Cisco FIPS Object Module

FIPS 140-2 Non-Proprietary Security Policy

Cisco Systems, Inc.

DOCUMENT VERSION: 3.0
March 3, 2014

Table of Contents

1 INTRODUCTION 1

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 3

3 CRYPTOGRAPHIC MODULE CHARACTERISTICS 4

3.1 MODULE INTERFACES 5
3.2 ROLES AND SERVICES 6
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 7
3.4.3 Non-Approved Cryptographic Algorithms 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 10
3.6.1 Self-tests performed 10

4 SECURE DISTRIBUTION AND OPERATION 13

4.1 SECURE DISTRIBUTION 13
4.2 SECURE OPERATION 13

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: 4.1) 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, the library, 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 for services such as IPSEC, SRTP, SSH, TLS, 802.1x, etc. 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.

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

3 Cryptographic Module Characteristics

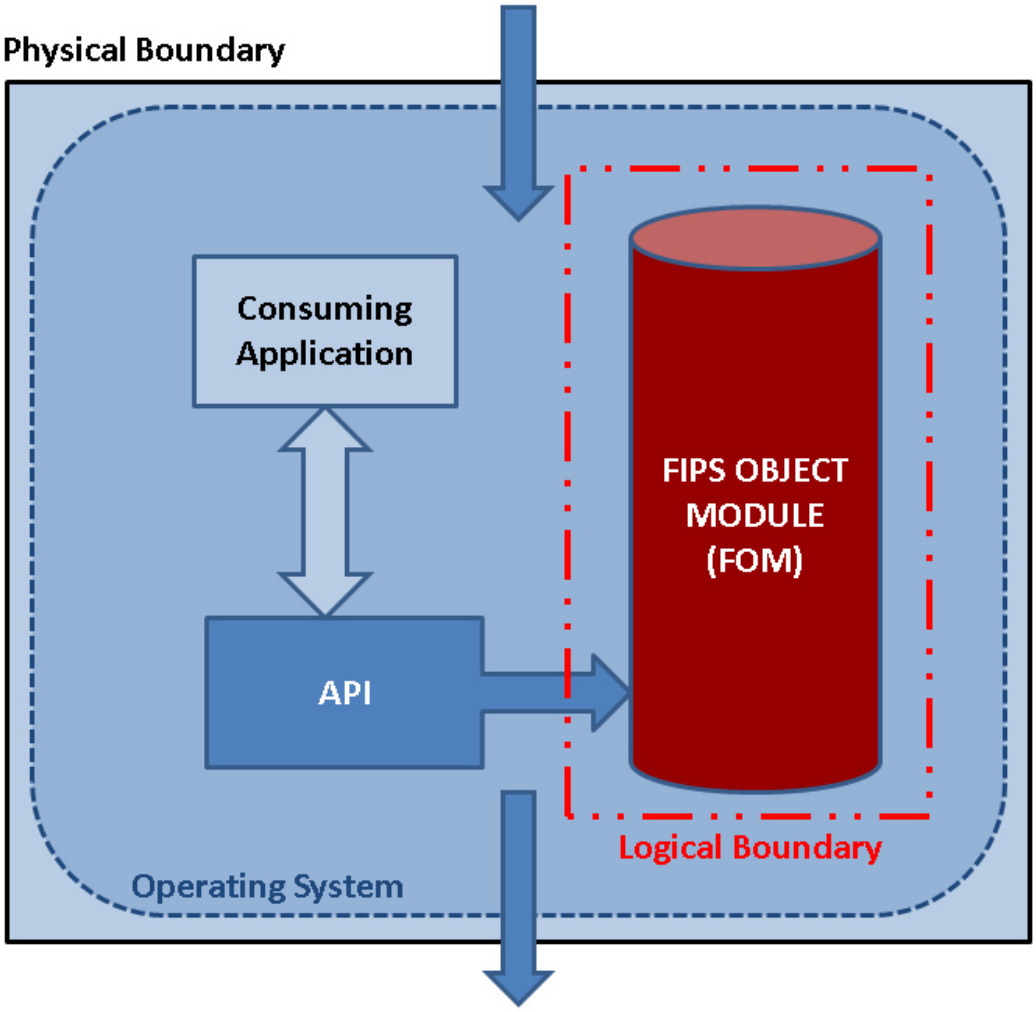

Figure 1 – FOM block diagram

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 red boundary.

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

#	Platform	Operating System	Processor
1	Cavium Octeon Evaluation Board EBH5200 (CN5230p2.0-750-SCP)	Linux 2.6	Cavium Octeon MIPS64
2	Cavium Octeon Evaluation Board EBH5200 (CN5230p2.0-750-SCP) (build with assembler code)	Linux 2.6	Cavium Octeon MIPS64
3	Cisco ASR1002	Linux 2.6	Freescale PowerPC-e500
4	Samsung Galaxy S II	Android 4.0	ARM Cortex-A9
5	Cisco UCS C200 M2	Windows 7	Intel Xeon
6	Cisco UCS C210 M2	Windows 7	Intel Xeon with AES-NI
7	Cisco UCS C210 M2	FreeBSD 9.0	Intel Xeon
8	Cisco UCS C22 M3	Linux 2.6	Intel Xeon with AES-NI
9	Cisco UCS C200 M2	Linux 2.6	Intel Xeon

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 device, 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, status output, and power. 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

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.

Service	Role	CSP	Access
Module Installation	Crypto Officer	None	N/A
Symmetric encryption/decryption	User, Crypto Officer	Symmetric keys AES, 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
Keyed Hash (HMAC)	User, Crypto Officer	HMAC key	Execute
Message digest (SHS)	User, Crypto Officer	None	N/A
Random number generation	User, Crypto Officer	Seed/entropy input, V, C, Key, and S	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

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 routers support the following FIPS 140-2 approved algorithm implementations:

Algorithm	Algorithm Certificate Numbers
AES	2678, 2685
DSA	812, 814
ECDSA	467, 471
HMAC	1664, 1672
DRBG	431, 435
RSA	1377, 1385
SHS	2247, 2256
Triple-DES	1606, 1611
CVL	151, 153

Table 5a – Approved Cryptographic Algorithms

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; non-compliant less than 112-bits of encryption strength)
- EC Diffie-Hellman (key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength; non-compliant less than 112-bits of encryption strength)
- RSA (key wrapping; key establishment methodology provides between 112 and 150 bits of encryption strength; non-compliant less than 112-bits of encryption strength)

3.4.3 Non-Approved Cryptographic Algorithms

As a result of the transitioning of the cryptographic algorithms and key lengths (as described in NIST Special Publication 800-131A) a number of the algorithms contained in the Table 5a above and listed under "Approved Cryptographic Algorithms" also have sizes which as of January 1, 2014 became Non-approved. Although the Approved and Non-approved algorithms share the same certificate number, it should be noted that the Non-approved versions appear in a separated parallel listing on the NIST CAVP website and which are referred to as 'Historical' to differentiate them from the Approved validation lists.

By policy, only those algorithms (and key lengths) which appear in the approved Validation lists (Table 5a above) shall be used in a FIPs approved mode of operation.

These algorithms may be used with the services specified in Table 4 in a non-Approved mode.

Algorithm	**Algorithm Certificate Numbers** (In the Historical Lists)
DSA	812, 814
ECDSA	467, 471
RSA	1377, 1385
CVL	151, 153

Table 6b – Non-approved Cryptographic Algorithms

3.5 Cryptographic Key Management

3.5.1 Key Generation

The Module supports generation of DH, ECDH, DSA, RSA, and FIPS 186-3 ECDSA public-private key pairs. The Module employs a NIST SP800-90A random number generator for creation of both symmetric keys and the seed for asymmetric key generation.

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
Symmetric Keys	AES Triple-DES	AES: 128, 192, 256 bits Triple-DES: 112, 168 bits Triple-DES Notes Related to use if Two-Key algorithm: *For the transition period from January 1, 2011 through December 31, 2015, the effective 100-bit security strength provided when no more than 2^{20} blocks are encrypted with the same key is allowed.* *To use the two-key Triple-DES algorithm to encrypt data or wrap keys in an Approved mode of operation, the module operator shall ensure that the same two-key Triple-DES key is not used for encrypting data (or wrapping keys) with more than 2^{20} plaintext data (or plaintext keys).*	Used for symmetric encryption/decryption
Asymmetric Keys	RSA DSA ECDSA	RSA: 1,024-4,096 bits DSA: 1,024, 2,048, 3,072 bits ECDSA: P-192, P-256, P-384, P-521	Used for signature verification. RSA: Also used for key transport.
Asymmetric Keys	RSA DSA ECDSA	RSA: 2,048-4,096 bits DSA: 2,048, 3,072 bits ECDSA: P-256, P-384, P-521	Used for signature generation with SHA-2 used in key pair generation. RSA: Also used for key transport.

ID	Algorithm	Size	Description
Diffie-Hellman/ EC Diffie-Hellman private key	DH ECDH	DH: Public Key – 2,048-10,000 bits Private Key – 224-512 bits ECDH: P-256, P-384, P-521	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 SP800-90A.
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 SP800-90A.
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 SP800-90A.
Dual_EC_DRBG	DRBG (as per NIST SP 800-90A)	– S (P-256, P-384, P-521) – entropy input (The length of the EC curve selected)	CSPs for Dual_EC_DRBG as per NIST SP800-90A.
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.

Table 7 – Cryptographic Keys and CSPs

3.6 Self-Tests

The Module performs both power-up self-tests at module initialization[1] and continuous condition 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.

3.6.1 Self-tests performed

- POST tests

[1] The FIPS mode initialization is performed when the application invokes the FIPS_mode_set() call which returns a "1" for success and "0" for failure

    - 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)
    - DRBG Known Answer Tests
        - HASH_DRBG Known Answer Test
        - HMAC_DRBG Known Answer Test
        - CTR_DRBG Known Answer Test
        - EC Dual_DRBG Known Answer Test
    - DSA Sign/Verify Test
    - FIPS 186-3 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
    - KAS ECC Primitive "Z" KAT
    - 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
    - Continuous random number generation test for approved DRBG.

- Critical Function Tests (applicable to the DRBG, as per SP800-90A, Section 11)
    - Instantiate Test
    - Generate Test
    - Reseed Test
    - Uninstantiate Test

A single initialization call, *FIPS_mode_set()*, is required to initialize 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.

The FIPS mode initialization is performed when 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 is uninitialized in the non-FIPS mode by default.

The *FIPS_mode_set()* function verifies 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.

A power-up self-test failure can only be cleared by a successful *FIPS_mode_set()* invocation. No operator intervention is required during the running of the self-tests.

[2] The FIPS_mode_set() function could be re-invoked but such re-invocation does not provide a means from 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 employees 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 4.0 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 4.0 FIPS Compliance Guide' document.

The HMAC fingerprint of the validated distribution file is:

```
3108a96df76a0f272c926f76eae363e0724c8086
```

Upon initialization of the Module using FIPS_mode_set, the power-up self- tests will run. 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.

[3] This database is internal to Cisco since the intended use of this crypto module is by Cisco dev teams