for a given session key. If this exhaustion condition is observed, the module returns an error indication to the calling application, which will then need to either abort the connection, or trigger a handshake to establish a new encryption key.

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

Scenario 2, Random IV

In this implementation, the module offers the interface RAND_bytes for compliance with Scenario 2 of IG C.H and SP800-38D Section 8.2.2. The AES-GCM IV is generated randomly internal to the module using the module's approved DRBG. The DRBG seeds itself from the entropy source. The GCM IV is 96 bits in length. The selection of the IV construction method is the responsibility of the calling application. In approved mode, only internally generated IVs within the TOEPP is considered compliant for use.

AES-XTS

AES-XTS shall only be used for storage applications. Per IG C.I, the module explicitly checks that Key_1 ≠ Key_2.

DRBG

The module relies on passively provided entropy. The default CTR_DRBG is used with a derivation function. The developer integrating the module may use the Hash DRBG or HMAC DRBG for random number generation. The developer must ensure that the entropy used to seed the DRBGs comes from a source located within the TOEPP.

HMAC

The calling application shall ensure that HMAC keys be generated as specified in SP 800-133 and an HMAC key shall have a security strength that meets or exceeds the security strength required to protect the data over which the HMAC is computed, that HMAC keys shall be kept secret, that when truncating the HMAC output to generate a MacTag to a desired length, λ, the λ left-most bits of the HMAC output shall be used as the MacTag and the length of λ shall be no less than 32 bits, and that if the probability of a forgery for a given MagTag length and the number of failed MacTag verifications is not acceptable for the system, the HMAC key shall be changed to a new value before the number of failed MAC verifications allowed, per IG C.L,

KAS

The module does not establish SSPs using an approved key agreement scheme (KAS). However, it does offer some or all of the underlying KAS cryptographic functionality to be used by an external operator/application as part of an approved KAS.

KTS

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

SHA-1

SHA-1 for digital signature verification is legacy use. SHA-1 for digital signature generation is non-approved. The calling application shall ensure that SHA-1 is not used in digital signature generation as SHA-1 is disallowed per SP 800-131Ar2.

Algorithms designated as “Legacy” can only be used on data that was generated prior to the Legacy Date specified in FIPS 140-3 IG C.M.

2.8 RBG and Entropy

N/A for this module.

The module does not implement or actively call any SP 800-90B entropy sources. (SP 800-140B table 10: *Entropy Certificates* has been omitted)

The module provides an SP800-90Arev1-compliant Deterministic Random Bit Generator (DRBG) using CTR_DRBG mechanism with AES-256 for creation of key components of asymmetric keys, and random number generation. Operators may instantiate and use the other Approved DRBGs offered by the module. The module receives entropy passively and uses 384 bits of entropy to seed the DRBG.

2.9 Key Generation

For generating RSA, ECDSA and EC Diffie-Hellman keys, the module implements asymmetric key generation services compliant with FIPS186-4 and using a DRBG compliant with SP800-90Arev1.

The random value used in asymmetric key generation is obtained from the DRBG. In accordance with FIPS 140-3 IG D.H, the cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys as per section 5.1 of SP800-133rev2 (vendor affirmed) by obtaining a random bit string directly from an approved DRBG and that can support the required security strength requested by the caller (without any V, as described in Additional Comments 2 of IG D.H).

The module does not provide a dedicated service for generating symmetric keys. However, symmetric keys can be derived using SP800-135rev1 for TLS KDF, IKE v1/2 KDF, and SSHv2 KDF algorithms, as well as SP800-108 counter KBKDF. This generation method maps to section 6.2 of SP800-133rev2.

2.10 Key Establishment

The module provides EC Diffie-Hellman and FFC Diffie-Hellman shared secret computation compliant with SP800-56Arev3, in accordance with scenario 2, path (1) of IG D.F. It also provides RSA OAEP key transport as KTS-IFC compliant with SP 800-56Br2 in accordance with IG D.G and applications may transport keys as TLS, SSHv2, or IPSec protocol payload compliant to SP 800-38F in accordance with IG D.G.

Additionally, the module also supports key derivation using TLS 1.0/1.1, TLS 1.2, IKE v1, IKE v2, SSHv2 KDF compliant to SP800-135rev1 and counter KBKDF compliant to SP800-108.

The module provides the cryptographic building blocks for symmetric AES key wrapping and asymmetric key encapsulation. A calling application may use these to implement a protocol that uses a compliant KTS for its key establishment.

2.11 Industry Protocols

The module does not implement any industry protocols. However, it provides the building blocks to support the following protocols.

Protocol	Reference
SSHv2	[IG D.F and SP 800-135]
TLS v1.0/v1.1/v1.2	[IG D.F, IG D.G and SP 800-135]
IPsec-v3	[RFC 4106, 5282, 7296]

Table A- Security Relevant Protocols Used in Approved Mode

	Key Exchange	Server/ Host Auth	Cipher	Integrity
DTLS [IG D.G]	See TLS entry in this table.			
SSHv2 [IG D.F and SP 800-135]	ECDH-SHA2-NISTP521, ECDH-SHA2-NISTP384, ECDH-SHA2-NISTP256, DIFFIE-HELLMAN GROUP14-SHA-1, DIFFIE-HELLMAN GROUP14-SHA256, DIFFIE-HELLMAN GROUP16-SHA512	ECDSA P-521, ECDSA P-384, ECDSA P-256, RSA	AES-GCM-128 AES-GCM-256 AES-CBC-128 AES-CBC-192 AES-CBC-256 AES-CTR-128 AES-CTR-192 AES-CTR-256	HMAC-SHA-1 HMAC-SHA2-256 HMAC-SHA2-512 AES-GCM-128 AES-GCM-256
TLS [IG D.G and SP 800-135]	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 for TLS v1.0, v1.1, v1.2			
	ECDHE	RSA	AES-GCM-128	AES-GCM-128
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 for TLS v1.0, v1.1, v1.2			
	ECDHE	RSA	AES-GCM-256	AES-GCM-256
	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 for TLS v1.0, v1.1, v1.2			
	ECDHE	ECDSA	AES-GCM-128	AES-GCM-128
	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 for TLS v1.0, v1.1, v1.			

	ECDHE	ECDSA	AES-GCM-256	AES-GCM-256
	TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8 for TLS v1.0, v1.1, v1.2			
	ECDHE	ECDSA	AES-CCM-256	AES-CCM-256
	TLS_ECDHE_ECDSA_WITH_AES_256_CCM for TLS v1.0, v1.1, v1.2			
	ECDHE	ECDSA	AES-CCM-256	AES-CCM-256
	TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 for TLS v1.0, v1.1, v1.2			
	ECDHE	ECDSA	AES-CCM-128	AES-CCM-128
	TLS_ECDHE_ECDSA_WITH_AES_128_CCM for TLS v1.0, v1.1, v1.2			
	ECDHE	ECDSA	AES-CCM-128	AES-CCM-128
	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 for TLS v1.0, v1.1, v1.2			
	ECDHE	ECDSA	AES-CBC-256	HMAC-SHA2-384
	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 for TLS v1.0, v1.1, v1.2			
	ECDHE	ECDSA	AES-CBC-128	HMAC-SHA2-256
	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA for TLS v1.0, v1.1, v1.2			
	ECDHE	ECDSA	AES-CBC-256	HMAC-SHA-1
	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA for TLS v1.0, v1.1, v1.2			
	ECDHE	ECDSA	AES-CBC-128	HMAC-SHA-1
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 for TLS v1.0, v1.1, v1.2			
	ECDHE	RSA	AES-CBC-128	HMAC-SHA2-256
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA256 for TLS v1.0, v1.1, v1.2			
	ECDHE	RSA	AES-CBC-256	HMAC-SHA2-256
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA for TLS v1.0, v1.1, v1.2			
	ECDHE	RSA	AES-CBC-256	HMAC-SHA-1
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA for TLS v1.0, v1.1, v1.2			
	ECDHE	RSA	AES-CBC-128	HMAC-SHA-1
	TLS_DHE_RSA_WITH_AES_256_CCM_8 for TLS v1.0, v1.1, v1.2			
	DHE	RSA	AES-CCM-256	AES-CCM-256
	TLS_DHE_RSA_WITH_AES_256_CCM for TLS v1.0, v1.1, v1.2			
	DHE	RSA	AES-CCM-256	AES-CCM-256
	TLS_DHE_RSA_WITH_AES_128_CCM_8 for TLS v1.0, v1.1, v1.2			

	DHE	RSA	AES-CCM-128	AES-CCM-128
	TLS_DHE_RSA_WITH_AES_128_CCM for TLS v1.0, v1.1, v1.2			
	DHE	RSA	AES-CCM-128	AES-CCM-128
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 for TLS v1.0, v1.1, v1.2			
	DHE	RSA	AES-CBC-256	HMAC-SHA2-256
	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 for TLS v1.0, v1.1, v1.2			
	DHE	RSA	AES-CBC-128	HMAC-SHA2-256
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA for TLS v1.0, v1.1, v1.2			
	DHE	RSA	AES-CBC-256	HMAC SHA-1
	TLS_DHE_RSA_WITH_AES_128_CBC_SHA for TLS v1.0, v1.1, v1.2			
	DHE	RSA	AES-CBC-128	HMAC SHA-1
IPsec-v3	diffie-hellman MODP-2048, MODP-3072, MODP-4096 , MODP-6144, MODP-8192 ec diffie-hellman secp256r1, secp384r1, secp521r1		AES-GCM-128 AES-GCM-192 AES-GCM-256 AES-CBC-128 AES-CBC-192 AES-CBC-256 AES-CTR-128 AES-CTR-192 AES-CTR-256 AES-CCM-128 AES-CCM-192 AES-CCM-256	AES-GCM-128 AES-GCM-192 AES-GCM-256 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512 AES-CCM-128 AES-CCM-192 AES-CCM-256

Table B - Security Relevant Protocols Used in Approved Mode

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

As a Software module, the module interfaces are defined as Software or Firmware Module Interfaces (SFMI), and there are no physical ports. The interfaces are mapped to the API provided by the module, through which the operator can interact. The interfaces are listed in the table below.

All data output via data output interface is inhibited under the following circumstances:

- When the module is in POST mode
- During zeroisation of data such as CSPs
- When the module enters error state.

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	API input parameters for data
N/A	Data Output	API output parameters for data

Physical Port	Logical Interface(s)	Data That Passes
N/A	Control Input	API function calls
N/A	Status Output	API return codes, error messages, logging messages

Table 11: Ports and Interfaces

The module does not support Control Output.

4 Roles, Services, and Authentication

4.1 Authentication Methods

Table 13 lists all operator roles supported by the module (for the role, CO indicates “Crypto Officer”) and the security strength of the authentication. The Module does not support a maintenance role nor bypass capability. The Module does not support concurrent operators.

Method Name	Description	Security Mechanism	Strength Each Attempt	Strength per Minute
Password	Password authentication mechanism	HMAC-SHA-1 (A4984)	95^16 (module enforces 16 character minimum password length)	Chance of guessing in one minute 1 in 9.03*10^18

Table 12: Authentication Methods

The module supports Role-based authentication using passwords as the SP 800-140E memorized secret. The module has a strength of authentication objective of at least 1/95^8, and to achieve that over a one-minute period the module enforces a minimum password length of 16 characters. The password must be set by the calling application through the “FIPS_set_password” API during module initialization.

Since the module enforces a minimum 16-character password length and there are 95 possible ASCII characters (upper and lower case, digits, special characters), it has an authentication strength of 95^{16}. Thus, the false acceptance rate is $1/95^{16}$. Assuming a very high-performing CPU that runs at 4 GHz with 24 cores which means it can perform 4 billion * 24 instructions per second, the probability of a successful random access within a minute is still extremely unlikely at $1/95^{16}$ * 4 billion * 24 cores * 60 seconds/min. It would take about 150 billion years to have a 1% chance of cracking the password in this scenario:

$1/95^{16}$ * 4 billion * 24 cores * 60 sec / min * 60 min / hr * 24 hr / day * 365 days / year * 150 billion = 0.0103

4.2 Roles

The module supports the Crypto Officer role only, whose authentication is performed by the module using passwords. This sole role is implicitly assumed by the operator of the module when performing a service after authentication.

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	Password

Table 13: Roles

4.3 Approved Services

The module provides services to operators who assume the available role. All services are described in detail in the developer documentation. For the role, CO indicates “Crypto Officer”. The following table lists the approved services that utilize approved and allowed security functions.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Authenticated Decryption	Authenticated Decryption	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	Ciphertext, Authentication Tag, Key, IV	Plaintext	Authenticated Decryption	Crypto Officer - AES Keys: W,E
Authenticated Encryption	Authenticated Encryption	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	Plaintext, key, IV	Ciphertext, authentication tag	Authenticated Encryption	Crypto Officer - AES Keys: W,E
Decryption	Decryption	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	Ciphertext, key	Plaintext	Decryption	Crypto Officer - AES Keys: W,E
Encryption	Encryption	Mode indicator 1, return code 1, syslog message	Plaintext, Key	Ciphertext	Encryption	Crypto Officer - AES Keys: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		indicating the service "starting", "performed" or "initialized"				
Key Derivation (TLS)	Deriving TLS keys	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	PRF algorithm, TLS master secret	Derived Keys	KDF-TLS	Crypto Officer - TLS Derived key/AES & HMAC: G,R - TLS master secret: G,E - TLS pre-master secret: W,E
Key Derivation (SSH)	Deriving SSH keys	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	PRF algorithm, SSH shared secret	Derived Keys	KDF-SSH	Crypto Officer - SSH Shared Secret: W,E - SSH Derived key/AES & HMAC: G,R
Key Derivation (IKE)	Deriving IKE keys	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	PRF algorithm, IKE shared secret	Derived Keys	KDF-IKE	Crypto Officer - IKE shared secret: W,E - IKE Derived key/AES & HMAC: G,R

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Key Derivation (SP 800-108r1)	Deriving keys	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	Shared Secret, key size	Derived Keys	KDF-SP800-108	Crypto Officer - Key Derivation Key: W,E - 800-108 Derived Key: G,R
Key Encapsulation	Key Encapsulation per SP 800-56Br2	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	RSA keypair, keying material to encapsulate	Encapsulated key	KTS-IFC	Crypto Officer - RSA Key Pair: W,E - Keying Material: R,W
Key Generation	Generating Key pair	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	Algorithm, key size	Key Pair	Asymmetric Key Pair Generation	Crypto Officer - ECDSA key pair: G,R - RSA Key Pair: G,R - DRBG Seed: W,E - DRBG C Value: W,E - DRBG V Value: W,E - DRBG Key Value: W,E
Key Un-encapsulation	Key Un-encapsulation per SP 800-56Br2	Mode indicator 1, return code 1, syslog	RSA keypair, keying material to	Un-encapsulated key	KTS-IFC	Crypto Officer - RSA Key Pair: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		message indicating the service "starting", "performed" or "initialized"	un-encapsulate			- Keying Material: R,W
Key Verification	Verifying the public key	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	Key to verify	Return codes and log messages	Asymmetric Key Verification	Crypto Officer - ECDSA key pair: W,E
Initialize	Initialize FIPS password using FIPS_set_password	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	Crypto Officer Password	None	MAC1	Crypto Officer - Crypto Officer Password: W,E - Hashed Password: E
Message Authentication Generation	MAC computation	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	Message, Algorithm, key	Message Authentication code	MAC2	Crypto Officer - AES Keys: W,E - HMAC key: W,E
Message Digest	Generating message digest	Mode indicator 1, return	Message	Digest of the message	Message Digest	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		code 1, syslog message indicating the service "starting", "performed" or "initialized"				
On-Demand Integrity Test	Initiate integrity test on-demand through FIPS_check_incore_fingerprint	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	None	Result of test (pass/fail)	MAC3	Crypto Officer
On-Demand Self-Test	Initiate pre-operational and conditional CAST self-tests through FIPS_selftest	Mode indicator 1, return code 1, syslog message indicating the self-tests were executed	None	Result of self-test (pass/fail)	KAS-ECC-SSC KAS-FFC-SSC KTS-IFC Digital Signature Generation Digital Signature Verification Encryption Authenticated Encryption MAC1 MAC2 Message Digest DRBG KDF-TLS KDF-SSH KDF-IKE Authenticated Decryption Decryption MAC3	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					KDF-SP800-108	
Random Number Generation	Generating random numbers	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	API call parameters	Return code, Random Bits	DRBG	Crypto Officer - DRBG Entropy Input: W,E - DRBG Seed: G,E - DRBG C Value: G,E - DRBG V Value: G,E - DRBG Key Value: G,E
Shared Secret Computation	Calculating Shared Secret	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	EC Curve or DH parameters, V's public key	Shared Secret	KAS-ECC-SSC KAS-FFC-SSC DRBG	Crypto Officer - DRBG Seed: W,E - DRBG C Value: W,E - DRBG V Value: W,E - DRBG Key Value: W,E - DH Private Key: G,E,Z - DH Public Key: G,E,Z - ECDH Private Key: G,E,Z - ECDH Public Key: G,E,Z
Show Status	Show status of the module state using FIPS_mode	None	None	Return code of 1	None	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
				indicates Approved mode enabled, 0 is disabled		
Show Version	Show the version of the module using FIPS_module_version_text	None	None	String indicating the module version and name	None	Crypto Officer
Signature Generation	Generating signature	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	Message, hash algorithm, private key	Signature	Digital Signature Generation	Crypto Officer - ECDSA key pair: W,E - RSA Key Pair: W,E
Signature Verification	Verifying signature	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	Message, Signature, hash algorithm, public key	Verification result	Digital Signature Verification	Crypto Officer - ECDSA key pair: W,E - RSA Key Pair: W,E
Zeroise	Zeroise SSP in volatile memory	None	Context containing SSPs	None	None	Crypto Officer - 800-108 Derived Key: Z - AES Keys: Z - Crypto Officer Password: Z - Hashed Password: Z - DRBG

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Entropy Input: Z - DRBG Seed: Z - DRBG C Value: Z - DRBG V Value: Z - DRBG Key Value: Z - ECDSA key pair: Z - HMAC key: Z - IKE shared secret: Z - IKE Derived key/AES & HMAC: Z - Keying Material: Z - Shared Secret: Z - SSH Shared Secret: Z - SSH Derived key/AES & HMAC: Z - TLS Derived key/AES & HMAC: Z - TLS master secret: Z - TLS pre-master secret: Z - RSA Key Pair: Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- DH Private Key: Z - DH Public Key: Z - ECDH Private Key: Z - ECDH Public Key: Z - Key Derivation Key: Z

Table 14: Approved Services

Service Indicator

The module implements a status indicator that indicates whether the invoked service is approved. When approved mode is active, all algorithms except DSA (disallowed) and RSA (disallowed) in the "Non-Approved, Not Allowed Algorithms" table are disabled. If a disabled algorithm is used in approved mode, an error code of 0 indicating failure is returned and the reason for failure is added to the error queue.

To verify if approved mode is active, the function FIPS_mode() should be called. This function is described in Section "2.4 Modes of Operation".

In addition to the return code, the module outputs syslog messages to indicate whether an invoked service is approved. The usage is as follows:

STEP 1: Check the system log output buffer for existing log messages.

STEP 2: Make a service call i.e., API function for performing a service.

STEP 3: Check the system log output buffer for a new log message indicating which service was invoked. For example, running the TLS key derivation service will generate a new log message saying "OpenSSL: Key derivation service for TLS performed", which indicates the invoked function was an approved service. In contrast, running the key wrapping service using RSA (disallowed) will generate a new log message saying "OpenSSL: RSA unapproved service - encryption with transitioned padding", which indicates that the invoked function was an non-approved service.

4.4 Non-Approved Services

The following table lists the non-approved services that utilize non-approved security functions.

Name	Description	Algorithms	Role
Decryption	Decryption - A return code 0 indicates the algorithm is disabled in Approved mode. A return code 1 indicates the algorithm is enabled in non-Approved mode.	Blowfish Camellia 128/192/256 CAST5 DES DES-X IDEA	CO

Name	Description	Algorithms	Role
		RC2 RC5 SEED Triple-DES	
Encryption	Encryption - A return code 0 indicates the algorithm is disabled in Approved mode. A return code 1 indicates the algorithm is enabled in non-Approved mode.	Blowfish Camellia 128/192/256 CAST5 DES DES-X IDEA RC2 RC5 SEED Triple-DES	CO
Key Wrapping	Encrypting/Decrypting key - A return code 0 indicates the Triple-DES KW algorithm is disabled in Approved mode. A return code 1 indicates the algorithm is enabled in non-Approved mode. A return code 1 along with a syslog message indicates the RSA PKCS1 for key wrapping/unwrapping is non-approved.	RSA (disallowed) Triple-DES KW (non-compliant)	CO
Message Digest	Hash computation - A return code 0 indicates the algorithm is disabled in Approved mode. A return code 1 indicates the algorithm is enabled in non-Approved mode.	MD4 MD5 RIPEMD-160 Whirlpool Triple-DES MAC HMAC-MD5	CO
DSA	FIPS 186-4 DSA KeyGen, PQGGen, PQGVer, SigGen, and SigVer - A return code 1 along with a syslog message indicates the algorithm is non-approved.	DSA (disallowed)	CO
Random Number Generation	Random number generation using Counter DRBG without a DF - A return code 0 indicates the algorithm is disabled in Approved mode. A return code 1 indicates the algorithm is enabled in non-Approved mode.	Counter DRBG (non-compliant)	CO

Table 15: Non-Approved Services

4.5 External Software/Firmware Loaded

The module does not have external software/firmware load capability.

5 Software/Firmware Security

5.1 Integrity Techniques

The integrity of the module is validated by comparing the module with a HMAC-SHA2-256 value generated after the build of fipscanister.o, which is the FIPS Object Module. This generated value is embedded into fipscanister.o before fipscanister.o is statically linked to libcrypto.so. During runtime the FIPS_mode_set() function calculates the digest over fipscanister.o, excluding the embedded hash value, and checks to see if the embedded value matches the calculated digest.

5.2 Initiate on Demand

The module provides on-demand integrity test. The integrity test is performed by the On-Demand Integrity Test service, which calls the FIPS_check_incore_fingerprint function. The integrity test is also performed as part of the Pre-Operational Self-Tests. One can also initiate the On Demand Integrity Test service by calling “openssl --fips” on the command line, which is a calling application that runs the module’s self-test API function. A successful test will show “FIPS mode is enabled”.

5.3 Open-Source Parameters

The source distribution package (including Arista own patches and updates) is located at Arista internal repository. The module is built with the following configuration for linux-x86_64 using Linux 5.10 and gcc 11.3:

OPENSSL_NO_BF (skip dir)
OPENSSL_NO_CAMELLIA (skip dir)
OPENSSL_NO_CAST (skip dir)
OPENSSL_NO_EC_NISTP_64_GCC_128 (skip dir)
OPENSSL_NO_GMP (skip dir)
OPENSSL_NO_IDEA (skip dir)
OPENSSL_NO_JPAKE (skip dir)
OPENSSL_NO_KRB5
OPENSSL_NO_MD2 (skip dir)
OPENSSL_NO_MD5 (skip dir)
OPENSSL_NO_MDC2 (skip dir)
OPENSSL_NO_RC2 (skip dir)
OPENSSL_NO_RC4 (skip dir)
OPENSSL_NO_RC5 (skip dir)
OPENSSL_NO_RFC3779 (skip dir)
OPENSSL_NO_RIPEMD (skip dir)
OPENSSL_NO_SEED (skip dir)
OPENSSL_NO_SRP (skip dir)
OPENSSL_NO_SSL2 (skip dir)
OPENSSL_NO_SSL3 (skip dir)
OPENSSL_NO_STORE (skip dir)
OPENSSL_NO_TLS1 (skip dir)
OPENSSL_NO_TLSEXT (skip dir)

IsMK1MF=0
CC =gcc
CFLAG =-DOPENSSL_FIPSCANISTER -fPIC -DOPENSSL_PIC -DOPENSSL_THREADS -D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H -g -Wa,--noexecstack -m64 -DL_ENDIAN -DTERMIO -O3 -Wall -DOPENSSL_IA32_SSE2 -DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_MONT5 -DOPENSSL_BN_ASM_GF2m -DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -DMD5_ASM -DAES_ASM -DWHIRLPOOL_ASM -DGHASH_ASM
EX_LIBS =-lm -ldl

To build the openssl-fips object module
./ a4 rpmbuild openssl-fips

This command executes the following steps:

1. Patch the OpenSSL object module code with Arista patches
2. Run “./config”
3. Run “make”
4. This creates the object module at fipscanister.o and hash at fipscanister.o.sha1, and other files such as fips_premain.o and fips_premain.o.sha1.
5. These files are placed in /usr/local/ssl/fips-2.0/lib folder, which OpenSSL is configured to use.

Run “a4 rpmbuild openssl”, which will build openssl using the FIPS object module and generate OpenSSL RPMs, which are installed on the software image during the build process.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

6.2 Configuration Settings and Restrictions

The module shall be built as stated in Section 5.3. All switch software in the operational environment shall be configured to limit access to the built-in 'network-admin' role as stated in Section 11.2 to protect the module against unauthorized execution, unauthorized modification, and unauthorized reading of SSPs, control and status data.

The operating system, as deployed on the tested platform and vendor-affirmed platforms, is hardened during the manufacturing process. The operating system ensures process isolation. Each process is given its own private address space and cannot directly access the memory or resources of another process. This ensures each instance of the module has control over its own SSPs. Any attempt by a process to access another process's memory results in a segmentation fault or access violation.

The operating system uses discretionary access control enforced through user IDs, group IDs, and access control lists to protect against unauthorized execution, reading or writing the syslog. A defined role with associated restrictive permissions can be configured to have exclusive rights to execute or modify the module, to modify the SSPs within the module boundary via the approved services, and to read and write audit data to the syslog, Each user logs in to the operating system with a user ID and password.

Process isolation and discretionary access control prevent all operators and running processes, not in the defined role with exclusive rights, from modifying, loading, and executing the cryptographic module. User processes cannot read or write to SSPs owned by other processes and to SSPs within the module boundary.

The operating system provides the Linux Audit subsystem to monitor the syslog file for read and write access events. The cryptographic module provides events to be recorded by the Linux Audit subsystem: security-relevant function and request to access authentication data within the cryptographic module. This audit mechanism can audit access to the syslog used by the cryptographic module.

7 Physical Security

N/A for this module.

8 Non-Invasive Security

N/A for this module.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
RAM	System Memory	Dynamic

Table 16: Storage Areas

SSPs are provided to the module by the calling process and are destroyed when released by the appropriate zeroisation function calls. The module does not perform persistent storage of SSPs.

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
SSP Input	Calling process	RAM	Plaintext	Automated	Electronic	
SSP Output	RAM	Calling process	Plaintext	Automated	Electronic	

Table 17: SSP Input-Output Methods

The module does not support manual SSP entry or intermediate key generation output. The module does not support entry and output of SSPs beyond the physical perimeter of the operational environment. Except for services designed to wrap or unwrap an SSP the SSPs are provided to the module via API input parameters in the plaintext form and output via API output parameters in the plaintext form to and from the calling application running on the same operational environment. SSPs provided for unwrapping are input encrypted using KTS-IFC's RSA-OAEP_basic, and SSPs the module wrapped are output encrypted using KTS-IFC's RSA-OAEP_basic.

The output of plaintext CSPs requires two independent internal actions. Specifically, the first action is creation of the cipher context to request the service and to hold the CSPs to be output from the module. The second action is to process the 'Key Derivation (SSH)', 'Key Derivation (IKE), Key Derivation (SP 800-108r1), 'Key Encapsulation', 'Key Un-encapsulation', or 'Key Generation' service request using the context created. Only after successful completion of this request, the generated CSP is output via the API output parameter.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
API call	The zeroisation is performed by the module overwriting zeroes to the memory location occupied by the SSP and further deallocating that area.	The calling application, interacting with the module, is responsible for calling the appropriate destruction functions using the zeroisation APIs listed in the above table to zeroise the calling application's copies of the SSP. The completion of a zeroisation routine will indicate that a zeroisation procedure succeeded	By invocation through API call
Module Restart	The zeroisation is performed by erasing the memory location occupied by the SSP and further deallocating that area.	Restart to zeroize the Hashed Password	Restart the module

Table 18: SSP Zeroization Methods

The zeroisation is performed by the module overwriting zeroes or predefined values to the memory location occupied by the SSP and further deallocating that area. The calling application, interacting with the module, is responsible for calling the appropriate destruction functions using the zeroisation APIs listed in the above table to zeroise the calling application's copies of the SSP. The completion of a zeroisation routine will indicate that a zeroisation procedure succeeded.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
800-108 Derived Key	Keying material derived from key derivation function SP 800-108rev1	128, 192, 256 - 128, 192, 256	Keying Material - CSP	KDF-SP800-108		
Key Derivation Key	Key-Derivation key for SP 800-108rev1	128, 192, 256 - 128, 192, 256	Key-derivation Key - CSP			KDF-SP800-108
AES Keys	AES Keys	128, 192, 256 - 128, 192, 256	Symmetric Keys - CSP	KDF-TLS KDF-SSH KDF-IKE		Encryption Authenticated Encryption MAC2 Authenticated Decryption Decryption
Crypto Officer Password	Crypto Officer Password used during the authentiication	N/A - N/A	Authentication Password - CSP			MAC1
Hashed Password	Hash of the Crypto Officer Password	N/A - N/A	Authentication Password - CSP	MAC1		MAC1
DRBG Entropy Input	Entropy material for DRBG.	384 - 384	DRBG material - CSP			DRBG
DRBG Seed	Seeding material for DRBG.	256 - 256	DRBG material - CSP	DRBG		DRBG
DRBG C Value	Used for DRBG.	256 - 256	DRBG material - CSP	Hash DRBG (A4984)		DRBG
DRBG V Value	Used for DRBG.	256 - 256	DRBG material - CSP	DRBG		DRBG
DRBG Key Value	Used for DRBG.	256 - 256	DRBG material - CSP	Counter DRBG (A4984) HMAC DRBG (A4984)		DRBG
ECDSA key pair	ECDSA key pair	P-256, P-384, P-521 - 128, 192, 256	Asymmetric key - CSP	Asymmetric Key Pair Generation		Asymmetric Key Verification Digital Signature Generation Digital Signature Verification
HMAC key	HMAC key used for message authentication	112-bits or greater - 112-bits or greater	HMAC Key - CSP	KDF-TLS KDF-SSH KDF-IKE KDF-SP800-108		MAC1 MAC2 MAC3

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
IKE shared secret	IKE shared secret for IKE key agreement	112-256 - 112-256	Shared Secret - CSP			KDF-IKE
IKE Derived key/AES & HMAC	IKE Derived Keys for IKE key agreement	112 or greater - 112 or greater	Symmetric keys - CSP	KDF-IKE		Encryption Authenticated Encryption Authenticated Decryption Decryption
Keying Material	KTS-IFC keying material to be encapsulated or un-encapsulated by RSA-OAEP_basic	112 or greater - 112 or greater	Keying Material - CSP		KTS-IFC	KTS-IFC
Shared Secret	Shared Secret for Key Agreement	112 or greater - 112 or greater	Bitstring - CSP		KAS-ECC-SSC KAS-FFC-SSC	
SSH Shared Secret	SSH Shared Secret for SSH Key Agreement	112 or greater - 112 or greater	Bitstring - CSP			KDF-SSH
SSH Derived key/AES & HMAC	SSH Derived keys for SSH key agreement	112 or greater - 112 or greater	Symmetric Keys - CSP	KDF-SSH		Encryption Authenticated Encryption Authenticated Decryption Decryption
TLS Derived key/AES & HMAC	TLS Derived keys for TLS key agreement	112 or greater - 112 or greater	Symmetric Keys - CSP	KDF-TLS		Encryption Authenticated Encryption Authenticated Decryption Decryption
TLS master secret	TLS Derived Keys for TLS key agreement	112-256 - 112-256	TLS Keys - CSP	KDF-TLS		
TLS pre-master secret	TLS Derived Keys for TLS key agreement	112-256 - 112-256	TLS Keys - CSP			KDF-TLS
RSA Key Pair	RSA Key Pair	2048, 3072, 4096 - 112, 128, 152	Asymmetric Keys - CSP	Asymmetric Key Pair Generation		KTS-IFC Digital Signature Generation Digital Signature Verification
DH Private Key	DH private key	2048, 3072, 4096, 8192 - 112, 128, 152, 200	Asymmetric Key - CSP	KAS-FFC-SSC		KAS-FFC-SSC

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DH Public Key	DH public key	2048, 3072, 4096, 8192 - 112, 128, 152, 200	Asymmetric Key - PSP	KAS-FFC-SSC		KAS-FFC-SSC
ECDH Private Key	ECDH private key	P-256, P-384, P-521 - 128-256	Asymmetric Key - CSP	KAS-ECC-SSC		KAS-ECC-SSC
ECDH Public Key	ECDH public key	P-256, P-384, P-521 - 128-256	Asymmetric Key - PSP	KAS-ECC-SSC		KAS-ECC-SSC

Table 19: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
800-108 Derived Key	SSP Output	RAM:Plaintext	Ephemeral	API call	Key Derivation Key:Derived From
Key Derivation Key	SSP Input	RAM:Plaintext	Ephemeral	API call	800-108 Derived Key:Derives
AES Keys	SSP Input	RAM:Plaintext	Ephemeral	API call	IKE shared secret:Derived From Shared Secret:Derived From SSH Shared Secret:Derived From
Crypto Officer Password	SSP Input	RAM:Plaintext	Ephemeral	API call	Hashed Password:Used With
Hashed Password		RAM:Plaintext	Ephemeral	Module Restart	Crypto Officer Password:Used With
DRBG Entropy Input	SSP Input	RAM:Plaintext	Ephemeral	API call	DRBG Seed:Used With DRBG C Value:Used With DRBG V Value:Used With DRBG Key Value:Used With
DRBG Seed		RAM:Plaintext	Ephemeral	API call	DRBG Entropy Input:Used With
DRBG C Value		RAM:Plaintext	Ephemeral	API call	DRBG Entropy Input:Used With
DRBG V Value		RAM:Plaintext	Ephemeral	API call	DRBG Entropy Input:Used With
DRBG Key Value		RAM:Plaintext	Ephemeral	API call	DRBG Entropy Input:Used With
ECDSA key pair	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call	DRBG V Value:Used With DRBG Seed:Used With DRBG C Value:Used With DRBG Key Value:Used With
HMAC key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call	Shared Secret:Derived From
IKE shared secret	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call	IKE Derived key/AES & HMAC:Used With DH Private Key:Used With DH Public Key:Used With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
					ECDH Private Key:Used With ECDH Public Key:Used With
IKE Derived key/AES & HMAC	SSP Output	RAM:Plaintext	Ephemeral	API call	IKE shared secret:Derived From
Keying Material	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call	RSA Key Pair:Used With
Shared Secret	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call	DH Private Key:Used With DH Public Key:Used With ECDH Private Key:Used With ECDH Public Key:Used With
SSH Shared Secret	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call	SSH Derived key/AES & HMAC:Used With DH Private Key:Used With DH Public Key:Used With ECDH Private Key:Used With ECDH Public Key:Used With
SSH Derived key/AES & HMAC	SSP Output	RAM:Plaintext	Ephemeral	API call	SSH Shared Secret:Derived From
TLS Derived key/AES & HMAC	SSP Output	RAM:Plaintext	Ephemeral	API call	TLS master secret:Derived From TLS pre-master secret:Used With
TLS master secret	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call	TLS Derived key/AES & HMAC:Used With TLS pre-master secret:Derived From
TLS pre-master secret	SSP Input	RAM:Plaintext	Ephemeral	API call	TLS Derived key/AES & HMAC:Used With TLS master secret:Used With
RSA Key Pair	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call	DRBG Seed:Used With DRBG C Value:Used With DRBG V Value:Used With DRBG Key Value:Used With Keying Material:Encrypts
DH Private Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call	DRBG Seed:Used With DRBG C Value:Used With DRBG Key Value:Used With Shared Secret:Used With
DH Public Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call	DH Private Key:Paired With
ECDH Private Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call	DRBG Seed:Used With DRBG C Value:Used With DRBG V Value:Used With DRBG Key Value:Used With Shared Secret:Used With
ECDH Public Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call	ECDH Private Key:Paired With

Table 20: SSP Table 2

Intermediate key generation values are never output from the module but are treated like CSPs and are automatically zeroised once no longer needed.

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A4984)	128-bit hardcoded key	Compare Hash Results	SW/FW Integrity	Error message to stdout	Single encompassing message authentication code

Table 21: Pre-Operational Self-Tests

The module performs pre-operational tests automatically when the module is powered on. The pre-operational self-tests ensure that the module is not corrupted and that the cryptographic algorithms work as expected. The module transitions to the operational state only after the pre-operational self-tests (and the cryptographic algorithm self-tests, which in this module are executed automatically after the pre-operational self-tests) are passed successfully.

The types of pre-operational self-tests are described in the next sub-section.

Pre-Operational Software Integrity Test

The HMAC-SHA2-256 Conditional CAST is performed before checking the module integrity. Then the integrity of the software component of the module is verified according to Section 5, using HMAC-SHA2-256. If the comparison verification fails, the module transitions to the error state (Section 10.4).

Pre-Operational Bypass and Critical Functions Tests

The module does not implement pre-operational bypass or critical functions tests. We note that the entropy source is not within the cryptographic boundary of the module, instead passively receiving entropy from the external entropy source. Thus, its critical functions tests are not included in the module.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-ECB (A4984)	128	KAT	CAST	Error message to stdout	Encrypt/Decrypt	Power-up
AES-GCM (A4984)	256	KAT	CAST	Error message to stdout	Encrypt/Decrypt	Power-up
AES-CCM (A4984)	192	KAT	CAST	Error message to stdout	Encrypt/ Decrypt	Power-up
AES-XTS Testing Revision 2.0 (A4984)	128, 256	KAT	CAST	Error message to stdout	Encrypt/ Decrypt	Power-up

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CMAC (A4984)	128, 192, 256	KAT	CAST	Error message to stdout	Generate/Verify	Power-up
Counter DRBG (A4984)	Chained instantiate, reseed, generate	KAT	CAST	Error message to stdout	SP 800-90A section 11.3 health tests	Power-up
Hash DRBG (A4984)	Chained instantiate, reseed, generate	KAT	CAST	Error message to stdout	SP 800-90A section 11.3 health tests	Power-up
HMAC DRBG (A4984)	Chained instantiate, reseed, generate	KAT	CAST	Error message to stdout	SP 800-90A section 11.3 health tests	Power-up
ECDSA SigGen (FIPS186-4) (A4984)	Curve P-256, P-384, P-521	KAT	CAST	Error message to stdout	Sign	Power-up
ECDSA SigVer (FIPS186-4) (A4984)	Curve P-256, P-384, P-521	KAT	CAST	Error message to stdout	Verify	Power-up
HMAC-SHA-1 (A4984)	HMAC-SHA-1	KAT	CAST	Error message to stdout	Generate	Power-up
HMAC-SHA2-224 (A4984)	HMAC-SHA2-224	KAT	CAST	Error message to stdout	Generate	Power-up
HMAC-SHA2-256 (A4984)	HMAC-SHA2-256	KAT	CAST	Error message to stdout	Generate	Power-up
HMAC-SHA2-384 (A4984)	HMAC-SHA2-384	KAT	CAST	Error message to stdout	Generate	Power-up
HMAC-SHA2-512 (A4984)	HMAC-SHA2-512	KAT	CAST	Error message to stdout	Generate	Power-up
KAS-ECC-SSC Sp800-56Ar3 (A4984)	P-224 and P-256 curves	KAT	CAST	Error message to stdout	Shared Secret 'z' computation	Power-up
KAS-FFC-SSC Sp800-56Ar3 (A4984)	ffdhe2048 safe prime group	KAT	CAST	Error message to stdout	Shared Secret 'z' computation	Power-up
KDF SP800-108 (A4984)	Counter mode	KAT	CAST	Error message to stdout	Derive	Power-up
KDF IKEv1 (A4984)	N/A	KAT	CAST	Error message to stdout	Derive	Power-up
KDF IKEv2 (A4984)	N/A	KAT	CAST	Error message to stdout	Derive	Power-up
KTS-IFC (A4984)	2048	KAT	CAST	Error message to stdout	Encrypt/Decrypt	Power-up

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
RSA SigGen (FIPS186-4) (A4984)	2048; PKCS 1.5 & PSS; SHA2-224, SHA2-256, SHA2-384, SHA2-512	KAT	CAST	Error message to stdout	Sign	Power-up
RSA SigVer (FIPS186-4) (A4984)	2048; PKCS 1.5 & PSS; SHA2-224, SHA2-256, SHA2-384, SHA2-512	KAT	CAST	Error message to stdout	Verify	Power-up
SHA2-224 (A4984)	N/A	KAT	CAST	Error message to stdout	Generate	Power-up
SHA2-256 (A4984)	N/A	KAT	CAST	Error message to stdout	Generate	Power-up
SHA2-512 (A4984)	N/A	KAT	CAST	Error message to stdout	Generate	Power-up
KDF SSH (A4984)	N/A	KAT	CAST	Error message to stdout	Derive	Power-up
KDF TLS (A4984)	TLS 1.0/1.1	KAT	CAST	Error message to stdout	Derive	Power-up
TLS v1.2 KDF RFC7627 (A4984)	TLS 1.2 SHA2-256, SHA2-512	KAT	CAST	Error message to stdout	Derive	Power-up
ECDSA KeyGen (FIPS186-4) (A4984)	Generated keypair	Sign/Verify	PCT	Error message to stdout	Sign/Verify	Keypair generated
KAS-ECC KeyPairGen	Generated keypair	SP 800-56Arev3 assurance checks	PCT	Error message to stdout	SP 800-56Arev3 assurance checks	Keypair generation
KAS-FFC KeyPairGen	Generated keypair	SP 800-56Arev3 assurance checks	PCT	Error message to stdout	SP 800-56Arev3 assurance checks	Keypair generation
ECDSA KeyVer (FIPS186-4) (A4984)	Generated keypair	Public key verify	PCT	Stdout, syslog message	Public key validity	Keypair generated
RSA KeyGen (FIPS186-4) (A4984)	Generated keypair	Sign/Verify	PCT	Error message to stdout	Sign/Verify	Keypair generated

Table 22: Conditional Self-Tests

Cryptographic Algorithm Self-Tests

The module performs self-tests on FIPS-Approved cryptographic algorithms supported in the approved mode of operation, using the tests shown in (and indicated as CASTs) and using the provision of IG 10.3.A and IG 10.3.B for optimization of the number of self-tests. Data output through the data output interface is inhibited during the self-tests. The cryptographic algorithm self-tests are performed in the form of Known Answer Tests (KATs), in which the calculated output is compared with the expected known answer (that are hard-coded in the module). A failed match causes a failure of the self-test.

If any of these self-tests fails, the module transitions to error state and is aborted.

Conditional Pairwise Consistency Tests

The module implements RSA and ECDSA key generation service and performs the respective pairwise consistency test using sign and verify functions when the keys are generated (Table 27). In addition, SP 800-56a Rev3 conditional tests are run when ephemeral keypairs are created for key agreement.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A4984)	Compare Hash Results	SW/FW Integrity	Every time the module is loaded	Start the module

Table 23: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-ECB (A4984)	KAT	CAST	During the module loading	Load the module
AES-GCM (A4984)	KAT	CAST	During the module loading	Load the module
AES-CCM (A4984)	KAT	CAST	During the module loading	Load the module
AES-XTS Testing Revision 2.0 (A4984)	KAT	CAST	During the module loading	Load the module
AES-CMAC (A4984)	KAT	CAST	During module loading	Load the module
Counter DRBG (A4984)	KAT	CAST	During the module loading	Load the module
Hash DRBG (A4984)	KAT	CAST	During the module loading	Load the module
HMAC DRBG (A4984)	KAT	CAST	During the module loading	Load the module
ECDSA SigGen (FIPS186-4) (A4984)	KAT	CAST	During the module loading	Load the module
ECDSA SigVer (FIPS186-4) (A4984)	KAT	CAST	During the module loading	Load the module
HMAC-SHA-1 (A4984)	KAT	CAST	During the module loading	Load the module
HMAC-SHA2-224 (A4984)	KAT	CAST	During the module loading	Load the module
HMAC-SHA2-256 (A4984)	KAT	CAST	During the module loading	Load the module
HMAC-SHA2-384 (A4984)	KAT	CAST	During the module loading	Load the module
HMAC-SHA2-512 (A4984)	KAT	CAST	During the module loading	Load the module
KAS-ECC-SSC Sp800-56Ar3 (A4984)	KAT	CAST	During the module loading	Load the module

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
KAS-FFC-SSC Sp800-56Ar3 (A4984)	KAT	CAST	During the module loading	Load the module
KDF SP800-108 (A4984)	KAT	CAST	During the module loading	Load the module
KDF IKEv1 (A4984)	KAT	CAST	During the module loading	Load the module
KDF IKEv2 (A4984)	KAT	CAST	During the module loading	Load the module
KTS-IFC (A4984)	KAT	CAST	During the module loading	Load the module
RSA SigGen (FIPS186-4) (A4984)	KAT	CAST	During the module loading	Load the module
RSA SigVer (FIPS186-4) (A4984)	KAT	CAST	During the module loading	Load the module
SHA2-224 (A4984)	KAT	CAST	During the module loading	Load the module
SHA2-256 (A4984)	KAT	CAST	During the module loading	Load the module
SHA2-512 (A4984)	KAT	CAST	During the module loading	Load the module
KDF SSH (A4984)	KAT	CAST	During the module loading	Load the module
KDF TLS (A4984)	KAT	CAST	During the module loading	Load the module
TLS v1.2 KDF RFC7627 (A4984)	KAT	CAST	During the module loading	Load the module
ECDSA KeyGen (FIPS186-4) (A4984)	Sign/Verify	PCT	N/A	N/A
KAS-ECC KeyPairGen	SP 800-56Arev3 assurance checks	PCT	N/A	N/A
KAS-FFC KeyPairGen	SP 800-56Arev3 assurance checks	PCT	N/A	N/A
ECDSA KeyVer (FIPS186-4) (A4984)	Public key verify	PCT	N/A	N/A
RSA KeyGen (FIPS186-4) (A4984)	Sign/Verify	PCT	N/A	N/A

Table 24: Conditional Periodic Information

On demand self-tests can be invoked by powering-off and reloading the module. This service performs the same pre-operational test that includes integrity test and cryptographic algorithm tests executed during power-up. The integrity test can also be performed on demand by calling the FIPS_check_incore_fingerprint function. During the execution of the on-demand self-tests, cryptographic services are not available, and no data output or input is possible.

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Conditional Error	Conditional Error state reached when a conditional test fails.	Conditional test failure	The module generates a new key and tests the key via a PCT. If the test fails, an error is returned.	Error message is placed into the error queue. An error code is returned from the key generation function
PreOp Error	PreOp Error state reached when a pre-operational test fails.	Pre-operational test failure	The module is aborted - restart module	Self-test function returns a return code 0. Error message "FATAL FIPS SELFTEST FAILURE" is output on stdout

Table 25: Error States

If the module fails any of the self-tests, the module enters the error state. In the error state, the module outputs the error through the status output interface and the abort function is called that raises the SIGABRT signal, causing the program termination such that the module is no longer operational. In the error state, as the module is no longer operational the data output interface is inhibited. In order to recover from the Error state, the module needs to be rebooted.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

Upon initializing the module by installing the module and setting the password, the operator must then manually set the module to approved mode, via the interface described in Section “2.4 Modes of Operation”.

The cryptographic module is the fipscanister.o file, though Arista does not distribute this file on its own. Instead, it is embedded into the shared library libcrypto.so which is part of OpenSSL, which in turn is distributed as part of the EOS product, in the image accessible through the Arista software downloads website. The product includes the EOS operating system, applications, OpenSSL, libcrypto.so, and fipscanister.o. While there is no need for the fipscanister.o library to be built by the user at any point in time, the file can be verified as the correct one by comparing the SHA2-256 hash sum. The SHA2-256 hash should be 65fcbf0765791ca942d703b5a38d85efe7f84bf6409d9085167ec02f928d80e4. In the Arista build process for building OpenSSL, this fipscanister.o file is linked into OpenSSL’s libcrypto.so shared library file and OpenSSL is configured to use it.

When downloading the product image, the SHA2-256 hash of the image is also made available. When an authorized operator downloads the product image, they can also download the hash file and compare the SHA2-256 hash of the product image to the one listed in the file to make sure that downloaded image is correct. Then they can install the product image onto the host or virtual machine.

Upon completion of installation, the user can confirm that the correct module has been installed by running the “show version” service by calling “show management security” from the CLI which should display “Arista Crypto Module Lvl2 v1.0”. Finally ensure correct operation of the module by running the on-demand self-test service as specified in Section 5 by calling “openssl --fips” from bash.

11.2 Administrator Guidance

All the hardware platforms listed in Table 3 are already hardened by manufacturer following the Arista EOS Hardening Guide to prevent all operators and running processes from modifying running cryptographic processes, and to prevent processes in user groups from gaining read, write or execute access to process not owned by them.

Role-based access control shall be used to control access to the module. Only user of the module shall be configured to use the built-in “network-admin” role:

```
switch(config) #aaa authorization commands all default local
switch(config) #aaa authorization exec default local
switch(config) #user <os-user> secret <password> role network-admin
```

The module contents are considered protected only after the above role-based access control is configured thus only “network-admin” role can access the CLI of a hardware platform.

The operating system shall be configured to audit read and write access to the syslog file /var/log/messages:

```
switch(config) #auditctl -w /var/log/messages -p rw -k syslog_access
switch(config) #service auditd start
switch(config) #systemctl enable auditd
```

The operating system shall be configured as specified in this section for the module contents to be considered protected.

After loading of the module the operator shall initialize the first time Crypto Officer Password using the FIPS_set_password function. Then to bring the module in the Approved mode, the operator shall call the FIPS_mode_set(1) function.

11.3 Non-Administrator Guidance

None.

11.4 Design and Rules

The module initializes upon power-on. After the pre-operational self-tests (POST) are successfully concluded, the module automatically transitions to the operational state. In this state, the module awaits service requests from the operator.

The operator must then manually set the module to approved mode, via the interface described in Section “2.4 Modes of Operation”.

11.5 End of Life

To cease using the module, power off the module. The module does not possess persistent storage of SSPs. The SSP value only exists in volatile memory and that value vanishes when the module is powered off. So as a first step for the secure sanitization, the module needs to be powered off. Then for actual deprecation, the module will be upgraded to a newer version that is approved. This upgrade process will uninstall/remove the old/terminated module and provide a new replacement.

12 Mitigation of Other Attacks

N/A

13 References and Definitions

The following standards are referred to in this Security Policy.

Abbreviation	Full Specification Name
[NIST]	National Institute of Standards and Technology
[FIPS140-3]	Security Requirements for Cryptographic Modules, March 22, 2019
[IG]	Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program
[ISO19790]	Information technology – Security techniques – Security requirements for cryptographic modules, 2012(2014)
[38A]	NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation, December 2001
[38B]	NIST Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, May 2005
[38C]	NIST Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, May 2004
[38D]	NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, November 2007
[38E]	NIST Special Publication 800-38E, Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices, January 2010
[38F]	NIST Special Publication 800-38F, Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, December 2012
[56Ar3]	NIST Special Publication 800-56A Revision 3, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, April 2018
[56Ar2]	NIST Special Publication 800-56A Revision 2, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, May 2013
[56Br2]	NIST Special Publication 800-56B Revision 2, Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography, March 2019
[67]	NIST Special Publication 800-67 Revision 2, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, November 2017
[90A]	NIST Special Publication 800-90A Revision 1, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, June 2015.
[90B]	NIST Special Publication 800-90B, Recommendation for the Entropy Sources Used for Random Bit Generation, January 2018
[90C]	(Second Draft) NIST Special Publication 800-90C, Recommendation for Random Bit Generator (RBG) Constructions, April 2016
[108]	NIST Special Publication 800-108, Recommendation for Key Derivation Using Pseudorandom Functions (Revised), October 2009
[131A]	NIST Special Publication 800-131A Revision 2, Transitioning the Use of Cryptographic Algorithms and Key Lengths, March 2019

Abbreviation	Full Specification Name
[132]	NIST Special Publication 800-132, Recommendation for Password-Based Key Derivation, Part 1: Storage Applications, December 2010
[133]	NIST Special Publication 800-133 Revision 2, Recommendation for Cryptographic Key Generation, June 2020
[135]	NIST Special Publication 800-135 Revision 1, Recommendation for Existing Application-Specific Key Derivation Functions, December 2011
[180]	Federal Information Processing Standards Publication 180-4, Secure Hash Standard (SHS), August 2015
[186]	Federal Information Processing Standards Publication 186-4, Digital Signature Standard (DSS), July 1 2013
[197]	Federal Information Processing Standards Publication 197, Advanced Encryption Standard (AES), November 26, 2001
[198]	Federal Information Processing Standards Publication 198-1, The Keyed-Hash Message Authentication Code (HMAC), July 2008
[202]	Federal Information Processing Standards Publication 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, August 2015
[RFC 4581]	IETF, The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST), May 2007