Arista Networks, Inc.

Arista Crypto Module v3.0 [Software, Software IPsec]

FIPS 140-3 Non-Proprietary Security Policy

Document Version: 1.5
Date: October 13, 2025

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 3.0 of the Arista Networks Inc. Arista Crypto Module v3.0 [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 1 module.

1.2 Security Levels

Section	Title	Security Level
1	General	1
2	Cryptographic module specification	1
3	Cryptographic module interfaces	1
4	Roles, services, and authentication	2
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 Arista Crypto Module v3.0 [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: Multi-Chip Standalone

Module Characteristics:

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)

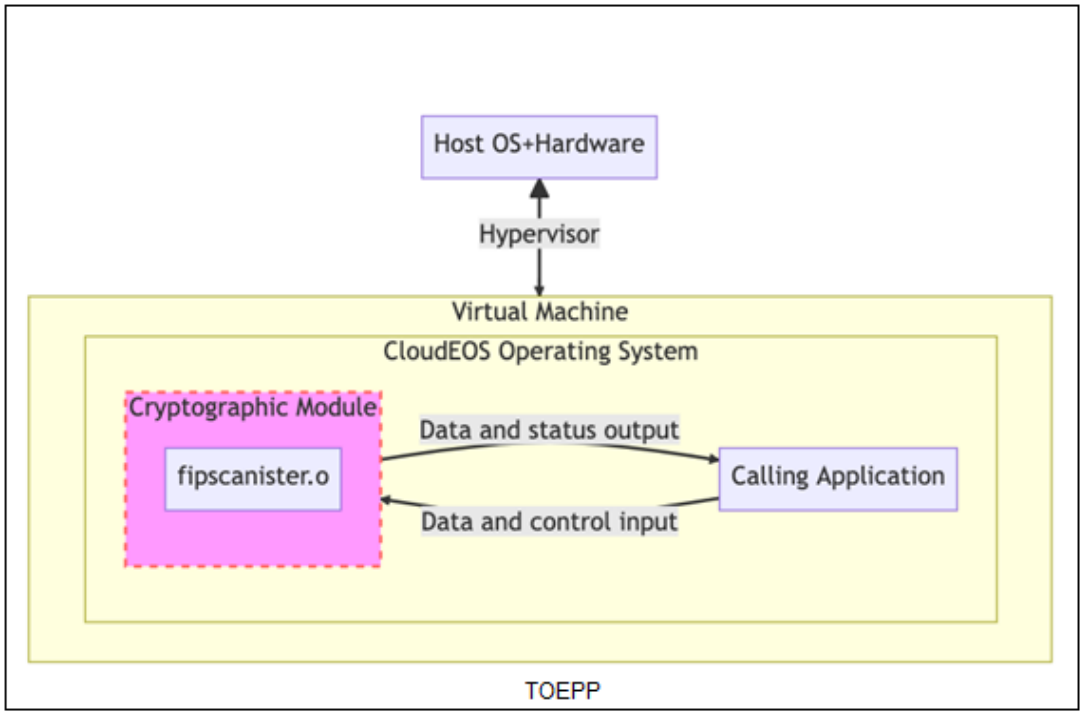

Figure 1 – Block Diagram

The block diagram depicts the cryptographic boundary (in pink) 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

Tested Module Identification – Hardware:

N/A for this module.

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
fipscanister.o	3.0	None	Message authentication with HMAC-SHA2-256

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

The module operates in a modifiable operational environment. The module runs on a commercially available virtual machine, based on a general-purpose operating system. The module executes on the hardware specified in Section 2.2. The module does not support concurrent operators.

Tested Module Identification – Hybrid Disjoint Hardware:

N/A for this module.

Tested Operational Environments - Software, Firmware, Hybrid:

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.

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
CloudEOS version 4.29 running on QEMU version 2.0.0 running on Linux 3.10.0-1160.el7.x86_64	Supermicro SYS-1029U-TR-CTO	Intel Xeon Gold 6240R	Yes	None	3.0
CloudEOS version 4.29 running on QEMU version 2.0.0 running on Linux 3.10.0-1160.el7.x86_64	Supermicro SYS-1029U-TR-CTO	Intel Xeon Gold 6240R	No	None	3.0

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

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

Operating System	Hardware Platform
CloudEOS	Any general-purpose computer (GPC)
Any compatible OS	Any general-purpose computer (GPC)

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 - Selected by calling the FIPS_mode_set(1) function.	Approved	Per service indication
Non-Approved Mode	Single Non-Approved Mode - Selected 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. Section 4.3 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). To validate that the Approved Mode is active, call FIPS_mode() and verify the return value is equal to “1”.

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	A3592	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A3592	Key Length - 128, 192, 256 Tag Length - 112, 128, 32, 48, 64, 80, 96 IV Length - IV Length: 56 Payload Length - Payload Length: 0-256 Increment 8 AAD Length - AAD Length: 0-524288 Increment 8	SP 800-38C
AES-CFB1	A3592	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB128	A3592	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A3592	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A3592	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B

Algorithm	CAVP Cert	Properties	Reference
		MAC Length - MAC Length: 16-128 Increment 8 Message Length - Message Length: 16-65536 Increment 8	
AES-CTR	A3592	Direction - Decrypt, Encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 8-128 Increment 8 Supports Counter larger than maximum value - Yes Incremental Counter - Yes Counter Tests Performed - Yes	SP 800-38A
AES-ECB	A3592	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A3592	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 Payload Length - Payload Length: 256, 384, 136, 392 AAD Length - AAD Length: 0, 256, 384, 136, 392	SP 800-38D
AES-XTS Testing Revision 2.0	A3592	Direction - Decrypt, Encrypt Key Length - 128, 256 Payload Length - Payload Length: 128-65536 Increment 128 Tweak Mode - Hex Data Unit Length - Data Unit Length: 128-65536 Increment 128 Data Unit Length Matches Payload Length - No	SP 800-38E
Counter DRBG	A3592	Prediction Resistance - No, Yes Supports Reseed - Yes Mode - AES-128, AES-192, AES-256 Derivation Function Enabled - No, Yes Additional Input - Additional Input: 0, 128, Additional Input: 0, 256, Additional Input: 0, 320, Additional Input: 0, 384 Entropy Input - Entropy Input: 128, Entropy Input: 192, Entropy Input: 256, Entropy Input: 320, Entropy Input: 384 Nonce - Nonce: 0, Nonce: 128, Nonce: 256, Nonce: 64 Personalization String Length - Personalization String Length: 0, 128, Personalization String Length: 0, 256, Personalization String Length: 0, 320, Personalization String Length: 0, 384 Returned Bits - 512	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-4)	A3592	Curve - P-256, P-384, P-521 Secret Generation Mode - Testing Candidates	FIPS 186-4
ECDSA KeyVer (FIPS186-4)	A3592	Curve - P-256, P-384, P-521	FIPS 186-4

Algorithm	CAVP Cert	Properties	Reference
ECDSA SigGen (FIPS186-4)	A3592	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)	A3592	Component - No Curve - P-256, P-384, P-521 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
Hash DRBG	A3592	Prediction Resistance - No, Yes Supports Reseed - Yes Mode - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512 Entropy Input - Entropy Input: 128, Entropy Input: 192, Entropy Input: 256 Nonce - Nonce: 128, Nonce: 64, Nonce: 96 Personalization String Length - Personalization String Length: 0, 128, Personalization String Length: 0, 192, Personalization String Length: 0, 256 Additional Input - Additional Input: 0, 128, Additional Input: 0, 192, Additional Input: 0, 256 Returned Bits - 1024, 1536, 2048, 640, 896	SP 800-90A Rev. 1
HMAC DRBG	A3592	Prediction Resistance - No, Yes Supports Reseed - Yes Mode - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512 Entropy Input - Entropy Input: 128, Entropy Input: 192, Entropy Input: 256 Nonce - Nonce: 128, Nonce: 64, Nonce: 96 Personalization String Length - Personalization String Length: 0, 128, Personalization String Length: 0, 192, Personalization String Length: 0, 256 Additional Input - Additional Input: 0, 128, Additional Input: 0, 192, Additional Input: 0, 256 Returned Bits - 1024, 1536, 2048, 640, 896	SP 800-90A Rev. 1
HMAC-SHA-1	A3592	MAC - MAC: 80-160 Increment 8 Key Length - Key Length: 256-2048 Increment 8	FIPS 198-1
HMAC-SHA2-224	A3592	MAC - MAC: 112-224 Increment 16 Key Length - Key Length: 256-2048 Increment 8	FIPS 198-1
HMAC-SHA2-256	A3592	MAC - MAC: 128-256 Increment 64 Key Length - Key Length: 256-2048 Increment 8	FIPS 198-1
HMAC-SHA2-384	A3592	MAC - MAC: 192-384 Increment 64 Key Length - Key Length: 256-2048 Increment 8	FIPS 198-1
HMAC-SHA2-512	A3592	MAC - MAC: 256-512 Increment 64 Key Length - Key Length: 256-2048 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A3592	Domain Parameter Generation Methods - P-256, P-384, P-521 Hash Function Z - SHA2-512	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
		Scheme - ephemeralUnified - KAS Role - initiator, responder	
KAS-FFC-SSC Sp800-56Ar3	A3592	Domain Parameter Generation Methods - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Hash Function Z - SHA2-384 Scheme - dhEphem - KAS Role - initiator, responder	SP 800-56A Rev. 3
KDF IKEv1 (CVL)	A3592	Authentication Method - Pre-shared Key Initiator Nonce Length - Initiator Nonce Length: 256-2048 Increment 128 Responder Nonce Length - Responder Nonce Length: 256-2048 Increment 128 Preshared Key Length - Preshared Key Length: 8-8192 Increment 8 Diffie-Hellman Shared Secret Length - Diffie-Hellman Shared Secret Length: 1024-8192 Increment 1024 Hash Algorithm - SHA-1, SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
KDF IKEv2 (CVL)	A3592	Initiator Nonce Length - Initiator Nonce Length: 256-2048 Increment 128 Responder Nonce Length - Responder Nonce Length: 256-2048 Increment 128 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 Hash Algorithm - SHA-1, SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
KDF SP800-108	A3592	KDF Mode - Counter MAC Mode - CMAC-AES128, CMAC-AES256 Supported Lengths - Supported Lengths: 128 Fixed Data Order - Before Fixed Data Counter Length - 8 Supports Empty IV - No Custom Key In Length - 0	SP 800-108 Rev. 1
KDF SSH (CVL)	A3592	Cipher - AES-128, AES-192, AES-256 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
KDF TLS (CVL)	A3592	TLS Version - v1.0/1.1	SP 800-135 Rev. 1
KTS-IFC	A3592	Function - keyPairGen, partialVal IUT ID - CAFECAFE Modulo - 2048, 3072, 4096 Key Generation Methods - rsakpg2-basic	SP 800-56B Rev. 2

Algorithm	CAVP Cert	Properties	Reference
		Scheme - KTS-OAEP-basic - KAS Role - initiator, responder Key Transport Method - Hash Algorithms - SHA2-224, SHA2-256, SHA2-384, SHA2-512 Supports Null Associated Data - Yes Associated Data Encoding - concatenation Key Length - 512	
RSA KeyGen (FIPS186-4)	A3592	Key Generation Mode - B.3.3, B.3.6 Modulo - 2048, 3072, 4096 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)	A3592	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA2-256	FIPS 186-4
RSA SigVer (FIPS186-4)	A3592	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA-1 Public Exponent Mode - Random	FIPS 186-4
SHA-1	A3592	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-224	A3592	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A3592	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A3592	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A3592	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
TLS v1.2 KDF RFC7627 (CVL)	A3592	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1

Table 6: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG Section 4	Key Type:Asymmetric	N/A	SP 800-133r2 Section 4, example 1: U is directly output without XORing V

Table 7: Vendor-Affirmed Algorithms

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.

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 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)	PQG Gen, Key Pair Gen, and Sig Gen. DSA Signature Verification and PQG Verification are approved for Legacy use only
RSA (disallowed)	Key Encryption using PKCS#1 v1.5. Key Decryption using PKCS#1 v1.5 is allowed for Legacy use only
AES/TripleDES KW (noncompliant)	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	Used to perform key agreement primitives	IG: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: (A3592)
KAS-FFC-SSC	KAS-SSC	Used to perform key agreement primitives	IG: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: (A3592)
KTS-IFC	AsymKeyPair-Decap AsymKeyPair-Encap	Key encapsulation and un-encapsulation	IG:IG D.G Caveat:Key transport provides between 112 and 152 bits of encryption strength	KTS-IFC: (A3592)
Asymmetric Key Pair Generation	AsymKeyPair-KeyGen CKG	Asymmetric Key Pair Generation performed by ECDSA or RSA		ECDSA KeyGen (FIPS186-4): (A3592) RSA KeyGen (FIPS186-4): (A3592) Counter DRBG: (A3592) Hash DRBG: (A3592) HMAC DRBG: (A3592) CKG Section 4: ()
Asymmetric Key Verification	AsymKeyPair-KeyVer	Asymmetric Key Pair Verification for ECDSA		ECDSA KeyVer (FIPS186-4): (A3592)

Name	Type	Description	Properties	Algorithms
Digital Signature Generation	DigSig-SigGen	Digital Signature Generation using ECDSA or RSA		ECDSA SigGen (FIPS186-4): (A3592) RSA SigGen (FIPS186-4): (A3592) SHA2-224: (A3592) SHA2-256: (A3592) SHA2-384: (A3592) SHA2-512: (A3592)
Digital Signature Verification	DigSig-SigVer	Digital Signature Verification using ECDSA or RSA		ECDSA SigVer (FIPS186-4): (A3592) RSA SigVer (FIPS186-4): (A3592) SHA-1: (A3592) SHA2-224: (A3592) SHA2-256: (A3592) SHA2-384: (A3592) SHA2-512: (A3592)
Encryption	BC-UnAuthEncrypt	Block Cipher Symmetric Encryption Non-Authenticated		AES-CBC: (A3592) AES-CFB1: (A3592) AES-CFB128: (A3592) AES-CFB8: (A3592) AES-CTR: (A3592) AES-ECB: (A3592) AES-XTS Testing Revision 2.0: (A3592)
Authenticated Encryption	BC-AuthEncrypt	Block Cipher Symmetric Encryption Authenticated		AES-CCM: (A3592) AES-GCM: (A3592)

Name	Type	Description	Properties	Algorithms
MAC1	MAC	Message Authentication Computation with HMAC-SHA-1		HMAC-SHA-1: (A3592) Key Length: 256-2048
MAC2	MAC	Message Authentication Computation with AES CMAC and HMAC		AES-CMAC: (A3592) Key Length: 128, 192, 256 HMAC-SHA-1: (A3592) Key Length: 256-2048 HMAC-SHA2-224: (A3592) Key Length: 256-2048 HMAC-SHA2-256: (A3592) Key Length: 256-2048 HMAC-SHA2-384: (A3592) Key Length: 256-2048 HMAC-SHA2-512: (A3592) Key Length: 256-2048
Message Digest	SHA	Message Digest		SHA-1: (A3592) SHA2-224: (A3592) SHA2-256: (A3592) SHA2-384: (A3592) SHA2-512: (A3592)
DRBG	DRBG	Generation of random numbers		Counter DRBG: (A3592) Hash DRBG: (A3592) HMAC DRBG: (A3592)
KDF-TLS	KAS-135KDF	Key derivation for TLS		KDF TLS: (A3592) TLS v1.2 KDF RFC7627: (A3592)
KDF-SSH	KAS-135KDF	Key derivation for SSH		KDF SSH: (A3592)

Name	Type	Description	Properties	Algorithms
KDF-IKE	KAS-135KDF	Key derivation for IKE		KDF IKEv1: (A3592) KDF IKEv2: (A3592)
Authenticated Decryption	BC-AuthDecrypt	Block Cipher Symmetric Decryption Authenticated		AES-CCM: (A3592) AES-GCM: (A3592)
Decryption	BC-UnAuthDecrypt	Block Cipher Symmetric Decryption		AES-CBC: (A3592) AES-CFB1: (A3592) AES-CFB128: (A3592) AES-CFB8: (A3592) AES-CTR: (A3592) AES-ECB: (A3592) AES-XTS Testing Revision 2.0: (A3592)
MAC3	MAC	MAC computation with HMAC-SHA2-256 used for the Integrity test		HMAC-SHA2-256: (A3592)
KDF-SP800-108	KBKDF	Key Derivation with SP 800-108r1		KDF SP800-108: (A3592)

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

Notes:

- 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.
- The KDF SSH (CVL) shall only be used within the context of the SSHv2 protocol.
- The KDF TLS (CVL) shall only be used within the context of the TLSv1.0/1.1 protocol.
- The TLS v1.2 KDF RFC7627 (CVL) shall only be used within the context of the TLS v1.2 protocol.
- The KDF IKEv1 (CVL) shall only be used within the context of the IKEv1 protocol.
- The KDF IKEv2 (CVL) shall only be used within the context of the IKEv2 protocol.

AES-GCM IV Generation

The module offers three AES GCM implementations. 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 without a derivation function. IG D.L requires that a CTR_DRNG used without a derivation function shall be seeded from an entropy source producing full-entropy outputs, and the entropy source shall be located within the TOEPP. It is the responsibility of the developer integrating the module to ensure that the entropy used to seed the DRBG comes from a source located within the TOEPP and which is capable of producing full entropy output. For Hash DRBG and HMAC DRBG, the developer must ensure that the entropy used to seed the DRBG comes from a source located within the TOEPP and that the hash function used by the Hash DRBG and HMAC DRBG can provide a security strength that meets or exceeds the minimum security strength required for the random bits that the DRBG generates.

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. The module also provides the KTS-IFC OAEP asymmetric encapsulation and decapsulation functions compliant to SP 800-56Br2 per IG D.G.

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 uses an SP800-90Arev1-compliant Deterministic Random Bit Generator (DRBG) using CTR_DRBG with AES-256 and use of derivation function for creation of asymmetric key components and random number generation. The module receives entropy passively and uses 384 bits of entropy to seed the DRBG. Full entropy source must be used for the default DRBG. Operators may instantiate CTR_DRBG instances with other options. Operators may instantiate and use the other Approved Hash_DRBG and HMAC_DRBG offered by the module.

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

Protocol	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-HELLMANGROUP 14-SHA1, 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			

Protocol	Key Exchange	Server/ Host Auth	Cipher	Integrity
	ECDHE	ECDSA	AES-GCM-128	AES-GCM-128
	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 for TLS v1.0, v1.1, v1.2			
	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

Protocol	Key Exchange	Server/ Host Auth	Cipher	Integrity
	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

2.12 Additional Information

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.

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”.

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
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

The table below 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 (A3592)	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 can be set by the calling application through the “FIPS_set_password” API.

The module has procedural controls and enforces that an operator must set a password prior to use of the module. The module is installed according to section 11.1 and the module authentication mechanism is included within the module software and so automatically included during that installation process.

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 password. 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 table below 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	Ciphertext, Authentication Tag, Key, IV	Plaintext	Authenticated Decryption	Crypto Officer - AES Key: W,E - AES GCM Key: W,E - AES GCM IV: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		"initialized"				
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 Key: W,E - AES GCM Key: W,E - AES GCM IV: G,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 Key: W,E
Encryption	Encryption	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	Plaintext, Key	Ciphertext	Encryption	Crypto Officer - AES Key: W,E
Key Derivation (TLS)	Deriving TLS keys	Mode indicator 1, return code 1,	PRF algorithm, TLS	Derived Keys	KDF-TLS	Crypto Officer - TLS Derived

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		syslog message indicating the service "starting", "performed" or "initialized"	master secret			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
Key Derivation (SP 800-108r1)	Deriving keys	Mode indicator 1, return code 1, syslog message indicating	Shared Secret, key size	Derived Keys	KDF-SP800-108	Crypto Officer - Key Derivation Key: W,E - 800-

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		the service "starting", "performed" or "initialized"				108 Derived Key: G,R
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 Encapsulation	Key encapsulation per SP 800-56Br2	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	RSA public key, keying material to encapsulate	Encapsulated key	KTS-IFC	Crypto Officer - RSA KEK Public: W,E - Keying Material : R,W

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Key Decapsulation	Key decapsulation per SP 800-56Br2	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	RSA private key, keying material to decapsulate	Decapsulated key	KTS-IFC	Crypto Officer - RSA KDK Private: W,E - 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 SVK Public: 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	Message, Algorithm, key	Message Authentication code	MAC2	Crypto Officer - AES Key: W,E - HMAC

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		indicating the service "starting", "performed" or "initialized"				key: W,E
Message Digest	Generating message digest	Mode indicator 1, return code 1, syslog message indicating the service "starting", "performed" or "initialized"	Message	Digest of the message	Message Digest	Crypto Officer
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	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					Authenticated Encryption MAC1 MAC2 Message Digest DRBG KDF-TLS KDF-SSH KDF-IKE Authenticated Decryption Decryption MAC3 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	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

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		"starting", "performed" or "initialized"				- 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 indicates Approved mode enabled, 0 is disabled	None	Crypto Officer
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	Message, hash algorithm, private key	Signature	Digital Signature Generation	Crypto Officer - ECDSA SGK Private: W,E - RSA SGK Private: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		"initialized"				
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 SVK Public: W,E - RSA SVK Public: W,E
Zeroise	Zeroise SSP in volatile memory	None	Context containing SSPs	None	None	Crypto Officer - 800-108 Derived Key: Z - AES Key: Z - Crypto Officer Password: Z - Hashed Password: Z - DRBG Entropy Input: Z - DRBG Seed: Z - DRBG C Value: Z - DRBG V Value: Z - DRBG Key Value: Z - HMAC key: Z

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

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- ECDH Private Key: Z - ECDH Public Key: Z - Key Derivation Key: Z - AES GCM Key: Z - AES GCM IV: Z - ECDSA Key Pair: Z - ECDSA SGK Private: Z - ECDSA SVK Public: Z - RSA Key Pair: Z - RSA SGK Private: Z - RSA SVK Public: Z - RSA KDK Private: Z - RSA KEK

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Public: 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, non-approved functions cannot be used. If a non-approved function 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”. If there is no log message, that is an indication that the invoked function was not an approved service.

4.4 Non-Approved Services

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

Name	Description	Algorithms	Role
Decryption	Decryption	Blowfish Camellia 128/192/256 CAST5 DES DES-X IDEA RC2 RC5 SEED Triple-DES	CO
Encryption	Encryption	Blowfish Camellia 128/192/256 CAST5 DES DES-X IDEA RC2 RC5	CO

Name	Description	Algorithms	Role
		SEED Triple-DES	
Key Wrapping	Encrypting/ Decrypting key	RSA (disallowed) AES/TripleDES KW (noncompliant)	CO
Message Digest	Hash computation	MD4 MD5 RIPEMD-160 Whirlpool Triple-DES MAC HMAC-MD5	CO
Signature Generation	Signature Generation	DSA (disallowed)	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 should be installed as stated in section 11.

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 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 Key	AES Key	128, 192, 256 - 128, 192, 256	Symmetric Key - CSP	KDF-TLS KDF-SSH KDF-IKE	KAS-ECC-SSC KAS-FFC-SSC	Encryption Authenticated Encryption MAC2 Authenticated Decryption Decryption
AES GCM Key	AES GCM key for encrypt/decrypt	128, 192, 256 - 128, 192, 256	Symmetric Key - CSP			Authenticated Encryption Authenticated Decryption

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES GCM IV	AES GCM IV for encrypt/decrypt	96 - 96	IV - CSP	DRBG		Authenticated Encryption Authenticated 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	Entropy Input - 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	DRBG		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	DRBG		DRBG
HMAC key	Message authentication	256-2048 - 128-256	HMAC Key - CSP	KDF-TLS KDF-SSH KDF-IKE KDF-SP800-108	KAS-ECC-SSC KAS-FFC-SSC	MAC1 MAC2 MAC3
IKE shared secret	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 Key - CSP	KDF-IKE		Encryption Authenticated Encryption MAC2 Authenticated Decryption Decryption
Keying Material	KTS-IFC keying material to be encapsulated or un-encapsulated	112 or greater - 112 or greater	Keying Material - CSP		KTS-IFC	KTS-IFC

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	by RSA-OAEP_basic					
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 securing SSH connection	112 or greater - 112 or greater	Symmetric Key - CSP	KDF-SSH		Encryption Authenticated Encryption MAC2 Authenticated Decryption Decryption
TLS Derived key/AES & HMAC	TLS derived keys for securing TLS connection	112 or greater - 112 or greater	Symmetric Key - CSP	KDF-TLS		Encryption Authenticated Encryption MAC2 Authenticated Decryption Decryption
TLS master secret	TLS master secret for TLS key derivation	384 - 112-256	TLS Key - CSP	KDF-TLS		
TLS pre-master secret	TLS pre-master secret for TLS key derivation	112-256 - 112-256	TLS Key - CSP		KAS-ECC-SSC KAS-FFC-SSC	KDF-TLS
DH Private Key	Key agreement	2048, 3072, 4096, 8192 - 112, 128, 152, 200	Asymmetric Key - CSP	Asymmetric Key Pair Generation		KAS-FFC-SSC
DH Public Key	DH public key	2048, 3072, 4096, 8192 - 112, 128, 152, 200	Asymmetric Key - PSP	Asymmetric Key Pair Generation		KAS-FFC-SSC
ECDH Private Key	ECDH private key	P-256, P-384, P-521 -	Asymmetric Key - CSP	Asymmetric Key Pair Generation		KAS-ECC-SSC

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		128, 192, 256				
ECDH Public Key	ECDH public key	P-256, P-384, P-521 - 128, 192, 256	Asymmetric Key - PSP	Asymmetric Key Pair Generation		KAS-ECC-SSC
ECDSA Key Pair	ECDSA key pair generation	P-256, P-384, P-521 - 128, 192, 256	Asymmetric Key - CSP	Asymmetric Key Pair Generation		
ECDSA SGK Private	ECDSA signature generation	P-256, P-384, P-521 - 128, 192, 256	Asymmetric Key - CSP			Digital Signature Generation
ECDSA SVK Public	ECDSA signature verification	P-256, P-384, P-521 - 128, 192, 256	Asymmetric Key - PSP			Asymmetric Key Verification Digital Signature Verification
RSA Key Pair	RSA key pair generation	2048, 3072, 4096 - 112, 128, 152	Asymmetric Key - CSP	Asymmetric Key Pair Generation		
RSA SGK Private	RSA signature generation	2048, 3072, 4096 - 112, 128, 152	Asymmetric Key - CSP			Digital Signature Generation
RSA SVK Public	RSA signature verification	2048, 3072, 4096 - 112, 128, 152	Asymmetric Key - PSP			Digital Signature Verification
RSA KDK Private	Asymmetric decapsulation	2048, 3072, 4096 - 112, 128, 152	Asymmetric Key - CSP			KTS-IFC
RSA KEK Public	Asymmetric encapsulation	2048, 3072, 4096 -	Asymmetric Key - PSP			KTS-IFC

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		112, 128, 152				

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 Key	SSP Input	RAM:Plaintext	Ephemeral	API call	IKE shared secret:Derived From Shared Secret:Derived From SSH Shared Secret:Derived From TLS master secret:Derived From
AES GCM Key	SSP Input	RAM:Plaintext	Ephemeral	API call	AES GCM IV:Used With
AES GCM IV		RAM:Plaintext	Ephemeral	API call	AES GCM Key:Used With
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 Seed:Computed From
DRBG V Value		RAM:Plaintext	Ephemeral	API call	DRBG Seed:Computed From
DRBG Key Value		RAM:Plaintext	Ephemeral	API call	DRBG Seed:Computed From
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 KDK Private:Used With RSA KEK Public: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:Derives 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		RAM:Plaintext	Ephemeral	API call	TLS Derived key/AES & HMAC:Derives TLS pre-master secret:Derived From
TLS pre-master secret	SSP Input	RAM:Plaintext	Ephemeral	API call	TLS master secret:Derives
DH Private Key		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 Output	RAM:Plaintext	Ephemeral	API call	DH Private Key:Paired With
ECDH Private Key		RAM:Plaintext	Ephemeral	API call	DRBG Seed:Used With DRBG C Value:Used With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
					DRBG V Value:Used With DRBG Key Value:Used With Shared Secret:Used With
ECDH Public Key	SSP Output	RAM:Plaintext	Ephemeral	API call	ECDH Private Key:Paired With
ECDSA Key Pair	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
ECDSA SGK Private	SSP Input	RAM:Plaintext	Ephemeral	API call	ECDSA SVK Public:Paired With
ECDSA SVK Public	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call	ECDSA SGK Private:Paired With
RSA Key Pair	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
RSA SGK Private	SSP Input	RAM:Plaintext	Ephemeral	API call	RSA SVK Public:Paired With
RSA SVK Public	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call	RSA SGK Private:Paired With
RSA KDK Private	SSP Input	RAM:Plaintext	Ephemeral	API call	Keying Material:Decapsulates RSA KEK Public:Paired With
RSA KEK Public	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call	Keying Material:Encapsulates RSA KDK Private: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 (A3592)	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 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.

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 (A3592)	128	KAT	CAST	Error message to stdout	Encrypt/Decrypt	Power-up
AES-GCM (A3592)	256	KAT	CAST	Error message to stdout	Encrypt/Decrypt	Power-up
AES-CCM (A3592)	192	KAT	CAST	Error message to stdout	Encrypt/ Decrypt	Power-up
AES-XTS Testing Revision 2.0 (A3592)	128, 256	KAT	CAST	Error message to stdout	Encrypt/ Decrypt	Power-up
AES-CMAC (A3592)	128, 192, 256	KAT	CAST	Error message to stdout	Generate/Verify	Power-up

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
Counter DRBG (A3592)	Chained instantiate, reseed, generate	KAT	CAST	Error message to stdout	SP 800-90A section 11.3 health tests	Power-up
Hash DRBG (A3592)	Chained instantiate, reseed, generate	KAT	CAST	Error message to stdout	SP 800-90A section 11.3 health tests	Power-up
HMAC DRBG (A3592)	Chained instantiate, reseed, generate	KAT	CAST	Error message to stdout	SP 800-90A section 11.3 health tests	Power-up
ECDSA SigGen (FIPS186-4) (A3592)	Curve P-384 and SHA-512	KAT	CAST	Error message to stdout	Sign	Power-up
ECDSA SigVer (FIPS186-4) (A3592)	Curve P-384 and SHA-512	KAT	CAST	Error message to stdout	Verify	Power-up
HMAC-SHA-1 (A3592)	HMAC-SHA-1	KAT	CAST	Error message to stdout	Generate	Power-up
HMAC-SHA2-224 (A3592)	HMAC-SHA2-224	KAT	CAST	Error message to stdout	Generate	Power-up
HMAC-SHA2-256 (A3592)	HMAC-SHA2-256	KAT	CAST	Error message to stdout	Generate	Power-up
HMAC-SHA2-384 (A3592)	HMAC-SHA2-384	KAT	CAST	Error message to stdout	Generate	Power-up
HMAC-SHA2-512 (A3592)	HMAC-SHA2-512	KAT	CAST	Error message to stdout	Generate	Power-up
KAS-ECC-SSC Sp800-56Ar3 (A3592)	P-224 and P-256 curves	KAT	CAST	Error message to stdout	Shared Secret 'z' computation	Power-up
KAS-FFC-SSC Sp800-56Ar3 (A3592)	ffdhe2048 safe prime group	KAT	CAST	Error message to stdout	Shared Secret 'z' computation	Power-up
KDF SP800-108 (A3592)	Counter mode	KAT	CAST	Error message to stdout	Derive	Power-up

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KDF IKEv1 (A3592)	N/A	KAT	CAST	Error message to stdout	Derive	Power-up
KDF IKEv2 (A3592)	N/A	KAT	CAST	Error message to stdout	Derive	Power-up
KTS-IFC (A3592)	2048	KAT	CAST	Error message to stdout	Encrypt/Decrypt	Power-up
RSA SigGen (FIPS186-4) (A3592)	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) (A3592)	2048; PKCS 1.5 & PSS; SHA2-224, SHA2-256, SHA2-384, SHA2-512	KAT	CAST	Error message to stdout	Verify	Power-up
SHA-1 (A3592)	N/A	KAT	CAST	Error message to stdout	Generate	Power-up
SHA2-256 (A3592)	N/A	KAT	CAST	Error message to stdout	Generate	Power-up
SHA2-512 (A3592)	N/A	KAT	CAST	Error message to stdout	Generate	Power-up
KDF SSH (A3592)	N/A	KAT	CAST	Error message to stdout	Derive	Power-up
KDF TLS (A3592)	TLS 1.0/1.1	KAT	CAST	Error message to stdout	Derive	Power-up
TLS v1.2 KDF RFC7627 (A3592)	TLS 1.2 SHA2-256, SHA2-512	KAT	CAST	Error message to stdout	Derive	Power-up
KAS-ECC-SSC Sp800-56Ar3 PCT	Generated keypair	SP 800-56Arev3 assurance checks	PCT	Error message to stdout	SP 800-56Arev3 assurance checks	Keypair generation
KAS-FFC-SSC Sp800-56Ar3 PCT	Generated keypair	SP 800-56Arev3	PCT	Error message to stdout	SP 800-56Arev3 assurance checks	Keypair generation

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
		assurance checks				
ECDSA KeyGen (FIPS186-4) PCT	Generated keypair	Sign/Verify	PCT	Error message to stdout	Sign/Verify	Keypair generated
ECDSA KeyVer (FIPS186-4) PCT	Generated EC keypair	Public key verify	PCT	Error message to stdout	Public key validity	Keypair generated
RSA KeyGen (FIPS186-4) PCT	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 Section 10.2 (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. 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 (A3592)	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 (A3592)	KAT	CAST	During the module loading	Load the module
AES-GCM (A3592)	KAT	CAST	During the module loading	Load the module

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CCM (A3592)	KAT	CAST	During the module loading	Load the module
AES-XTS Testing Revision 2.0 (A3592)	KAT	CAST	During the module loading	Load the module
AES-CMAC (A3592)	KAT	CAST	During module loading	Load the module
Counter DRBG (A3592)	KAT	CAST	During the module loading	Load the module
Hash DRBG (A3592)	KAT	CAST	During the module loading	Load the module
HMAC DRBG (A3592)	KAT	CAST	During the module loading	Load the module
ECDSA SigGen (FIPS186-4) (A3592)	KAT	CAST	During the module loading	Load the module
ECDSA SigVer (FIPS186-4) (A3592)	KAT	CAST	During the module loading	Load the module
HMAC-SHA-1 (A3592)	KAT	CAST	During the module loading	Load the module
HMAC-SHA2-224 (A3592)	KAT	CAST	During the module loading	Load the module
HMAC-SHA2-256 (A3592)	KAT	CAST	During the module loading	Load the module
HMAC-SHA2-384 (A3592)	KAT	CAST	During the module loading	Load the module
HMAC-SHA2-512 (A3592)	KAT	CAST	During the module loading	Load the module
KAS-ECC-SSC Sp800-56Ar3 (A3592)	KAT	CAST	During the module loading	Load the module
KAS-FFC-SSC Sp800-56Ar3 (A3592)	KAT	CAST	During the module loading	Load the module
KDF SP800-108 (A3592)	KAT	CAST	During the module loading	Load the module
KDF IKEv1 (A3592)	KAT	CAST	During the module loading	Load the module
KDF IKEv2 (A3592)	KAT	CAST	During the module loading	Load the module
KTS-IFC (A3592)	KAT	CAST	During the module loading	Load the module
RSA SigGen (FIPS186-4) (A3592)	KAT	CAST	During the module loading	Load the module

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
RSA SigVer (FIPS186-4) (A3592)	KAT	CAST	During the module loading	Load the module
SHA-1 (A3592)	KAT	CAST	During the module loading	Load the module
SHA2-256 (A3592)	KAT	CAST	During the module loading	Load the module
SHA2-512 (A3592)	KAT	CAST	During the module loading	Load the module
KDF SSH (A3592)	KAT	CAST	During the module loading	Load the module
KDF TLS (A3592)	KAT	CAST	During the module loading	Load the module
TLS v1.2 KDF RFC7627 (A3592)	KAT	CAST	During the module loading	Load the module
KAS-ECC-SSC Sp800-56Ar3 PCT	SP 800-56Arev3 assurance checks	PCT	N/A	N/A
KAS-FFC-SSC Sp800-56Ar3 PCT	SP 800-56Arev3 assurance checks	PCT	N/A	N/A
ECDSA KeyGen (FIPS186-4) PCT	Sign/Verify	PCT	N/A	N/A
ECDSA KeyVer (FIPS186-4) PCT	Public key verify	PCT	N/A	N/A
RSA KeyGen (FIPS186-4) PCT	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

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 CloudEOS product, in the CloudEOS image accessible through the Arista software downloads website. The CloudEOS product includes the CloudEOS operating system, virtual machine, 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 SHA256 hash sum. The SHA256 hash should be 8b92b97d92571963b66649d0bb3ca62fba77100a316757e9487ad2091eddcc18. 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 CloudEOS image, the SHA-256 hash of the image is also made available. When an authorized operator downloads the CloudEOS image, they can also download the hash file and compare the SHA-256 hash of the CloudEOS image to the one listed in the file to make sure that the downloaded image is correct. Then they can install the CloudEOS image onto the virtual machine.

Upon completion of installation, the user can confirm that the correct module has been installed by running the "show version" service should display the module base name and version number, "Crypto Module: Arista Crypto Module v3.0". Correct operation of the module can be verified by running the on-demand self-test service as specified in Section 5 by calling "openssl --fips" from bash.

11.2 Administrator Guidance

None

11.3 Non-Administrator Guidance

None

11.4 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 and provide a new replacement.

12 Mitigation of Other Attacks

N/A for this module.