Trellix OpenSSL
FIPS Object Module
Software Version 1.0.3

FIPS 140-2 Non-Proprietary Security Policy
Version 3.0

June 2022

Prepared by:
Acumen Security
2400 Research Blvd
Suite 395, Rockville, MD 20850
www.acumensecurity.net

Prepared for:
Trellix
6220 America Center Drive
San Jose, CA 95002

Modification History

Version	Date	Description
Version 1.0	11/1/2016	Initial Release
Version 1.1	11/29/2016	Updated based on internal review
Version 1.2	12/14/2016	Updated based on internal review
Version 1.3	12/23/2016	Updated based on internal review
Version 1.4	12/27/2016	Added CAVP algorithm certificate numbers
Version 1.5	12/28/2016	Updates based on quality review
Version 1.6	12/30/2016	Updates to cover page, removal of watermark and branding of document
Version 1.7	28/05/2017	Updates to address CMVP comments and quality review
Version 1.8	June 2017	Updated to address Comments
Version 1.9	October 2017	Updated to address typos
Version 2.0	May 2021	Updated to add new software version and operating environments
Version 2.1	February 2022	Updated to meet all current IGs and to prepare for upcoming transitions.
Version 3.0	June 2022	Updated to meet IG D.1-rev3 and add Trellix branding

References

Reference]	Full Specification Name
[FIPS 140-2]	Security Requirements for Cryptographic modules, May 25, 2001
[FIPS 180-4]	Secure Hash Standard
[FIPS 186-4]	Digital Signature Standard
[FIPS 197]	Advanced Encryption Standard
[FIPS 198-1]	The Keyed-Hash Message Authentication Code (HMAC)
[SP 800-38A]	Recommendation for Block Cipher Modes of Operation: Methods and Techniques
[SP 800-38B]	Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication
[SP 800-38C]	Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality
[SP 800-38D]	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC
[SP 800-56A R3]	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography
[SP 800-67 R2]	Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher
[SP 800-89]	Recommendation for Obtaining Assurances for Digital Signature Applications
[SP 800-90A R1]	Recommendation for Random Number Generation Using Deterministic Random Bit Generators
[SP 800-131A R2]	Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths

Table of Contents

1 Introduction

This document is the non-proprietary security policy for the Trellix OpenSSL FIPS Object Module, hereafter referred to as the Module.

The Module is a software library providing a C-language application program interface (API) for use by other processes that require cryptographic functionality. The Module is classified by FIPS 140-2 as a software module, multi-chip standalone module embodiment. The physical cryptographic boundary is the general purpose computer on which the module is installed. The logical cryptographic boundary of the Module is the fipscanister object module, a single object module file named *fipscanister.o (Linux®/Unix® and VxWorks®) or fipscanister.lib (Microsoft Windows ®)*. The Module performs no communications other than with the calling application (the process that invokes the Module services).

The FIPS 140-2 security levels for the Module are as follows:

Security Requirement	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	2
Finite State Model	1
Physical Security	NA
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	3
Mitigation of Other Attacks	NA

Table 1 – Security Level of Security Requirements

The Module's software version for this validation is 1.0.3.

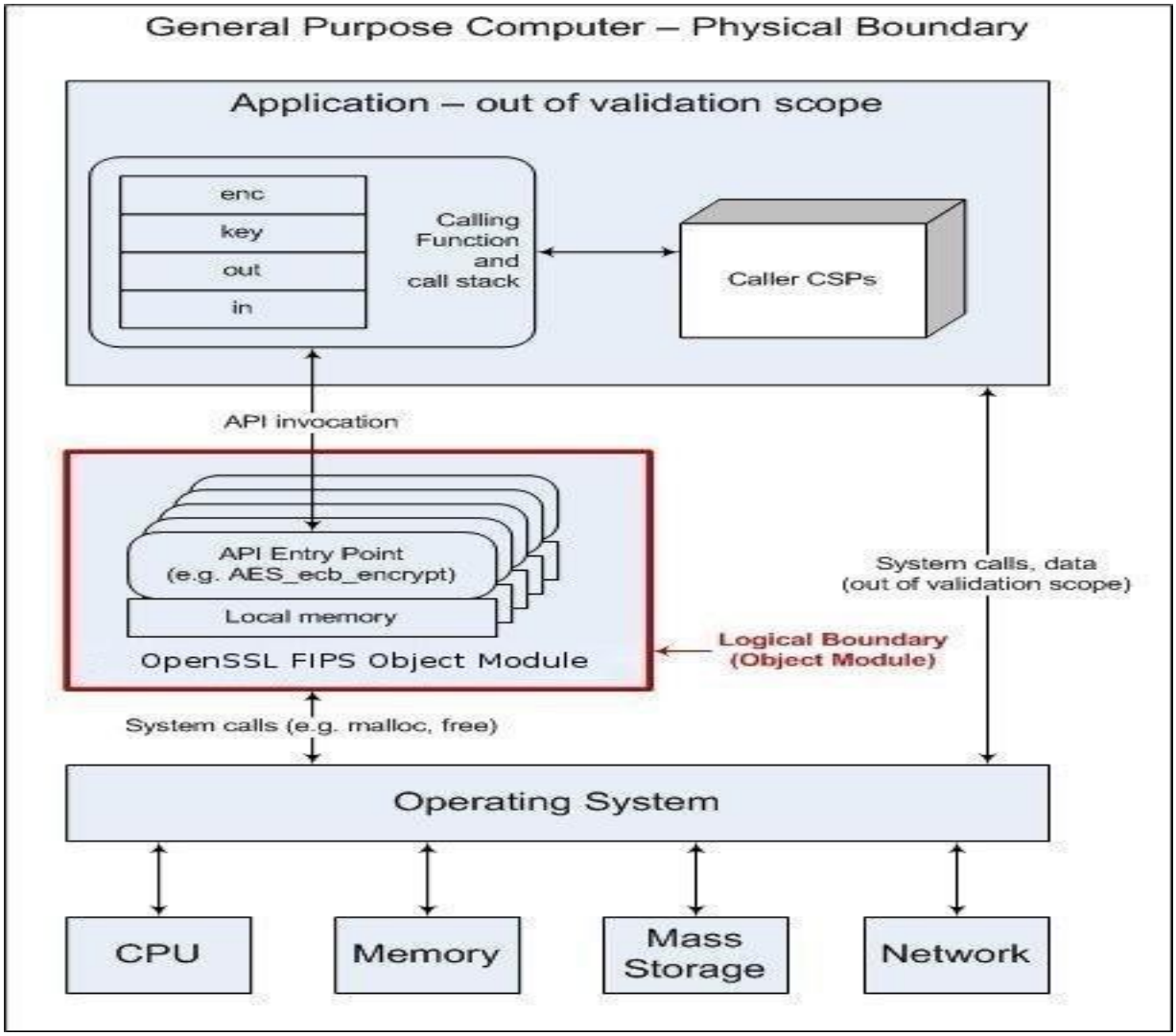

Figure 1 – Module Block Diagram

2 Tested Configurations

#	Operational Environment	Platform and Processor
1	Windows Server 2019 H2 64-bit on VMware ESXi 6.7.0	Intel Taylor Pass 2U Xeon DP Quad Board Server with Intel® Xeon® CPU E5-2699 without PAA
2	Windows 10 32-bit on VMware ESXi 6.7.0	Intel Taylor Pass 2U Xeon DP Quad Board Server with Intel® Xeon® CPU E5-2699 without PAA
3	Ubuntu Server 16.04 on VMware ESXi 6.7.0	Intel Taylor Pass 2U Xeon DP Quad Board Server with Intel® Xeon® CPU E5-2699 with PAA
4	McAfee Linux Operating System v3.8.0	Dell PowerEdge 610 with Intel® Xeon® CPU X5560 with PAA
5	SUSE Enterprise 12 SP3 on VMware ESXi 6.7.0	Intel Taylor Pass 2U Xeon DP Quad Board Server with Intel® Xeon® CPU E5-2699 with PAA
6	Darwin 10.15.7 (MacOS) on ESXi 6.7.0	MacBook Pro 13 with Intel® Xeon® CPU E5-1680 with PAA
7	Darwin 10.15.7 (MacOS) on ESXi 6.7.0	MacBook Pro 13 with Intel® Xeon® CPU E5-1680 without PAA

Table 2 – Tested Configurations

See Appendix A for additional information on build method. See Appendix C for a list of the specific compilers used to generate the Module.

3 Ports and Interfaces

The physical ports of the Module are the same as the system on which it is executing. The logical interface is a C-language application program interface (API).

Logical interface type	Description
Control input	API entry point and corresponding stack parameters
Data input	API entry point data input stack parameters
Status output	API entry point return values and status stack parameters
Data output	API entry point data output stack parameters

Table 3 – Logical interfaces

As a software module, control of the physical ports is outside module scope. However, when the module is performing self-tests, or is in an error state, all output on the logical data output interface is inhibited. The module is single-threaded and in error scenarios returns only an error value (no data output is returned).

4 Modes of Operation and Cryptographic Functionality

The Module supports FIPS 140-2 Approved, Allowed and Non-Approved algorithms in a single mixed mode of operation.

Note: There are some algorithms that were tested but are not used in the Approved mode of operation. Only the algorithms, modes, and key sizes that are used by the module in the Approved mode of operation are shown in this table

Function	Algorithm	Options	Certificate #
Random Number Generation; Symmetric key Generation	[SP 800-90] DRBG Prediction resistance supported for all variations	Hash DRBG HMAC DRBG, no reseed CTR DRBG (AES), derivation function supported	A2624
	[FIPS 197] AES [SP 800-38A]	128/ 192/256 ECB, CBC, OFB, CFB 1, CFB 8, CTR, CCM; GCM; CMAC generate and verify 128/192 CFB 128	A2624
	[SP 800-38B] CMAC [SP 800-38C] CCM [SP 800-38D] GCM		
Message Digests	[FIPS 180-4]	SHA-1, SHA-2 (224, 256, 384, 512)	A2624
Keyed Hash	[FIPS 198] HMAC	SHA-1, SHA-2 (224, 256, 384, 512)	A2624
Digital Signature and Asymmetric Key Generation	[FIPS 186-4] RSA	SigGen9.31 (4096 with SHA-256, 384, 512)	A2624
		SigGenPSS (4096 with SHA-224, 256, 384, 512)	
		SigGenPKCS1.5 (4096 with SHA-224, 256, 384, 512)	
		SigVer9.31 (2048/3072/4096 with SHA-1, 256, 384, 512)	
		SigVerPKCS1.5 (2048/3072,4096 with SHA-1, 224, 256, 384, 512)	
		SigVerPSS (1024, 1536, 2048/3072,4096 with SHA-1, 224, 256, 384, 512)	
	[FIPS 186-4] RSA	SigGen9.31 (2048/3072 with SHA-1, 224, 256, 384, 512) Note: SHA-1 affirmed for use with protocols only	A2624
		SigVer9.31 (2048/3072 with SHA-1, 224, 256, 384, 512)	
		SigGenPSS (2048/3072 with SHA-1, 224, 256, 384, 512) Note: SHA-1 affirmed for use with protocols only	
		SigVerPSS (2048/3072 with SHA-1, 224, 256, 384, 512)	
		SigGenPKCS1.5 (2048/3072 with SHA-1, 224, 256, 384, 512) Note: SHA-1 affirmed for use with protocols only	
		SigPKCS1.5 (2048/3072 with SHA-1, 224, 256, 384, 512)	
		Key Generation (Random e) PGM (Probable Prime	

Function	Algorithm	Options	Certificate #
		Condition), (2048/3072)	
	[FIPS 186-4] DSA	PQG(gen) (2048, 224 with SHA-224, 256, 384, 512; 2048, 256 with SHA-256, 384, 512; 3072,256 with SHA-256, 384, 512)	A2624
		PQG Ver (1024, 160 with SHA-1, 224, 256, 384, 512; 2048, 224 with SHA-224, 256, 384, 512; 2048, 256 with SHA-256, 384, 512; 3072,256 with SHA-256, 384, 512)	
		Key Pair Gen (2048,224; 2048,256; 3072,256)	
		Sig Gen (2048,224 with SHA-1, 224, 256, 384, 512; 2048,256 with SHA-1, 224, 256, 384, 512; 3072,256 with SHA-1, 224, 256, 384, 512) note: SHA-1 affirmed for use with protocols only	
		SigVer (1024, 160 with SHA-1, 224, 256, 384, 512; 2048, 224 with SHA-1, 224, 256, 384, 512; 2048, 256 with SHA-1, 224, 256, 384, 512; 3072,256 with SHA-1, 256, 384, 512)	
	[FIPS 186-4] ECDSA	PKV: CURVES (ALL-P ALL-K ALL-B)	A2624
		PKG: CURVES (P-224 P-256 P-384 P-521 K-233 K-283 K-409 K-571 B-233 B-283 B-409 B-571 ExtraRandomBits TestingCandidates)	
		SigGen: CURVES(P-224: (SHA-1, 224, 256, 384, 512) P-256: (SHA-1, 224, 256, 384, 512) P-384: (SHA-1, 224, 256, 384, 512) P-521: (SHA-1, 224, 256, 384, 512) K-233: (SHA-1, 224, 256, 384, 512) K-283: (SHA-1, 224, 256, 384, 512) K-409: (SHA-1, 224, 256, 384, 512) K-571: (SHA-1, 224, 256, 384, 512) B-233: (SHA-1, 224, 256, 384, 512) B-283: (SHA-1, 224, 256, 384, 512) B-409: (SHA-1, 224, 256, 384, 512) B-571: (SHA-1, 224, 256, 384, 512)) SIG(gen) with SHA-1 affirmed for use with protocols only.	
		SigVer: CURVES(P-192: (SHA-1, 224, 256, 384, 512) P-224: (SHA-1, 224, 256, 384, 512) P-256: (SHA-1, 224, 256, 384, 512) P-384: (SHA-1, 224, 256, 384, 512) P-521: (SHA-1, 224, 256, 384, 512) K-163: (SHA-1, 224, 256, 384, 512) K-233: (SHA-1, 224, 256, 384, 512) K-283: (SHA-1, 224, 256, 384, 512) K-409: (SHA-1, 224, 256, 384, 512) K-571: (SHA-1, 224, 256, 384, 512) B-163: (SHA-1, 224, 256, 384, 512) B-233: (SHA-1, 224, 256, 384, 512) B-283: (SHA-1, 224, 256, 384, 512) B-409: (SHA-1, 224, 256, 384, 512) B-571: (SHA-1, 224, 256, 384, 512))	A2624
KAS-ECC-SSC	[SP 800-56A R3]	CURVES P-224, P-256 P-384, P-521	A2624
CKG	SP 800-133r2		Vendor affirmed

Table 4a – FIPS Approved Cryptographic Functions

The Module supports only NIST defined curves for use with ECDSA and ECC CDH.

Category	Algorithm	Description
Key Encryption, Decryption	RSA	The RSA algorithm may be used by the calling application for encryption or decryption of keys. No claim is made for SP 800-56B compliance, and no CSPs are established into or exported out of the module using these services

Table 4b – Non-FIPS Approved But Allowed Cryptographic Functions

The module supports the following non-FIPS 140-2 approved but allowed algorithms:

- RSA (key wrapping; key establishment methodology provides between 112 and 270 bits of encryption strength; non-compliant less than 112 bits of encryption strength)

The Module implements the following services which are Non-Approved per the SP 800-131A transition and current Implementation Guidance:

Function	Algorithm	Options
Digital Signature and Asymmetric Key Generation	[FIPS 186-2] RSA (non-compliant)	GenKey9.31, SigGen9.31, SigGenPKCS1.5, SigGenPSS (2048/3072/4096 with SHA-1)
	[FIPS 186-2] DSA (non-compliant)	PQG Gen, Key Pair Gen, Sig Gen (2048/3072 with SHA-1)
	[FIPS 186-4] DSA (non-compliant)	PQG Gen, Key Pair Gen, Sig Gen (2048/3072 with SHA-1)
	[FIPS 186-2] ECDSA (non-compliant)	PKG: CURVES (P-192 K-163 B-163) SIG(gen): CURVES(P-192 P-224 P-256 P-384 P-521 K-163 K-233 K-283 K-409 K-571 B-163 B-233 B-283 B-409 B-571)
	[FIPS 186-4] ECDSA (non-compliant)	PKG: CURVES (P-192 K-163 B-163) SigGen: CURVES(P-192: (SHA-1, 224, 256, 384, 512) P-224:(SHA-1) P-256:(SHA-1) P-384: (SHA-1) P-521:(SHA-1) K-163: (SHA-1, 224, 256, 384, 512) K-233:(SHA-1) K-283:(SHA-1) K-409:(SHA-1) K-571:(SHA-1) B-163: (SHA-1, 224, 256, 384, 512) B-233:(SHA-1) B-283: (SHA-1) B-409:(SHA-1) B-571:(SHA-1))
ECC CDH (non-compliant)	[SP 800-56A] (§5.7.1.2)	All NIST B, K curves and P curves size 192
Triple-DES (non-compliant)	[SP 800-67]	3-Key Triple-DES TECB, TCBC, TCFB, TOFB; CMAC generate and verify
XTS (non-compliant)	[SP 800-38E]	128/256 XTS

Table 4c – Non-Approved Cryptographic Functions

These algorithms shall not be used when operating in the FIPS Approved mode of operation.

Per IG 9.10, the Module implements a default entry point and automatically runs the FIPS self-tests upon startup.

4.1 Critical Security Parameters and Public Keys

All CSPs used by the Module are described in this section. All access to these CSPs by Module services are described in Section 4. The CSP names are generic, corresponding to API parameter data structures.

CSP Name	Description
RSA SGK	RSA (2048 to 15360 bits) signature generation key

RSA KDK	RSA (2048 to 16384 bits) key decryption (private key transport) key
DSA SGK	[FIPS 186-4] DSA (2048/3072) signature generation key
ECDSA SGK	ECDSA (All NIST defined B, K, and P curves except sizes 163 and 192) signature generation key
EC DH Private	EC DH (All NIST defined P curves except sizes 163 and 192) private key agreement key.
AES EDK	AES (128/192/256) encrypt / decrypt key
AES CMAC	AES (128/192/256) CMAC generate / verify key
AES GCM	AES (128/192/256) encrypt / decrypt / generate / verify key
HMAC Key	Keyed hash key (160/224/256/384/512)
Hash_DRBG CSPs	V (440/888 bits) and C (440/888 bits), entropy input (length dependent on security strength)
HMAC_DRBG CSPs	V (160/224/256/384/512 bits) and Key (160/224/256/384/512 bits), entropy input (length dependent on security strength)
CTR_DRBG CSPs	V (128 bits) and Key (AES 128/192/256), entropy input (length dependent on security strength)
CO-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for Crypto Officer role authentication
User-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for User role authentication

Table 4.1a – Critical Security Parameters

Authentication data is loaded into the module during the module build process, performed by an authorized operator (Crypto Officer), and otherwise cannot be accessed.

The module does not output intermediate key generation values.

CSP Name	Description
RSA SVK	RSA (1024 to 16384 bits) signature verification public key
RSA KEK	RSA (2048 to 16384 bits) key encryption (public key transport) key
DSA SVK	[FIPS 186-4] DSA (2048/3072) signature verification key
ECDSA SVK	ECDSA (All NIST defined B, K and P curves) signature verification key
EC DH Public	EC DH (All NIST defined P curves) public key agreement key.

Table 4.1b – Public Keys

For all CSPs and Public Keys:

Storage: RAM, associated to entities by memory location. The Module stores DRBG state values for the lifetime of the DRBG instance. The module uses CSPs passed in by the calling application on the stack. The Module does not store any CSP persistently (beyond the lifetime of an API call), with the exception of DRBG state values used for the Modules' default key generation service.

Generation: The Module implements SP 800-90A compliant DRBG services for creation of symmetric keys, and for generation of DSA, elliptic curve, and RSA keys as shown in Table 4a. The direct output, U, from the Approved DRBG is used as keying material as discussed in IG D.12 and SP 800-133r2. The calling application is responsible for storage of generated keys returned by the module.

Entry: All CSPs enter the Module's logical boundary in plaintext as API parameters, associated by memory location. However, none cross the physical boundary.

Output: The Module does not output CSPs, other than as explicit results of key generation services. However, none cross the physical boundary.

Destruction: Zeroization of sensitive data is performed automatically by API function calls for temporarily stored CSPs. In addition, the module provides functions to explicitly destroy CSPs related to random number generation services. The calling application is responsible for parameters passed in and out of the module.

Private and secret keys as well as seeds and entropy input are provided to the Module by the calling

application and are destroyed when released by the appropriate API function calls. Keys residing in internally allocated data structures (during the lifetime of an API call) can only be accessed using the Module defined API. The operating system protects memory and process space from unauthorized access. Only the calling application that creates or imports keys can use or export such keys. All API functions are executed by the invoking calling application in a non-overlapping sequence such that no two API functions will execute concurrently. An authorized application as user (Crypto-Officer and User) has access to all key data generated during the operation of the Module.

Because the amount of entropy loaded by the application is dependent on the “num” parameter used by the calling application, the minimum number of bits of entropy is considered equal to the “num” parameter selection of the calling application. The calling application must call the RAND_add() with the “num” parameter of at least 32-bytes (256-bits).

In the event Module power is lost and restored the calling application must ensure that any AES-GCM keys used for encryption or decryption are re-distributed.

Module users (the calling applications) shall use entropy sources that meet the security strength required for the random number generation mechanism as shown in [SP 800-90] Table 2 (Hash_DRBG, HMAC_DRBG), Table 3 (CTR_DRBG). This entropy is supplied by means of callback functions. Those functions must return an error if the minimum entropy strength cannot be met.

5 Roles, Authentication and Services

The Module implements the required User and Crypto Officer roles and requires authentication for those roles. Only one role may be active at a time and the Module does not allow concurrent operators. The User or Crypto Officer role is assumed by passing the appropriate password to the *FIPS_module_mode_set()* function. The password values may be specified at build time and must have a minimum length of 16 characters. Any attempt to authenticate with an invalid password will result in an immediate and permanent failure condition rendering the Module unable to enter the FIPS mode of operation, even with subsequent use of a correct password.

Authentication data is loaded into the Module during the Module build process, performed by the Crypto Officer, and otherwise cannot be accessed.

Since minimum password length is 16 characters, the probability of a random successful authentication attempt in one try is a maximum of $1/256^{16}$, or less than $1/10^{38}$. The Module permanently disables further authentication attempts after a single failure, so this probability is independent of time.

Both roles have access to all of the services provided by the Module.

- User Role (User): Loading the Module and calling any of the API functions.
- Crypto Officer Role (CO): Installation of the Module on the host computer system and calling of any API functions.

All services implemented by the Module are listed below, along with a description of service CSP access. The access modes are determined as follows:

- Generate (G): Generates the Critical Security Parameter (CSP_ using an approved Random Bit

Generator

- Read (R): Export the CSP

- Write (W): Enter/establish and store a CSP

- Destroy (D): Overwrite the CSP

- Execute (E): Employ the CSP

Service	Role	Description
Initialize	User, CO	Module initialization. Does not access CSPs. E: CO-AD-Digest, User-AD-Digest
Self-test	User, CO	Perform self tests (FIPS_selftest). Does not access CSPs.
Show status	User, CO	Functions that provide module status information: • Version (as unsigned long or const char *) • FIPS Mode (Boolean) Does not access CSPs.
Zeroize	User, CO	Functions that destroy CSPs: fips_drbg_uninstantiate D: DRBG CSPs (Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs) All other services automatically overwrite CSPs stored in allocated memory. Stack cleanup is the responsibility of the calling application.

Service	Role	Description
Random number generation	User, CO	Used for random number and symmetric key generation. • Seed or reseed a DRBG instance • Determine security strength of a DRBG instance • Obtain random data E: Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs.
Asymmetric key generation	User, CO	Used to generate DSA, ECDSA and RSA keys: G: RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK
Symmetric encrypt/decrypt	User, CO	Used to encrypt or decrypt data. E: AES EDK, AES GCM (passed in by the calling process).
Symmetric digest	User, CO	Used to generate or verify data integrity with CMAC. E: AES CMAC (passed in by the calling process)
Message digest	User, CO	Used to generate a SHA-1 or SHA-2 message digest. Does not access CSPs.
Keyed Hash	User, CO	Used to generate or verify data integrity with HMAC. E: HMAC Key (passed in by the calling process).
Key transport	User, CO	Used to encrypt or decrypt a key value on behalf of the calling process (does not establish keys into the module). E: RSA KDK, RSA KEK (passed in by the calling process).
Key agreement	User, CO	Used to perform key agreement primitives on behalf of the calling process (does not establish keys into the module). E: EC Diffie-Hellman Private, EC Diffie-Hellman Public (passed in by the calling process)
Digital signature	User, CO	Used to generate or verify RSA, DSA or ECDSA digital signatures. E: RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK (passed in by the calling process).
Utility	User, CO	Miscellaneous helper functions. Does not access CSPs.

Table 5 – Services and CSP Access

"Key transport" can refer to a) moving keys in and out of the module, or b) the use of keys by an external application. The latter definition is the one that applies to the OpenSSL FIPS Object Module

6 Self-test

The Module performs the self-tests listed below on invocation of Initialize or Self-test.

Algorithm	Type	Test Attributes
Software integrity	KAT	HMAC-SHA1
HMAC	KAT	One KAT per SHA1, SHA224, SHA256, SHA384 and SHA512 Per IG 9.3, this testing covers SHA POST requirements.
AES	KAT	Separate encrypt and decrypt, ECB mode, 128-bit key length
AES CCM	KAT	Separate encrypt and decrypt, 192 key length
AES GCM	KAT	Separate encrypt and decrypt, 256 key length
XTS-AES	KAT	128, 256-bit key sizes to support either the 256-bit key size (for XTS-AES-128) or the 512-bit key size (for XTS-AES-256) (legacy test)
AES CMAC	KAT	Sign and verify CBC mode, 128, 192, 256 key lengths
Triple-DES	KAT	Separate encrypt and decrypt, ECB mode, 3-Key (legacy test)
Triple-DES CMAC	KAT	CMAC generate and verify, CBC mode, 3-Key (legacy test)
RSA	KAT	Sign and verify using 2048 bit key, SHA-256, PKCS#1
DSA	PCT	Sign and verify using 2048 bit key, SHA-384
DRBG	KAT	CTR_DRBG: AES, 256-bit with and without derivation function HASH_DRBG: SHA256 HMAC_DRBG: SHA256
ECDSA	PCT	Keygen, sign, verify using P-224, K-233 and SHA512. The K-233 self-test is not performed for operational environments that support prime curve only (see Table 2).
ECC CDH	KAT	Shared secret calculation per SP 800-56A R3§5.7.1.2

Table 6a - Power On Self Tests (KAT = Known answer test; PCT = Pairwise consistency test)

The Module is installed using one of the set of instructions in Appendix A, as appropriate for the target system. The HMAC-SHA-1 of the Module distribution file as tested by the CMT Laboratory and listed in Appendix A is verified during installation of the Module file as described in Appendix A.

The *FIPS_mode_set()* function performs all power-up self-tests listed above with no operator intervention required, returning a "1" if all power-up self-tests succeed, and a "0" otherwise. This function is run as part of every module initialization. If any component of the power-up self-test fails an internal flag is set to prevent subsequent invocation of any cryptographic function calls.

The power-up self-tests may also be performed on-demand by calling *FIPS_selftest()*, which returns a "1" for success and "0" for failure. Interpretation of this return code is the responsibility of the calling application.

The Module also implements the following conditional tests:

Algorithm	Test
DRBG	Tested as required by [SP800-90] Section 11
DRBG	FIPS 140-2 continuous test for stuck fault
NDRNG	FIPS 140-2 continuous test for NDRNG
DSA	Pairwise consistency test on each generation of a key pair
ECDSA	Pairwise consistency test on each generation of a key pair
RSA	Pairwise consistency test on each generation of a key pair

Table 6b - Conditional Tests

In the event of a DRBG self-test failure the calling application must uninstantiate and re-instantiate the DRBG per the requirements of [SP 800-90A]; this is not something the Module can do itself.

Pairwise consistency tests are performed for both possible modes of use, e.g. Sign/Verify and Encrypt/Decrypt.

7 Operational Environment

The tested operating systems segregate user processes into separate process spaces. Each process space is logically separated from all other processes by the operating system software and hardware. The Module functions entirely within the process space of the calling application, and implicitly satisfies the FIPS 140-2 requirement for a single user mode of operation.

8 Mitigation of other Attacks

The module is not designed to mitigate against attacks which are outside of the scope of FIPS 140-2.

Appendix A Installation and Usage Guidance

The build and target systems may be the same type of system or even the same device or may be different systems – the Module supports cross-compilation environments.

Each of these command sets are relative to the top of the directory containing the uncompressed and expanded contents of the distribution file *trellix-fips-1.0.3a.txz*. The command sets are:

Linux:

```
./config
make
make install
```

Mac:

```
cd build_tools
make
make install
```

Windows 64:

```
cd build_tools

trellix-fips.bat <path>/trellix-fips-1.0.3a/build_tools Release x64 build
```

Windows 32:

```
cd build_tools

trellix-fips.bat <path>/trellix-fips-1.0.3a/build_tools Release Win32 build
```

Installation instructions

1. Download and copy the distribution file to the build system.
2. Verify the SHA-256 digest of the distribution file; see Appendix B. An independently acquired FIPS 140-2 validated implementation of SHA-256 must be used for this digest verification. Note that this verification can be performed on any convenient system and not necessarily on the specific build or target system.
3. Unpack the distribution
   gunzip -c trellix-fips-1.0.3a.txz
4. Execute one of the installation command set as shown above. No other command sets shall be used.
5. The resulting *fipscanister.o* or *fipscanister.lib* file is now available for use.

Note that failure to use one of the specified commands sets exactly as shown will result in a module that cannot be considered compliant with FIPS 140-2.

Linking the Runtime Executable Application

Note that applications interfacing with the FIPS Object Module are outside of the cryptographic boundary. When linking the application with the FIPS Object Module two steps are necessary:

1. The HMAC-SHA-1 digest of the FIPS Object Module file must be calculated and verified against the installed digest to ensure the integrity of the FIPS object module.

2. A HMAC-SHA1 digest of the FIPS Object Module must be generated and embedded in the FIPS Object Module for use at runtime initialization.

The fips_standalone_sha1 command can be used to perform the verification of the FIPS Object Module and to generate the new HMAC-SHA-1 digest for the runtime executable application. Failure to embed the digest in the executable object will prevent initialization of FIPS mode.

AES-GCM IV Construction/Usage

The AES GCM IV generation is in compliance with IG A.5 Scenario 1 for TLS 1.2. The IV is generated in accordance with RFC5288 and is compliant with Section 3.3.1 of SP 800-52 rev2. The module ensures that the 64-bit nonce_explicit part of the IV is a strictly increasing counter. The module ensures that that when the deterministic part of the IV uses the maximum number of possible values and new session key is established. In case the module's power is lost and then restored, the key used for the AES GCM encryption or decryption shall be redistributed.

Appendix B Controlled Distribution File Fingerprint

The *McAfee OpenSSL FIPS Object Module v1.0.3* consists of the McAfee FIPS Object Module (the *fipscanister.o* or *fipscanister.lib* contiguous unit of binary object code) generated from the specific source files.

The source files are in the specific special OpenSSL distribution *trellix-fips-1.0.3a.txz*with SHA-256 digest of 75d3754b855a0c646a28e5917600a2500caba922b8ba7b52eb5043566fabb53c

The distributions are obtained by directly contacting the Internal Product Certification team.

The set of files specified in this tar file constitutes the complete set of source files of this module.
There shall be no additions, deletions, or alterations of this set as used during module build. The OpenSSL distribution tar file (and patch file if used) shall be verified using the above SHA-256 digest.

Appendix C Compilers

This appendix lists the specific compilers used to generate the Module for the respective Operational Environments. Note this list does not imply that use of the Module is restricted to only the listed compiler versions, only that the use of other versions has not been confirmed to produce a correct result.

#	Operational Environment	Compiler
1	Windows Server 2019 64 bit	Microsoft Visual Studio Professional 2017 15.8.5
2	Windows 10 64 bit	Microsoft Visual Studio Professional 2017 15.8.5
3	Ubuntu Server 16.04 on VMware ESXi 6.7.0	gcc 10.2.0
4	McAfee Linux Operating System v3.8.0	gcc 4.8.5
5	SUSE Enterprise 12 SP3 on VMware ESXi 6.7.0	gcc 7.5.0
6	Darwin 10.15.7 on ESXi 6.7.0	clang-1200.0.32.29
7	Darwin 10.15.7 on ESXi 6.7.0	clang-1200.0.32.29

Table 7 - Compilers