Cisco FIPS Object Module

FIPS 140-2 Non-Proprietary Security Policy

Cisco Systems, Inc.

DOCUMENT VERSION: 1.0
July 26, 2017

Table of Contents

1 INTRODUCTION1

1.1 PURPOSE 1
1.2 MODULE VALIDATION LEVEL 1
1.3 REFERENCES 1
1.4 TERMINOLOGY 2
1.5 DOCUMENT ORGANIZATION 2

2 CISCO FIPS OBJECT MODULE 2

3 CRYPTOGRAPHIC MODULE CHARACTERISTICS 3

3.1 MODULE INTERFACES 4
3.2 ROLES AND SERVICES 5
3.3 PHYSICAL SECURITY 7
3.4 CRYPTOGRAPHIC ALGORITHMS 7
3.4.1 Approved Cryptographic Algorithms 7
3.4.2 Non-FIPS Approved Algorithms Allowed in FIPS Mode 8
3.5 CRYPTOGRAPHIC KEY MANAGEMENT 8
3.5.1 Key Generation 8
3.5.2 Key Storage 8
3.5.3 Key Access 8
3.5.4 Key Protection and Zeroization 9
3.6 SELF-TESTS 11
Self-tests performed 12

4 SECURE DISTRIBUTION AND OPERATION 14

4.1 SECURE DISTRIBUTION 14
4.2 SECURE OPERATION 14

APPENDIX A – ACRONYMS AND ABBREVIATIONS 15

1 Introduction

1.1 Purpose

This document is the non-proprietary Cryptographic Module Security Policy for the Cisco FIPS Object Module (FOM). This security policy describes how the FOM (Software Version: 6.2) meets the security requirements of FIPS 140-2, and how to operate it in a secure FIPS 140-2 mode. This policy was prepared as part of the Level 1 FIPS 140-2 validation of the Cisco FIPS Object Module.

FIPS 140-2 (Federal Information Processing Standards Publication 140-2 — *Security Requirements for Cryptographic Modules*) details the U.S. Government requirements for cryptographic Modules. More information about the FIPS 140-2 standard and validation program is available on the NIST website at http://csrc.nist.gov/groups/STM/index.html.

1.2 Module Validation Level

The following table lists the level of validation for each area in the FIPS PUB 140-2.

No.	Area Title	Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services, and Authentication	1
4	Finite State Model	1
5	Physical Security	N/A
6	Operational Environment	1
7	Cryptographic Key management	1
8	Electromagnetic Interface/Electromagnetic Compatibility	1
9	Self-Tests	1
10	Design Assurance	3
11	Mitigation of Other Attacks	N/A
	Overall Module validation level	**1**

Table 1 – Module Validation Level

1.3 References

This document deals only with operations and capabilities of the Cisco FIPS Object Module in the technical terms of a FIPS 140-2 cryptographic Module security policy. More information is available from the following sources:

For answers to technical or sales related questions please refer to the contacts listed on the Cisco Systems website at www.cisco.com.

The NIST Validated Modules website (http://csrc.nist.gov/groups/STM/cmvp/validation.html) contains contact information for answers to technical or sales-related questions for the Module.

1.4 Terminology

In this document, the Cisco FIPS Object Module is referred to as FOM or the Module.

1.5 Document Organization

The Security Policy document is part of the FIPS 140-2 Submission Package. In addition to this document, the Submission Package contains:

- Vendor Evidence document
- Finite State Machine
- Other supporting documentation as additional references

This document provides an overview of the Cisco FIPS Object Module and explains the secure configuration and operation of the Module. This introduction section is followed by Section 2, which details the general features and functionality of the Module. Section 3 specifically addresses the required configuration for the FIPS-mode of operation.

With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Validation Submission Documentation is Cisco-proprietary and is releasable only under appropriate non-disclosure agreements. For access to these documents, please contact Cisco Systems.

2 Cisco FIPS Object Module

The Cisco FIPS Object Module is a software library that provides cryptographic services to a vast array of Cisco's networking and collaboration products.

The Module provides FIPS 140 validated cryptographic algorithms and KDF functionality for services such as IPSec (IKEv2), SRTP, SSH, TLS, and SNMPv3. The Module does not directly implement any of these protocols, instead it provides the cryptographic primitives and functions to allow a developer to implement the various protocols. These protocols have not been reviewed or tested by either the CAVP or the CMVP.

The Module is based on the OpenSSL FIPS canister with additions to support Suite B algorithms.

3 Cryptographic Module Characteristics

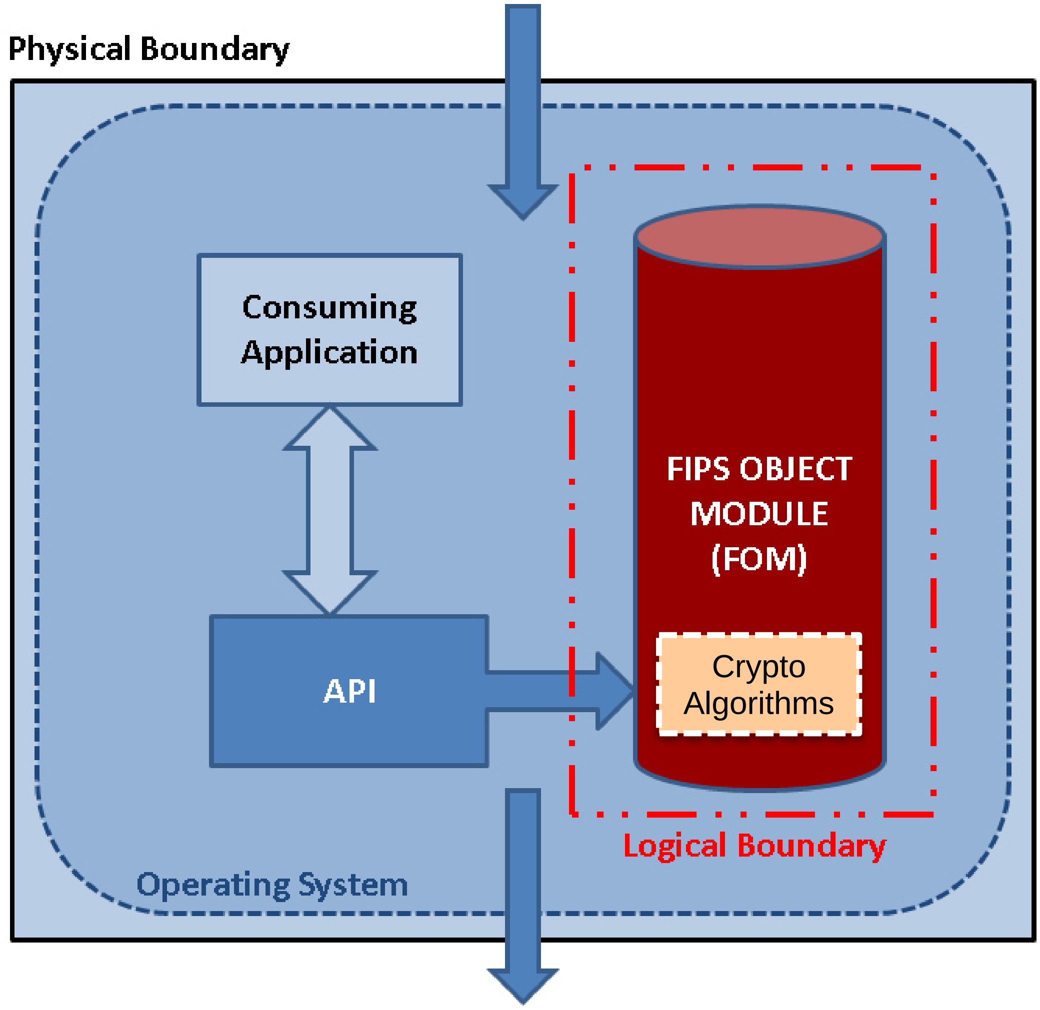

Figure 1 – FOM block diagram

The Module is a multi-chip standalone cryptographic Module. For the purposes of the FIPS 140-2 level 1 validation, the FOM is a single object Module file named fipscanister.o (Linux / FreeBSD Android) or fipscanister.lib (Microsoft Windows). The object code in the object Module file is incorporated into the runtime executable application at the time the binary executable is generated. The Module performs no communications other than with the consuming application (the process that invokes the Module services via the Module's API).

The Module's logical block diagram is shown in Figure 1 above. The dashed red border denotes the logical cryptographic boundary of the Module. The physical cryptographic boundary of the Module is the enclosure of the system on which it is executing and is denoted by the solid black border.

This Module was tested on the following platforms for the purposes of this FIPS validation:

#	Platform	Operating System	Processor
1	Google Nexus 5x	Android 3.10	ARM v8
2	Apple iPad Air 2	Apple iOS 9	ARM v8
3	Supermicro Intel Xeon E5	FreeBSD 10.3	Intel Xeon
4	Lenovo M900	Linux 3.10	Intel Core i5
5	Lenovo M900	Linux 3.10	Intel Core i5 (with AES-NI)
6	Cisco WLC 5508	Linux 3.10	MIPS 64
7	Cisco ASA FPR-2100	Linux 3.10	MIPS 64 (little Endian – with assembler)
8	Cisco WLC 5508	Linux 2.6	MIPS 64 (big Endian – with assembler)
9	Lenovo M900	MS Windows 10	Intel Core i5
10	Lenovo M900	MS Windows 10	Intel Core i5 (with AES-NI)
11	Cisco UCS C220 M4	FreeBSD 10.3	Intel Xeon E5

Table 2 – Tested Operational Environments (OEs)

3.1 Module 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).

The Data Input interface consists of the input parameters of the API functions. The Data Output interface consists of the output parameters of the API functions. The Control Input interface consists of the actual API functions. The Status Output interface includes the return values of the API functions.

The Module provides a number of physical and logical interfaces to the application (and the device upon which it is running), and the physical interfaces provided by the Module are mapped to the following FIPS 140-2 defined logical interfaces: data input, data output, control input, and status output. The logical interfaces and their mapping are described in the following table:

Interface	Description
Data Input	API input parameters - plaintext and/or ciphertext data
Data Output	API output parameters - plaintext and/or ciphertext data
Control Input	API function calls - function calls, or input arguments that specify commands and control data used to control the operation of the Module
Status Output	API return codes- function return codes, error codes, or output arguments that receive status information used to indicate the status of the Module
Power	Not Applicable

Table 3 – FIPS 140-2 Logical Interfaces

3.2 Roles and Services

The Module meets all FIPS 140-2 level 1 requirements for Roles and Services, implementing both Crypto-User and Crypto-Officer roles. As allowed by FIPS 140-2, the Module does not support user authentication for those roles. Only one role may be active at a time and the Module does not allow concurrent operators.

The User and Crypto Officer roles are implicitly assumed by the entity accessing services implemented by the Module. The Crypto Officer can install and initialize the Module. The Crypto Officer role is implicitly entered when installing the Module or performing system administration functions on the host operating system.

- User Role: Loading the Module and calling any of the API functions. This role has access to all of the services provided by the Module.
- Crypto-Officer Role: All of the User Role functionality as well as installation of the Module on the host computer system. This role is assumed implicitly when the system administrator installs the Module library file.

The following table lists the approved or non-approved but allowed services available in FIPS Approved mode.

Service	Role	CSP	Access
Module Installation	Crypto Officer	None	N/A
Symmetric encryption/decryption	User, Crypto Officer	Symmetric keys AES, AES-XTS, 3-key Triple-DES	Execute
Symmetric legacy decryption	User, Crypto Officer	2-key Triple-DES	Execute
Symmetric Digest	User, Crypto Officer	AES CMAC key	Execute
Key transport	User, Crypto Officer	Asymmetric private key RSA	Execute
Key agreement	User, Crypto Officer	DH and ECDH private key	Execute
Digital signature	User, Crypto Officer	Asymmetric private key RSA, DSA, ECDSA	Execute
Key Generation (Asymmetric)	User, Crypto Officer	Asymmetric keys DSA, ECDSA, and RSA	Write/execute
Key Generation (Symmetric)	User, Crypto Officer	Symmetric keys AES, Triple-DES	Write/execute
Key Derivation	User, Crypto Officer	AES, Shared Secret, HMAC	Write/execute
Keyed Hash (HMAC)	User, Crypto Officer	HMAC key (Key sizes must be a minimum of 112-bits)	Execute
Message digest (SHS)	User, Crypto Officer	None	N/A
Random number generation	User, Crypto Officer	Seed/entropy input, V, C, and Key	Write/execute
Show status	User, Crypto Officer	None	N/A
Module initialization	User, Crypto Officer	None	N/A
Perform Self-test	User, Crypto Officer	None	N/A
Zeroization	User, Crypto Officer	All CSPs	N/A

Table 4 – Roles, Services, and Keys (Approved Mode)

The following table lists the non-Approved services available in non-approved mode.

Service	Role	CSP	Access
Random number generation (as per the DRBG defined in SP 800-90A)	User, Crypto Officer	Seed/entropy input, V, C, and Key	Write/execute
Keyed Hash (HMAC)	User, Crypto Officer	HMAC key (Key sizes less than 112-bits)	Execute
Symmetric legacy encryption	User, Crypto Officer	2-key Triple-DES	Execute
Legacy digital signature	User, Crypto Officer	Asymmetric private key RSA	Execute

Table 5 – Roles, Services, and Keys (Non-Approved Mode)

3.3 Physical Security

The Module is comprised of software only and thus does not claim any physical security.

3.4 Cryptographic Algorithms

The Module implements a variety of approved and non-approved algorithms.

3.4.1 Approved Cryptographic Algorithms

The Module supports the following FIPS 140-2 approved algorithm implementations:

Algorithm	Algorithm Certificate Numbers
AES	4233, 4234, 4235, 4236, 4237
CCM	4233, 4234
CVL	981, 982, 983, 984
SP 800-90A DRBG	1316, 1317
DSA	1129, 1130
ECDSA	978, 979
HMAC	2771, 2772, 2773, 2774, 2775, 2776
KBKDF (SP800-108)	108, 109
RSA	2285, 2286
SHS	3470, 3471, 3472, 3473, 3474, 3475
Triple-DES	2292, 2293, 2294

Table 6 – Approved Cryptographic Algorithms

It should be noted that the XTS-AES mode, included in the AES algorithm certificates numbers 4233 and 4234 in Table 6, and as defined in NIST SP 800-38E and referred to in "*Annex A: Approved Security Functions for FIPS PUB 140-2*" '*Symmetric Key*', Section 1, '*Advanced Encryption Standard (AES)*', should only be used for the cryptographic protection of data on storage devices.

3.4.2 Non-FIPS Approved Algorithms Allowed in FIPS Mode

The Module supports the following non-FIPS approved algorithms which are permitted for use in the FIPS approved mode:

- Diffie-Hellman (key agreement; key establishment methodology provides between 112 and 219 bits of encryption strength)
- EC Diffie-Hellman (key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength)
- RSA (key wrapping; key establishment methodology provides between 112 and 132 bits of encryption strength)
- MD5 – (Per IG G.13 may be allowed in Approved mode of operation when used as part of an approved key transport scheme)

3.5 Cryptographic Key Management

3.5.1 Key Generation

The Module supports generation of DH, ECDH, FIPS 186-4 DSA, FIPS 186-4 RSA, and FIPS 186-4 ECDSA public-private key pairs. The Module employs a NIST SP 800-90A random number generator for creation of both symmetric keys and the seed for asymmetric key generation.

The entropy and seeding material for the NDRNG is provided to it by the external calling application (and not by the Module) which is outside the Module's logical boundary but contained within the Module's physical boundary. The minimum effective strength of the SP 800-90A DRBG seed is required to be at least 112 bits when used in a FIPS approved mode of operation, therefore the minimum number of bits of entropy requested when the Module makes a call to the SP 800-90A DRBG is 112. No assurance of the minimum strength of generated keys.

Module users (the external calling applications) shall use entropy sources which meet the security strength required for the random number generation mechanism as shown in SP 800-90A, Table 2 (Hash_DRBG, HMAC_DRBG), and 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.

3.5.2 Key Storage

Public and private keys are provided to the Module by the calling process, and are destroyed when released by the appropriate API function calls. The Module does not perform persistent storage of keys.

3.5.3 Key Access

An authorized application as user (the Crypto-User) has access to all key data generated during the operation of the Module.

3.5.4 Key Protection and Zeroization

Keys residing in internally allocated data structures can only be accessed using the Module defined API. The operating system protects memory and process space from unauthorized access. Zeroization of sensitive data is performed automatically by API function calls for intermediate data items.

Only the process that creates or imports keys can use or export them. No persistent storage of key data is performed by the Module. All API functions are executed by the invoking process in a non-overlapping sequence such that no two API functions will execute concurrently.

All CSPs can be zeroized by power-cycling the Module (with the exception of the Software Integrity key). In the event Module power is lost and restored the consuming application must ensure that any AES-GCM keys used for encryption or decryption are re-distributed.

The Module supports the following keys and critical security parameters (CSPs):

ID	Algorithm	Size	Description
Asymmetric Keys	RSA DSA ECDSA	RSA: 2,048, 3,072 bits DSA: 2,048, 3,072 bits ECDSA: P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571	Used for signature verification. RSA: Also used for key transport (where the size of the modulus is greater than or equal to 2048 bits)
Asymmetric Keys	RSA DSA ECDSA	RSA: 2,048, 3,072 bits DSA: 2,048, 3,072 bits ECDSA: P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571	Used for signature generation with SHA-2 used in key pair generation. RSA: Also used for key transport (where the size of the modulus is greater than or equal to 2048 bits)
Symmetric Keys	AES Triple-DES	AES: 128, 192, 256 bits AES-XTS: 256, 512 bits Triple-DES: 128, 192 bits	Used for symmetric encryption/decryption
Diffie-Hellman/ EC Diffie-Hellman private key	DH ECDH	DH: Public Key – 2,048-10,000 bits Private Key – 224-512 bits ECDH: P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571	Used for key agreement
Hash_DRBG	DRBG (as per NIST SP 800-90A)	– V (440/888 bits) – C (440/888 bits) – entropy input (The length of the selected hash)	CSPs for Hash_DRBG as per NIST SP 800-90A.
HMAC_DRBG	DRBG (as per NIST SP 800-90A)	– V (160/224/256/384/512 bits) – Key (160/224/256/384/512 bits) – entropy input (The length of the selected hash)	CSPs for HMAC_DRBG as per NIST SP 800-90A.
CTR_DRBG	DRBG (as per NIST SP 800-90A)	– V (128 bits) – Key (AES 128/192/256) – entropy input (The length of the selected AES)	CSPs for CTR_DRBG as per NIST SP 800-90A.
Keyed Hash key	HMAC	All supported key sizes for HMAC (Key sizes must be a minimum of 112-bits)	Used for keyed hash
Software Integrity key	HMAC	HMAC-SHA-1	Used to perform software integrity test at power-on. This key is embedded within the Module.

ID	Algorithm	Size	Description
SNMPv3 Session Key	AES	AES: 128 bits	Derived via key derivation function defined in SP800-135 KDF (SNMPv3).
TLS Master Secret	Shared secret	48 bytes of pseudo-random data	Derived via key derivation function defined in SP800-135 KDF (TLS).
SSHv2 Session Key	AES	AES: 128, 192, 256 bits	Derived via key derivation function defined in SP800-135 KDF (SSH).
SKEYSEED	Shared secret	160 bits	Derived via key derivation function defined in SP800-135 KDF (IKEv2).
SKEYID	Shared secret	160 bits	Derived via key derivation function defined in SP800-135 KDF (IKEv2).
IKEv2 session authentication key	HMAC	HMAC-SHA-1	Derived via key derivation function defined in SP800-135 KDF (IKEv2).
IKEv2 session encryption key	AES	AES: 128, 192, 256 bits	Derived via key derivation function defined in SP800-135 KDF (IKEv2).

Table 7 – Cryptographic Keys and CSPs

3.6 Self-Tests

The Module performs both power-up self-tests at Module initialization[1] and continuous conditional tests during operation. Input, output, and cryptographic functions cannot be performed while the Module is in a self-test or error state as the Module is single threaded and will not return to the calling application until the power-up self-tests are complete. If the power-up self- tests fail subsequent calls to the Module will fail and thus no further cryptographic operations are possible.

[1] The FIPS mode initialization is performed prior to the application invoking the FIPS_mode_set() function call (which returns a “1” for success and “0” for failure). Initialization is performed by an OS Loader on Module power up.

Self-tests performed

- POSTs
  - AES Known Answer Test (Separate encrypt and decrypt)
  - AES-CCM Known Answer Test (Separate encrypt and decrypt)
  - AES-GCM Known Answer Test (Separate encrypt and decrypt)
  - AES-CMAC Known Answer Test
  - AES-XTS Known Answer Test (Separate encrypt and decrypt)
  - SP 800-90A DRBG Known Answer Tests
    - HASH_DRBG Known Answer Test
    - HMAC_DRBG Known Answer Test
    - CTR_DRBG Known Answer Test
  - FIPS 186-4 DSA Sign/Verify Test
  - FIPS 186-4 ECDSA Sign/Verify Test
  - HMAC Known Answer Tests
    - HMAC-SHA1 Known Answer Test
    - HMAC-SHA224 Known Answer Test
    - HMAC-SHA256 Known Answer Test
    - HMAC-SHA384 Known Answer Test
    - HMAC-SHA512 Known Answer Test
  - ECC CDH KAT
  - FIPS 186-4 RSA Known Answer Test (Separate sign and verify)
  - SHA-1 Known Answer Test
  - Software Integrity Test (HMAC-SHA1)
  - Triple-DES Known Answer Test (Separate encrypt and decrypt)

- Conditional tests
  - Pairwise consistency tests for RSA, DSA, and ECDSA
  - SP 800-90A DRBG Continuous random number generation tests
    - HASH_DRBG Continuous random number generation test
    - HMAC_DRBG Continuous random number generation test
    - CTR_DRBG Continuous random number generation test
- Critical Function Tests (applicable to the DRBG, as per SP 800-90A, Section 11)
  - Instantiate Test
  - Generate Test
  - Reseed Test
  - Uninstantiate Test

A single function call, *FIPS_mode_set()*, is required to enable the Module for operation in the FIPS 140-2 Approved mode. When the Module is in FIPS mode all security functions and cryptographic algorithms are performed in Approved mode.

FIPS mode can only be enabled after the application invokes the *FIPS_mode_set()* call which returns a "1" for success and "0" for failure. Interpretation of this return code is the responsibility of the host application. Prior to this invocation the Module has already gone through its initialization sequence.

The *FIPS_mode_set()* function checks that the initialization sequence and POSTs (performed by the OS Loader at Module power-up) have completed successfully. The initialization sequence starts with a check of the integrity of the runtime executable using a HMAC-SHA-1 digest computed at build time. If this computed HMAC-SHA-1 digest matches the stored known digest then the power-up self-tests, consisting of the algorithm specific Pairwise Consistency and Known Answer tests, are performed. If any component of the power-up self-test fails an internal global error flag is set to prevent subsequent invocation of any cryptographic function calls. Any such power-up self-test failure is a hard error that can only be recovered by reinstalling the Module[2]. If all components of the power-up self-test are successful then the Module is in FIPS mode. This function call also returns a "1" for success and "0" for failure, and interpretation of this return code is the responsibility of the host application.

[2] The FIPS_mode_set() function could be re-invoked but such re-invocation does not provide a means of recovering from an integrity test or known answer test failure.

4 Secure Distribution and Operation

4.1 Secure Distribution

The Cisco FOM is intended only for use by Cisco personnel and as such is accessible only from the secure Cisco internal web site. Only authorized Cisco personnel have access to the Module.

4.2 Secure Operation

The tested operating systems segregate user processes into separate process spaces. Each process space is an independent virtual memory area that is logically separated from all other processes by the operating system software and hardware. The Module functions entirely within the process space of the process that invokes it, and thus satisfies the FIPS 140-2 requirement for a single user mode of operation.

The Module is installed using one of the set of instructions in the 'CiscoSSL 6.2 FIPS Compliance Guide' document appropriate to the target system. A complete revision history of the source code from which the Module was generated is maintained in a version control database[3]. The HMAC-SHA-1 of the Module distribution file as tested by the CSTL Laboratory is verified during installation of the Module file as described in the 'CiscoSSL 6.2 FIPS Compliance Guide' document.

The HMAC fingerprint of the validated distribution file is:

`05184c2d26633c1ea1cdd4786ababdd09494467c`

Upon initialization of the Module by the OS loader directly after Module power-up, the power-up self-tests will execute. Successful completion of the power-up self- tests ensures that the Module is operating in the FIPS mode of operation.

The self-tests are called when initializing the Module, or alternatively using the *FIPS_selftest()* function call. Either of the aforementioned operations will enable the Module for operation in the FIPS 140-2 Approved mode. When the Module is in FIPS mode all security functions and cryptographic algorithms are performed in Approved mode.

[3] This database is internal to Cisco since the intended use of this crypto Module is by Cisco development teams.

Appendix A – Acronyms and Abbreviations

Term	Expansion / Definition
AES	Advanced Encryption Standard
API	Application Program Interface
CAVP	Cryptographic Algorithm Validation Program
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CDH	Cofactor Diffie-Hellman
CMAC	Cipher-Based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSE	Communications Security Establishment
CSP	Critical Security Parameter
CSTL	Commercial Solutions Testing Laboratory
CTR	Counter
CVL	Component Validation List
DES	Data Encryption Standard
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard
FOM	FIPS Object Module
GCM	Galois/Counter Mode
HMAC	Hash Message Authentication Code
HTTP	Hyper Text Transfer Protocol
IKE	Internet Key Exchange
IPSec	Internet Protocol Security
KAT	Known Answer Test
KBKDF	Key Based Key Derivation Function
KDF	Key Derivation Function
MAC	Message Authentication Code
MS	Microsoft
NDRNG	Non-deterministic RNG
NIST	National Institute of Standards and Technology
OS	Operating System
POST	Power-On Self-Test
RSA	Rivest Shamir and Adleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SNMP	Simple Network Management Protocol
SP	Special Publication
SRTP	Secure Real-time Transport Protocol
SSH	Secure Shell

Term	Expansion / Definition
STM	Security Management & Assurance
UCS	Unified Computing System
TLS	Transport Layer Security
WLC	Wireless LAN Controller
XEX	XOR Encrypt XOR
XOR	Exclusive OR
XTS	XEX Tweakable Block Cipher with Ciphertext Stealing