Arista Networks Inc.

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

Version: 3.0

Non-Proprietary FIPS 140-3 Security Policy

Document Version: v1.4

Date: July 6, 2024

Table of Contents

1.0 - General Information **4**
1.1 Overview 4
1.2 Security Levels 4
2.0 Cryptographic Module Specification **4**
2.1 Description 4
2.2 Version Information 5
2.3 Operating Environments 6
2.4 Excluded Components 7
2.5 Modes of Operation 7
2.6 Approved Algorithms 7
2.7 Algorithm Specific Information 12
2.8 RBG and Entropy 14
2.9 Key Generation 14
2.10 Key Establishment 14
2.11 Industry Protocols 14
2.12 Design and Rules 18
2.13 Initialization 18
3.0 - Cryptographic Module Interfaces **18**
3.1 Ports and Interfaces 18
4.0 - Roles, Services and Authentication **18**
4.1 Authentication Methods 18
4.2 Roles 19
4.3 Approved Services 21
4.4 Non-Approved Services 24
4.5 External Software/Firmware Loaded – N/A 24
5.0 - Software/Firmware security **24**
5.1 Integrity Techniques 24
5.2 Initiate on Demand 25
6.0 Operational environment **25**
6.1 Operational Environment Type and Requirements 25
6.2 Configuration Settings and Restrictions 25
7.0 - Physical security – N/A **25**
8.0 - Non-invasive security – N/A **25**
9.0 Sensitive Security Parameters Management **25**
9.1 Storage Areas 25
9.2 SSP Input-Output Methods 25
9.3 SSP Zeroisation Methods 26
9.4 SSPs 26
10. Self-tests **27**

10.1 Pre-Operational Self-Tests 27
10.2 Conditional Self-Tests 28
10.3 Periodic Self-Tests 31
10.4 Error States 31
11. Life-cycle Assurance 31
11.1 Startup Procedures 31
11.2 Administrator Guidance 32
11.3 Non-Administrator Guidance 32
11.4 Maintenance Requirements – N/A 32
11.5 End of Life 32
12.0 Mitigation of other attacks – N/A 32
13.0 References and Definitions 32

1.0 - General Information

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

ISO/IEC 24759 Section 6.	FIPS 140-3 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

Table 1 – Security Levels

Overall security level 1.

2.0 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: None

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)

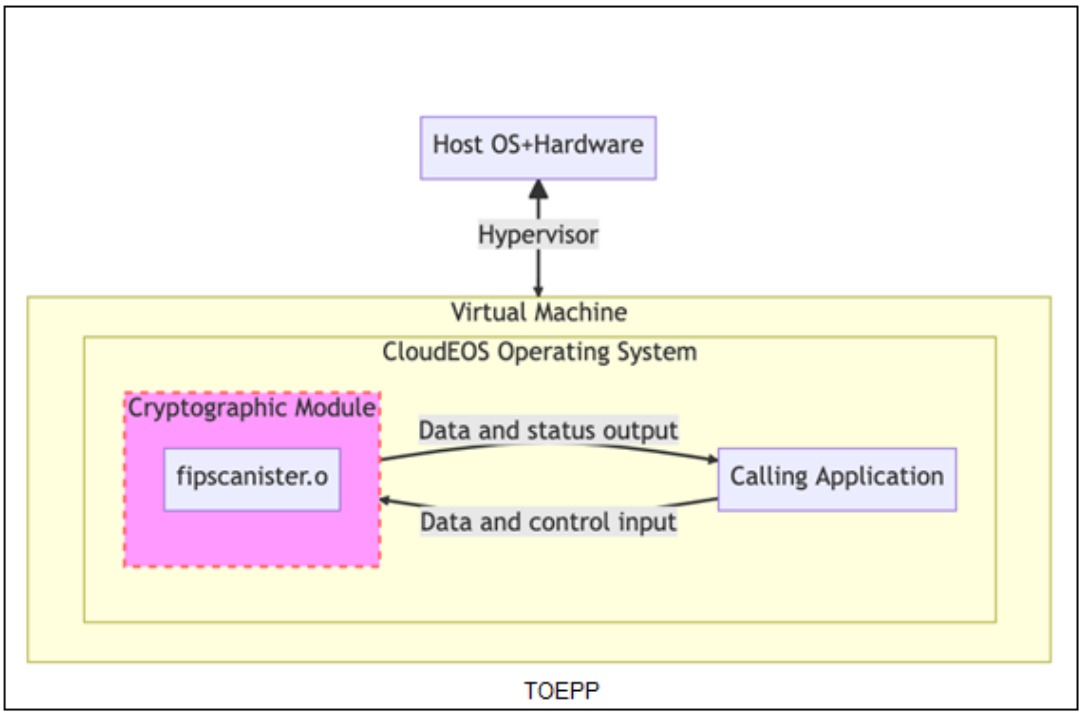

Figure 1 – Block diagram depicting the cryptographic boundary (in pink) and data flow between the module interfaces and operator. The boundary also includes the instantiation of the cryptographic module in memory.

The TOEPP is the physical perimeter of the hardware platform listed in Table 2 – Tested Operational Environment.

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 Version Information

Type	Versions
Software	Name: Arista Crypto Module v3.0 [Software, Software IPsec] Version: 3.0

Table A – Version Information

The module does not contain any hardware or firmware.

2.3 Operating Environments

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. The module does not support concurrent operators.

Hardware Operating Environments – N/A

Software, Firmware, Hybrid Testing Operating Environments:

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	Processor	PAA/Acceleration
1	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
2	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

Table 2 – Tested Operational Environments

Code Sets:

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

Vendor Affirmed Operating Environments:

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
1	CloudEOS	Any general-purpose computer (GPC)
2	Any compatible OS	Any general-purpose computer (GPC)

Table 3 - Vendor Affirmed Operational Environments

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: The 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.

2.4 Excluded Components

There are no excluded components for the module.

2.5 Modes of Operation

Modes List and Description:

Name	Description	Approved Mode	Status Indicator
Approved Mode	Single Approved Mode – selected by calling the FIPS_mode_set(1) function.	Yes	The status indicator is a return value 1 from the FIPS_mode() function.
Non-Approved Mode	Selected by default in CloudEOS	No	The status indicator is a return value 0 from the FIPS_mode() function.

Table B - Modes of Operation

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 indicators:

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”.

2.6 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.

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A3592	AES-CBC	AES	128, 192, 256	Encrypt, Decrypt
A3592	AES-CCM	AES	128, 192, 256	Encrypt, Decrypt
A3592	AES-CFB1	AES	128, 192, 256	Encrypt, Decrypt
A3592	AES-CFB128	AES	128, 192, 256	Encrypt, Decrypt
A3592	AES-CFB8	AES	128, 192, 256	Encrypt, Decrypt
A3592	AES-CMAC	AES	128, 192, 256	Message Authentication
A3592	AES-CTR	AES	128, 192, 256	Encrypt, Decrypt
A3592	AES-ECB	AES	128, 192, 256	Encrypt, Decrypt
A3592	AES-GCM	AES	128, 192, 256	Authenticated Encrypt, Authenticated Decrypt, Message Authentication
A3592	AES-XTS Testing Revision 2.0	AES	128, 256	Confidentiality on storage devices only [XTS-AES is compliant to IG C.I by checking for Key_1 ≠ Key_2.]
A3592	Counter DRBG	Counter DRBG	128, 192, 256	Deterministic Random Bit Generation [Module defaults to Counter DRBG with 256-bit security strength]
A3592	ECDSA KeyGen (FIPS186-4)	Secret Generation Mode: Testing Candidates	P-256, P-384, P-521	KeyGen
A3592	ECDSA KeyVer (FIPS186-4)	ECDSA KeyVer	P-256, P-384, P-521	KeyVer
A3592	ECDSA SigGen (FIPS186-4)	ECDSA SigGen	Curve: P-256, P-384, P-521; Hash Algorithm: SHA2-224, SHA2-256, SHA2-384, SHA2-512	SigGen
A3592	ECDSA SigVer (FIPS186-4)	ECDSA SigVer	Curve: P-256, P-384, P-521; Hash Algorithm: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	SigVer
A3592	HMAC DRBG	HMAC DRBG	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	Deterministic Random Bit Generation
A3592	HMAC-SHA-1	HMAC	Key: 256-2048 Increment 8; MAC: 80-160 Increment 8	Message Authentication, password obfuscation
A3592	HMAC-SHA2-224	HMAC	Key: 256-2048 Increment 8; MAC: 112-224 Increment 16	Message Authentication
A3592	HMAC-SHA2-256	HMAC	Key: 256-2048 Increment 8; MAC: 128-256 Increment 64	Message Authentication, KDF primitive, integrity test
A3592	HMAC-SHA2-384	HMAC	Key: 256-2048 Increment 8; MAC: 192-384 Increment 64	Message Authentication, KDF primitive

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A3592	HMAC-SHA2-512	HMAC	Key: 256-2048 Increment 8; MAC: 256-512 Increment 64	Message Authentication, KDF primitive
A3592	Hash DRBG	Hash DRBG	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	Deterministic Random Bit Generation
A3592	KAS-ECC-SSC Sp800-56Ar3	KAS	ephemeralUnified: P-256, P-384, P-521	Key Agreement [Relies on calling application to feed shared secret into KDF
A3592	KAS-FFC-SSC Sp800-56Ar3	KAS	dhEphem: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	Key Agreement [Relies on calling application to feed shared secret into KDF]
A3592	CVL KDF IKEv1	KDF IKEv1	Hash Algorithm: SHA-1, SHA2-256, SHA2-384, SHA2-512	Key Derivation for IKEv1
A3592	CVL KDF IKEv2	KDF IKEv2	Hash Algorithm: SHA-1, SHA2-256, SHA2-384, SHA2-512	Key Derivation for IKEv2
A3592	KDF SP800-108	KDF SP800-108	KDF Mode: Counter; MAC Mode: CMAC-AES128, CMAC-AES256	Key Derivation
A3592	CVL KDF SSH	KDF SSH	Hash Algorithm: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	Key Derivation for SSHv2
A3592	CVL KDF TLS	KDF TLS	TLS Version: v1.0/1.1	Key Derivation for TLS
A3592	KTS-IFC	KTS	Modulo: 2048, 3072, 4096; KTS-OAEP-basic	Key Transport
A3592	RSA KeyGen (FIPS186-4)	RSA KeyGen	Key Generation Mode: B.3.3; Modulo: 2048, 3072, 4096	KeyGen
A3592	RSA SigGen (FIPS186-4)	RSA SigGen	Modulo 2048, 3072, 4096; ANSI X9.31 (SHA2-256, SHA2-384, SHA2-512), PKCS 1.5 (SHA2-224, SHA2-256, SHA2-384, SHA2-512), PKCSPSS (SHA2-224, SHA2-256, SHA2-384, SHA2-512)	SigGen
A3592	RSA SigVer (FIPS186-4)	RSA SigVer	Modulo 1024, 2048, 3072, 4096; ANSI X9.31 (SHA-1 SHA2-256, SHA2-384, SHA2-512), PKCS 1.5 (SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512), PKCSPSS (SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512)	SigVer
A3592	SHA-1	SHS	Message Length: 0-65536 Increment 8	Message Digest Generation
A3592	SHA2-224	SHS	Message Length: 0-65536 Increment 8	Message Digest Generation

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A3592	SHA2-256	SHS	Message Length: 0-65536 Increment 8	Message Digest Generation
A3592	SHA2-384	SHS	Message Length: 0-65536 Increment 8	Message Digest Generation
A3592	SHA2-512	SHS	Message Length: 0-65536 Increment 8	Message Digest Generation
A3592	CVL TLS v1.2 KDF RFC7627	TLS v1.2 KDF RFC7627	Hash Algorithm: SHA2-256, SHA2-384, SHA2-512	Key Derivation for TLS

Table 5 - Approved Algorithms

Note: IG D.R states for modules submitted after May 16, 2023 it is non-approved to use of SHA2-224 or SHA2-384 within Hash DRBG or HMAC DRBG.

Vendor Affirmed Approved Algorithms

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

Algorithm	Caveat	Use or Function
CKG [IG D.H]		Cryptographic key generation per SP 800-133rev2 and IG D.I * Generation of asymmetric keys for signature generation per [133] section 5.1. * Generation of asymmetric keys for key establishment per [133] section 5.2. * Symmetric key derivation for industry standard protocols from a key agreement shared secret per [133] section 6.2.1. * Symmetric key derivation from existing key per [133] section 6.2.2.

Table 6 – Vendor Affirmed Approved Algorithms

Non-Approved Algorithms Allowed in the Approved Mode of Operation

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 Algorithms 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 Table 15.

Algorithm	Caveat	Use or Function

MD5	Allowed per IG 2.4.A	Message digest used in TLS 1.0/1.1 KDF only

Table 8 – Non-Approved Algorithms Allowed in the Approved Mode of Operation with No Security Claimed

Security Function Implementation (SFI)

Name	Type	Description	SF Properties [O]	Algorithms/CAVP Cert
KAS-ECC	KAS	SP 800-56Arev3. KAS_ECC_SSC per IG D.F Scenario 2, path (2). No key confirmation, key derivation per IG 2.4.B. SP 800-135. KDFs (TLS 1.0/1.1, 1.2, SSHv2, IKE v1, IKE v2)	P-256, P-384, P-521 curves providing 128, 192, or 256 bits of encryption strength	KAS-ECC-SSC Sp800-56Ar3/A3592 KDF IKEv1/A3592 KDF IKEv2/A3592 KDF SSH/A3592 KDF TLS/A3592 TLS v1.2 KDF RFC7627/A3592
KAS-FFC	KAS	SP 800-56Arev3. KAS_FFC_SSC per IG D.F Scenario 2, path (2). No key confirmation, key derivation per IG 2.4.B. SP 800-135. KDFs (TLS 1.0/1.1, 1.2, SSHv2, IKE v1, IKE v2)	2048, 3072, 4096, 6144, and 8192-bit moduli providing 112, 128, 152, 176, or 200 bits of encryption strength	KAS-FFC-SSC Sp800-56Ar3/A3592 KDF IKEv1/A3592 KDF IKEv2/A3592 KDF SSH/A3592 KDF TLS/A3592 TLS v1.2 KDF RFC7627/A3592
KTS-IFC	KTS	SP 800-56Brev2. KTS-IFC (key encapsulation and un-encapsulation) per IG D.G.	2048, 3072, and 4096-bit moduli providing 112, 128, or 152 bits of encryption strength	KTS-IFC KTS-OAEP-basic/A3592
TLS-KTS	KTS	SP 800-38D and SP 800-38F. KTS (key wrapping and unwrapping) per IG D.G, Additional Comment 8.	128 and 256-bit keys providing 128 or 256 bits of encryption strength	AES-GCM/A3592 AES-CCM/A3592 AES-CBC/A3592 HMAC/A3592
SSHv2-KTS	KTS	SP 800-38D and SP 800-38F. KTS (key wrapping and unwrapping) per IG D.G, Additional Comment 8.	128, 192, 256-bit keys providing 128, 192, or 256 bits of encryption strength	AES-GCM/A3592 AES-CBC/A3592 AES-CTR/A3492 HMAC/A3592
IPsec-KTS	KTS	SP 800-38D and SP 800-38F. KTS (key wrapping and unwrapping) per IG D.G, Additional Comment 8.	128, 192, 256-bit keys providing 128, 192, or 256 bits of encryption strength	AES-GCM/A3592 AES-CCM/A3592 AES-CBC/A3592 HMAC/A3592

Table 9 – Security Function Implementation (SFI)

Entropy Certificates

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

Non-Approved Algorithms Not Allowed In the approved Mode of Operation

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

Algorithm/Function	Use/Function
DSA (disallowed)	Digital Signature and Asymmetric Key Generation; PQG Gen, Key Pair Gen, Sig Gen
RSA (disallowed)	Key Encryption, Decryption using PKCS#1 v1.5
Hash DRBG w/ SHA2-224 or SHA2-384 (disallowed)	Random Bit Generation
HMAC DRBG w/ SHA2-224 or SHA2-384 (disallowed)	Random Bit Generation
AES/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 11 - Non-Approved Algorithms Not Allowed In the approved Mode of Operation

2.7 Algorithm Specific Information

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

For TLS 1.2, the module offers the GCM implementation via the functions aes_gcm_tls_cipher, which calls CRYPTO_gcm128_encrypt_ctr32, and uses the context of Scenario 1 of IG C.H. The module is compliant with SP800-52rev2 and the mechanism

for IV generation is compliant with RFC5288. The module supports acceptable AES-GCM ciphersuites from Section 3.3.1 of SP800-52rev2.

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 consuming 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

For SSH, the module offers the GCM implementation via the functions CRYPTO_gcm128_encrypt_ctr32, and uses the context of Scenario 1 of IG C.H. The module is compliant with RFCs 4252, 4253, and 5647.

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

Scenario 1, IPsec-v3

For IPsec, the module offers the GCM implementation via the functions CRYPTO_gcm128_encrypt_ctr32, and uses the context of Scenario 1 of IG C.H. The module is compliant with RFCs 4106 and 5282. The module uses RFC 7296 compliant IKEv2 to establish the shared secret SKEYSEED from which the AES-GCM encryption keys are derived.

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 consuming 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 interfaces 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. Per Section 9, this 96-bit IV contains 96 bits of entropy.

XTS-AES Key Generation

The module checks for Key_1 ≠ Key_2 before using the keys in the XTS-AES algorithm in compliance with IG C.I.

2.8 RBG and Entropy

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 (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.

2.11 Industry Protocols

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

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 C- 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-NIST P521, ECDH-SHA2-NIST P384, ECDH-SHA2-NIST P256, DIFFIE-HELLMAN GROUP14-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			
	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			

Protocol	Key Exchange	Server/ Host Auth	Cipher	Integrity
	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			

Protocol	Key Exchange	Server/ Host Auth	Cipher	Integrity
	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 D - Security Relevant Protocols Used in Approved Mode

2.12 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 “2.5 Modes of Operation”.

2.13 Initialization

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 “2.5 Modes of Operation”:

3.0 - 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	Data that passes over the interface
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 12 – Ports and Interfaces

The module does not support Control Output.

4.0 - Roles, Services and Authentication

4.1 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 passwords. This sole role is implicitly assumed by the operator of the module when performing a service after authentication.

Table 13 provides a mapping of services to the roles that can utilize them, in this case the sole role of the module, and the service inputs and outputs.

Role	Service	Input	Output
CO	Authenticated Decryption	Ciphertext, authentication tag, key, IV	Plaintext
CO	Authenticated Encryption	Plaintext, key, IV	Ciphertext, authentication tag
CO	Decryption	Ciphertext, key	Plaintext
CO	Encryption	Plaintext, key	Ciphertext
CO	Key Derivation (TLS)	PRF algorithm, TLS master secret	Derived Keys
CO	Key Derivation (SSH)	PRF algorithm, SSH shared secret	Derived Keys
CO	Key Derivation (IKE)	PRF algorithm, IKE shared secret	Derived Keys
CO	Key Derivation (SP 800-108r1)	Shared secret, key size	Derived Keys
CO	Key Encapsulation	RSA keypair, keying material to encapsulate	Encapsulated key
CO	Key Generation	Algorithm, key size	Key Pair

Role	Service	Input	Output
CO	Key Un-encapsulation	RSA keypair, keying material to un-encapsulate	Un-encapsulated key
CO	Key Verification	Key to verify	Return codes and log messages
CO	Initialize	Crypto Officer Password	None
CO	Message Authentication Generation	Message, Algorithm, key	Message Authentication code
CO	Message Digest	Message	Digest of the message
CO	On-Demand Integrity Test	None	Result of test (pass/fail)
CO	On-Demand self-test	None	Result of self-test (pass/fail)
CO	Random number generation	Size	Random bytes
CO	Shared secret computation	EC Curve or DH parameters, V's public key	Shared secret
CO	Show Status	None	Return code of 1 indicates approved mode enabled, 0 is disabled
CO	Show Version	None	String indicating the module version and name
CO	Signature Generation	Message, hash algorithm, private key	Signature
CO	Signature Verification	Message, Signature, hash algorithm, public key	Verification result
CO	Zeroise	Context containing SSPs	None

Table 13 – Roles, Services, Input, and Output

Table 14 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.

Role	Authentication Method	Authentication Strength
CO (Crypto Officer)	Password	95^16 (module enforces 16 character minimum password

		length); chance of guessing in one minute 1 in 9.03*10^18

Table 14 – Roles and Authentication

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.

Service	Description	Approved Security Functions	Keys/SSPs	Roles	Access rights to Keys/SSPs	Indicator
Authenticated Decryption	Authenticated Decryption	AES-GCM, AES-CCM	AES key	CO	W, E	Return code 1, log message indicating approval
Authenticated Encryption	Authenticated Encryption	AES-GCM, AES-CCM	AES key	CO	W, E	Return code 1, log message indicating approval
Decryption	Decryption	AES CBC, CTR, ECB, CFB1, CFB128, CFB8, XTS	AES key	CO	W, E	Return code 1, log message indicating approval
Encryption	Encryption	AES CBC, CTR, ECB, CFB1, CFB128, CFB8, XTS	AES key	CO	W, E	Return code 1, log message indicating approval
Key Derivation (TLS)	Deriving TLS keys	KDF TLS 1.0/1/1/1.2	TLS pre_master_secret; TLS master secret; TLS derived keys	CO	TLS pre_master_secret - W, E; TLS master secret - G, E; TLS derived keys G, R	Return code 1, log message indicating approval
Key Derivation (SSH)	Deriving SSH keys	KDF SSH v2	SSH shared secret; SSH derived keys	CO	SSH shared secret - W, E; SSH derived key - G, R	Return code 1, log message indicating approval
Key Derivation (IKE)	Deriving IKE keys	KDF IKE v1, v2	IKE shared secret; IKE derived key	CO	IKE shared secret - W, E; IKE derived key - G, R	Return code 1, log message indicating approval
Key Derivation (SP 800-108r1)	Deriving keys	KDF SP800-108	Shared secret; 800-108 derived key	CO	Shared secret - W, E; 800-108 derived key - G, R	Return code 1, log message indicating approval
Key Encapsulation	Key Encapsulation	KTS-IFC	RSA key pair, keying material	CO	RSA key pair - W, E; keying material	Return code 1, log message indicating

Service	Description	Approved Security Functions	Keys/SSPs	Roles	Access rights to Keys/SSPs	Indicator
	per SP 800-56Br2				- W, R	approval
Key Generation	Generating Key pair	ECDSA, RSA, DRBG	ECDSA key pair; RSA key pair	CO	ECDSA key pair; RSA key pair - G, R; DRBG Seed, V, C, Key - W, E	Return code 1, log message indicating approval
Key Un-encapsulation	Key Un-encapsulation per SP 800-56Br2	KTS-IFC	RSA key pair, keying material	CO	RSA key pair - W, E; keying material - W, R	Return code 1, log message indicating approval
Key Verification	Verifying the public key	ECDSA	ECDSA public key	CO	W, E	Return code 1, log message indicating approval
Initialize	Initialize FIPS password using FIPS_set_password	HMAC SHA-1	Crypto Officer Password, Hashed Password	CO	Crypto Officer Password - W, E; Hashed Password - E	Return code 1
Message Authentication Generation	MAC computation	AES CMAC, HMAC	AES key; HMAC key	CO	W, E	Return code 1, log message indicating approval
Message Digest	Generating message digest	SHS	N/A	CO	N/A	Return code 1, log message indicating approval
On-Demand Integrity Test	Initiate integrity test on-demand through FIPS_check_incore_fingerprint	HMAC SHA2-256	N/A (keys for self-tests are not SSPs)	CO	N/A	Return code 1
On-Demand self-test	Initiate pre-operational and conditional CAST self-tests through FIPS_selftest	AES, CMAC, DRBG, ECDSA, HMAC, KAS-ECC-SSC, KAS-FFC-SSC, KDF, KTS, IKE KDF, RSA, SHS, TLS KDF, SSH KDF	N/A (keys for self-tests are not SSPs)	CO	N/A	Return code 1
Random number generation	Generating random numbers	DRBG	DRBG Entropy Input; DRBG Seed, V, C, Key	CO	DRBG Entropy Input - W, E; DRBG Seed, V, C, Key - G, E	Return code 1, log message indicating approval
Shared secret computation	Calculating Shared secret	KAS-ECC-SSC, KAS-FFC-SSC,	DH key pair; ECDH key pair; DRBG	CO	DH key pair - G, E, Z; ECDH key	Return code 1, log message indicating

Service	Description	Approved Security Functions	Keys/SSPs	Roles	Access rights to Keys/SSPs	Indicator
		DRBG	Seed, V Key; Shared secret		pair G, E, Z; DRBG Seed, V, C, Key - W, E; Shared secret - G, R	approval
Show Status	Show status of the module state using FIPS_mode	N/A	N/A	CO	N/A	N/A
Show Version	Show the version of the module using FIPS_module_version_text	N/A	N/A	CO	N/A	N/A
Signature Generation	Generating signature	ECDSA, RSA, SHS	ECDSA key pair; RSA key pair	CO	W, E	Return code 1, log message indicating approval
Signature Verification	Verifying signature	ECDSA, RSA, SHS	ECDSA key pair; RSA key pair	CO	W, E	Return code 1, log message indicating approval
Zeroise	Zeroise SSP in volatile memory	N/A	Context containing SSPs	CO	SSPs – Z	N/A

Table 15 – 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 “2.5 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 following table lists the non-approved services that utilize non-approved security functions.

Name	Description	Algorithms Accessed	Role	Indicator
Decryption	Decryption	Blowfish, Camillia, CAST5, DES, DES-X, IDEA, RC2, RC5, SEED, Triple-DES listed in Table 11	CO	Return code 0, absence of approved log message
Encryption	Encryption	Blowfish, Camillia, CAST5, DES, DES-X, IDEA, RC2, RC5, SEED, Triple-DES listed in Table 11	CO	Return code 0, absence of approved log message
Key Wrapping	Encrypting/Decrypting key	AES/Triple-DES KW, RSA PKCS #1 v1.5 listed in Table 11	CO	Return code 0, absence of approved log message
Message Digest	Hash computation	MD4, MD5 outside TLS 1.0 usage, RIPEMD-160, Whirlpool, Triple-DES MAC, HMAC-MD5 listed in Table 11	CO	Return code 0, absence of approved log message

Table E - Non-approved services

4.5 External Software/Firmware Loaded – N/A

5.0 - 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”.

6.0 Operational environment

6.1 Operational Environment Type and Requirements

Type of Operating Environment: Modifiable

6.2 Configuration Settings and Restrictions

The module should be installed as stated in section 11.

7.0 - Physical security – N/A

8.0 - Non-invasive security – N/A

9.0 Sensitive Security Parameters Management

9.1 Storage Areas

Name	Description	Persistence Type
RAM	System Memory	Dynamic

Table F – 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

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 Zeroisation 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

Key/SSP/Name/ Type	Strength	Security Function Cert Number	Generation	Import/Export	Establishment	Storage	Zeroisation	Use & related keys
800-108 derived key	128, 192, 256	A3592	SP 800-108 KDF	N/A / Plaintext	N/A	Ephemeral in RAM	OPENSSL_cleanse	Derived for output to calling application. Used with Shared Secret
AES Key	128, 192, 256	A3592	External or KDF	Plaintext / Plaintext	KAS-ECC or KAS-FFC	Ephemeral in RAM	OPENSSL_cleanse	Authenticated Encryption, Authenticated Decryption, Encryption, Decryption, Message Authentication Generation. Used with Shared Secret
Crypto Officer Password	N/A	N/A	N/A	Plaintext / N/A	N/A	Ephemeral in RAM	Automatic at end of service call	Crypto Officer authentication. Used with Hashed Password
Hashed Password	N/A	A3592	HMAC SHA-1 of Crypto Officer Password	N/A	N/A	Ephemeral in RAM	Restart module	Crypto Officer authentication. Used with Crypto Officer Password
DH key pair	112 – 200	A3592	Internal per SP 800-56Arev3	N/A / Public key in plaintext	N/A	Ephemeral in RAM	DH_free	Key agreement. Used with: DRBG Seed, V, C, and Key, Shared Secret
DRBG Entropy Input	384	A3592	External	Plaintext / N/A	N/A	Ephemeral in RAM	FIPS_DRBG_free	Random number generation. Used with DRBG Seed, V, C, and Key
DRBG Seed	256	A3592	From DRBG entropy input; within SP 800-90A Hash_DRBG, HMAC_DRBG, and CTR_DRBG DRBGs	N/A / N/A	N/A	Ephemeral in RAM	FIPS_DRBG_free	Random number generation. Used with DRBG Entropy Input and generated keys
DRBG V	256	A3592	From DRBG entropy input; within SP 800-90A Hash_DRBG, HMAC_DRBG, and CTR_DRBG DRBGs	N/A / N/A	N/A	Ephemeral in RAM	FIPS_DRBG_free	Random number generation. Used with DRBG Entropy Input and generated keys
DRBG C	256	A3592	From DRBG entropy input; within SP 800-90A Hash_DRBG	N/A / N/A	N/A	Ephemeral in RAM	FIPS_DRBG_free	Random number generation. Used with DRBG Entropy Input and generated keys
DRBG Key	256	A3592	From DRBG entropy input; within SP 800-90A HMAC_DRB, and CTR_DRBG DRBGs	N/A / N/A	N/A	Ephemeral in RAM	FIPS_DRBG_free	Random number generation. Used with DRBG Entropy Input and generated keys
ECDH key pair	128-256	A3592	Internal per SP 800-56Arev3	N/A / Public key in plaintext	N/A	Ephemeral in RAM	EC_GROUP_free, EC_POINT_free, EC_KEY_free	Key agreement. Used with: DRBG Seed, V, C, and Key, Shared Secret
ECDSA key pair	128, 192, 256	A3592	External or per FIPS 186-4	Plaintext / Plaintext	N/A	Ephemeral in RAM	EC_GROUP_free, EC_POINT_free, EC_KEY_free	Signature generation and verification. Used with DRBG Seed, V, C, and Key
HMAC key	112 or greater	A3592	External or KDF	Plaintext / Plaintext	KAS-ECC or KAS-FFC	Ephemeral in RAM	HMAC_CTX_cleanup	Message Authentication. Used with Shared secret
IKE shared secret	112 -256	A3592	N/A	Plaintext / Plaintext	KAS-ECC-SSC or KAS-FFC-SSC	Ephemeral in RAM	OpenSSL_cleanse	KE key agreement. Used with IKE derived key, DH key pair, ECDH key pair
IKE Derived key/AES &	112 or greater	A3592	KDF IKE	N/A / Plaintext	N/A	Ephemeral in RAM	OpenSSL_cleanse	IKE key agreement Used with IKE shared secret

Key/SSP/Name/ Type	Strength	Security Function Cert Number	Generation	Import/Export	Establishment	Storage	Zeroisation	Use & related keys
HMAC								
Keying material	112 or greater	A3592	External	Plaintext or Encrypted / Encrypted or Plaintext	KTS-IFC	Ephemeral in RAM	OpenSSL_cleanse	KTS-IFC keying material to be encapsulated or un-encapsulated by RSA-OAEP_basic. Used with RSA key pair
RSA key pair	112, 128, 152	A3592	External or per FIPS 186-4	Plaintext / Plaintext	N/A	Ephemeral in RAM	RSA_free	Signature generation and verification or KTS-IFC. Used with DRBG Seed, V, C, and Key; and keying material to encapsulate/un-encapsulate
Shared secret	112 or greater	A3592	N/A	Plaintext / Plaintext	KAS-ECC-SSC or KAS-FFC-SSC	Ephemeral in RAM	OpenSSL_cleanse	For key agreement. Used with DH key pair, ECDH key pair
SSH shared secret	112 or greater	A3592	N/A	Plaintext / Plaintext	KAS-ECC-SSC or KAS-FFC-SSC	Ephemeral in RAM	OpenSSL_cleanse	SSH key agreement. Used with SSH Derived key, DH key pair, ECDH key pair
SSH Derived key/AES & HMAC	112 or greater	A3592	KDF SSH	N/A / Plaintext	N/A	Ephemeral in RAM	OpenSSL_cleanse	SSH key agreement Used with SSH shared secret
TLS Derived key/AES & HMAC	112 or greater	A3592	KDF TLS 1.0/1.1, 1.2 RFC7627	N/A / Plaintext	N/A	Ephemeral in RAM	OpenSSL_cleanse	TLS key agreement Used with TLD master secret, TLS pre-master secret
TLS master secret	112-256	A3592	From TLS pre-master secret	Plaintext / Plaintext	KAS-ECC-SSC or KAS-FFC-SSC	Ephemeral in RAM	OpenSSL_cleanse	TLS key agreement Used with TLS pre-master secret, TLS Derived key
TLS pre-master secret	112 - 256	A3592	N/A	Plaintext / Plaintext	KAS-ECC-SSC or KAS-FFC-SSC	Ephemeral in RAM	OpenSSL_cleanse	TLS key agreement Used with TLS master secret, TLS Derived key

Table 20 – SSPs

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

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.

Algorithm Tested	Implementation	Test Properties	Test Method	Type	Indicator	Test Details
HMAC-SHA2-256		128-bit hardcoded key	Compare Hash Results	SW Integrity	Stdout, log message	Single encompassing message authentication code

Table G – Pre-Operational Test Methods

10.2 Conditional Self-Tests

Algorithm Tested	Implementation	Test Properties	Test Method	Type	Indicator	Test Details	Conditions
AES	AES-ECB	128	KAT	CAST	Stdout, log message	Encrypt/ Decrypt	Power-up
AES	AES-GCM	256	KAT	CAST	Stdout, log message	Encrypt/ Decrypt	Power-up
AES	AES-CCM	192	KAT	CAST	Stdout, log message	Encrypt/ Decrypt	Power-up
AES	AES-XTS	128, 256	KAT	CAST	Stdout, log message	Encrypt/ Decrypt	Power-up
CMAC	CMAC-AES	128, 192, 256	KAT	CAST	Stdout, log message	Generate/ Verify	Power-up
DRBG	Counter DRBG	Chained instantiate, reseed, generate	KAT	CAST	Stdout, log message	SP 800-90A section 11.3 health tests	Power-up
DRBG	Hash DRBG	Chained instantiate, reseed, generate	KAT	CAST	Stdout, log message	SP 800-90A section 11.3 health tests	Power-up
DRBG	HMAC DRBG	Chained instantiate, reseed, generate	KAT	CAST	Stdout, log message	SP 800-90A section 11.3 health tests	Power-up
ECDSA		P-224, P-384	KAT	CAST	Stdout, log	Sign/ Verify	Power-up

Algorithm Tested	Implementation	Test Properties	Test Method	Type	Indicator	Test Details	Conditions
					message		
HMAC		HMAC SHA2-224	KAT	CAST	Stdout, log message	Generate	Power-up
HMAC		HMAC SHA2-256	KAT	CAST	Stdout, log message	Generate	Power-up
HMAC		HMAC SHA2-512	KAT	CAST	Stdout, log message	Generate	Power-up
IKE KDF			KAT	CAST	Stdout, log message	Derive	Power-up
KAS-ECC-SSC		P-224, P256	KAT	CAST	Stdout, log message	Shared secret “z” computation	Power-up
KAS-FFC-SSC		2048	KAT	CAST	Stdout, log message	Shared secret “z” computation	Power-up
KBKDF	Counter mode		KAT	CAST	Stdout, log message	Derive	Power-up
RSA		2048; PKCS 1.5 & PSS; SHA2-224, SHA2-256, SHA2-384, SHA2-512	KAT	CAST	Stdout, log message	Sign/ Verify	Power-up
RSA	KTS-IFC	2048	KAT	CAST	Stdout, log message	Encrypt/ Decrypt	Power-up
SHS	SHA-1		KAT	CAST	Stdout, log message	Generate	Power-up
SHS	SHA2-224		KAT	CAST	Stdout, log message	Generate	Power-up
SHS	SHA2-256		KAT	CAST	Stdout, log message	Generate	Power-up

Algorithm Tested	Implementation	Test Properties	Test Method	Type	Indicator	Test Details	Conditions
SHS	SHA2-384		KAT	CAST	Stdout, log message	Generate	Power-up
SHS	SHA2-512		KAT	CAST	Stdout, log message	Generate	Power-up
SSH KDF			KAT	CAST	Stdout, log message	Derive	Power-up
TLS KDF			KAT	CAST	Stdout, log message	Derive	Power-up
ECDSA			PCT	CPCT	N/A	Sign/ Verify	Generate Key Pair
KAS-ECC-SSC			PCT	CPCT	N/A	SP 800-56Arev3 assurance checks	Generate Key Pair
KAS-FFC-SSC			PCT	CPCT	N/A	SP 800-56Arev3 assurance checks	Generate Key Pair
RSA			PCT	CPCT	N/A	Sign/ Verify	Generate Key Pair

Table H – 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 H). In addition, SP 800-56a Rev3 conditional tests are run when ephemeral keypairs are created for key agreement.

10.3 Periodic Self-Tests

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 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 and an error is returned from the API.
PreOp Error		Pre-operational test failure	The module is aborted – restart module	Error message is output on stderr.

Table I - 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 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 Maintenance Requirements – N/A

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 and provide a new replacement.

12.0 Mitigation of other attacks – N/A

13.0 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

Abbreviation	Full Specification Name
[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.*

Abbreviation	Full Specification Name
[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
[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), July1 2013*
[186-2]	*Federal Information Processing Standards Publication 186-2, Digital Signature Standard (DSS), January 2000*
[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*

Table J - References

Acronym	Definition
CO	Cryptographic Officer role
CloudEOS	Name of the Arista operating system
VA	Vendor Affirmed cryptographic algorithms are Approved algorithms for which no CAVP tests are available yet. The vendor performs their own testing as the basis for their affirmation.

Table K - Acronyms and Definitions