# IOS Common Cryptographic Module (IC2M)

**FIPS 140-2 Non-Proprietary Security Policy**
**Level 1 Validation**

**Version 1.5**

**June 12, 2020**

# Table of Contents

# 1 Introduction

## 1.1 Purpose

This is a non-proprietary Cryptographic Module Security Policy for Cisco's IOS Common Cryptographic Module (IC2M) Version Rel 5, This security policy describes how the module meets the security requirements of FIPS 140-2 Level 1 and how to run the modules in a FIPS 140-2 mode of operation and may be freely distributed.

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 | 1 |
| 6 | Operational Environment | N/A |
| 7 | Cryptographic Key management | 1 |
| 8 | Electromagnetic Interface/Electromagnetic Compatibility | 1 |
| 9 | Self-Tests | 1 |
| 10 | Design Assurance | 1 |
| 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 IC2M listed above in section 1 in the technical terms of a FIPS 140-2 cryptographic module security policy. More information is available on the routers 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, Cisco IOS Common Cryptographic Module is referred to as IC2M 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 IC2M identified in section 1 above 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 appliances. 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 IOS Common Cryptographic Module (IC2M)

This module provides the FIPS validated cryptographic algorithms for services requiring those algorithms. The module does not implement any protocols directly. Instead, it provides the cryptographic primitives and functions to allow IOS to implement those various protocols.

## 2.1 Cryptographic Module Characteristics

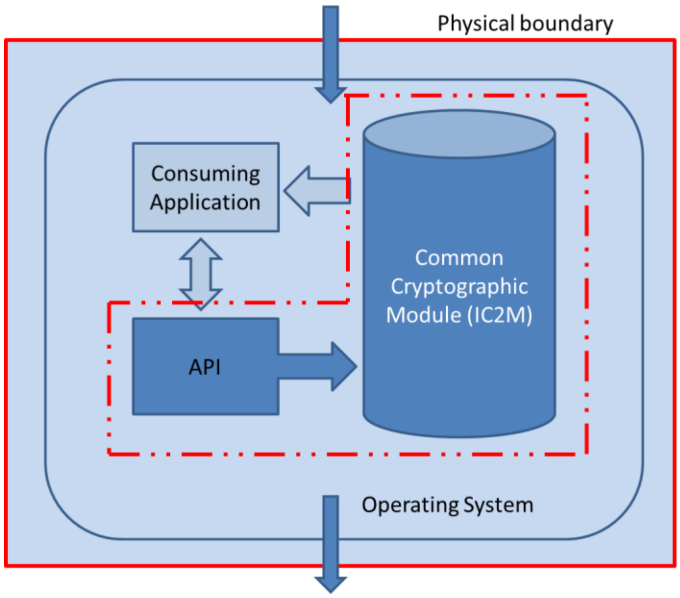


**Figure 1 - Cisco IC2M logical block diagram**

The module's logical block diagram is shown in Figure 1 above. The red dashed line area denotes the logical cryptographic boundary of the module. While the red solid line denotes the physical cryptographic boundary of the module which is the enclosure of the system on which the module is executed.

The IC2M is a single binary object file, sub_crypto_ic2m_k9.o (Cisco IOS) classed as a multi-chip standalone firmware, cryptographic module.  It is capable of being utilized and is validated on any platform running Cisco IOS containing the hardware listed in table 2:

| Processor | Operating System | Platform |
|---|---|---|
| Intel Xeon | IOS-XE 3.13 | Cisco ASR1K RP2 |
| Freescale SC8548H | IOS-XE 3.13 | Cisco ASR1K RP1 |
| Freescale 8752E | IOS 15.4 | Cisco ISR 2951 |
| Cavium CN5020 | IOS 15.4 | Cisco ISR 1921 |
| Cavium CN5220 | IOS 15.4 | Cisco ISR 2921 |
| MPC8358E | IOS 15.4 | Cisco ISR 891 |
| MPC8572C | IOS 15.4 | Cisco ESR 5940 |

 

| | | |
|---|---|---|
| MIPS64 | IOS-XE 16.3.2 | Cisco Catalyst 3850 |
| PPC 405 | IOS 15.2(4)E | Cisco IE2000 |
| PPC 465 | IOS 15.2(4)E | Cisco Catalyst 3560CX |
| PPC e5500 | IOS-XE 3.9.0E | Cisco Catalyst 4000 with SUP8LE |
| PPC e500 | IOS 15.4(1)SY1 | Cisco Catalyst 6000 with SUP2T |
| Intel Pentium | IOS 15.4 | Cisco Catalyst 6840 |
| Intel Xeon | IOS 16.3.2 | Cisco ASR1K RP1 |
| Intel Core i3 | IOS 15.4(1)SY1 | Cisco Catalyst 6000 with SUP6T |
| Intel Atom | IOS 16.3.2 | Cisco ISR 4321 |

**Table 2 - Tested Platforms**

## 2.2 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 logical interfaces to the system, and is 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 |
|---|---|
| **Date Input** | API input parameters<br><br>plaintext and/or ciphertext data |
| **Data Output** | API output parameters<br><br>plaintext and/or ciphertext data |
| **Control Input** | API function calls<br><br>function calls, or input arguments that specify commands and control data used to control the operation of the module |
| **Status Output** | API return codes<br><br>function return codes, error codes, or output arguments that receive status information used to indicate the status of the module |

**Table 3 - Logical Interfaces Details**

## 2.3 Roles and Services

The Module meets all FIPS 140-2 level 1 requirements for Roles and Services, implementing both Crypto Officer and User roles, which are classed as processes. As allowed by FIPS 140-2, the Module does not support user authentication for these roles, which is handled by the system implementing IC2M. 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.

- Installation of the Crypto Module which is embedded in the IOS image and installed on the IOS platform is assumed implicitly as the Crypto Officer when install occurs.

The services available only in FIPS mode to the Crypto Officer and User roles consist of the following:

| Services | Access | CSPs | Crypto Officer | User |
|---|---|---|---|---|
| Encryption/decryption* | execute | Symmetric keys AES, Triple-DES | X | X |
| Hash (HMAC)** | execute | HMAC SHA-1 key | X | X |
| Key agreement*** | execute | DH and ECDH public/private key | X | X |
| Key generation* | Write/execute | Symmetric key AES, Triple-DES | X | X |
| Key transport*** | execute | Asymmetric private key RSA | X | X |
| Message Digest (SHS)** | execute | None | X | X |
| Perform Self-Tests | Execute/read | N/A | X | X |
| Random number generator | execute | seed | X | X |
| Show Status | Execute | N/A | X | X |
| Signature signing*** | execute | Asymmetric private key ECDSA, RSA | X | X |
| Signature verification*** | execute | Asymmetric public key ECDSA, RSA | X | X |
| IKE/IPsec, SSH, TLS, SNMP, sRTP Key Establishment | execute | skeyid, skeyid_d, IKE session encrypt key, IKE session authentication key, IPsec encryption key, IPsec authentication key, SSH Session Key, TLS pre-master secret, TLS session encryption key, TLS session integrity key, SNMPv3 Password, snmpEngineID, SNMP session key, sRTP master key, sRTP encryption key, sRTP authentication key | X | X |

| | | | | |
|---|---|---|---|---|
| Zeroization | execute | Symmetric key, asymmetric key, HMAC-SHA-1 key, seed, skeyid, skeyid_d, IKE session encrypt key, IKE session authentication key, IPsec encryption key, IPsec authentication key, SSH Session Key, TLS pre-master secret, TLS session encryption key, TLS session integrity key, SNMPv3 Password, snmpEngineID, SNMP session key, sRTP master key, sRTP encryption key, sRTP authentication key | **X** | **X** |

**Table 4 – Services**

Note: Services marked with a single asterisk (*) may use non-compliant encryption algorithms (DES, RC2, RC4, or SEAL). Use of these algorithms are prohibited in a FIPS-approved mode of operation.

Note: Services marked with a double asterisk (**) may use non-compliant hashing algorithms (HMAC-MD5, MD2, or MD5). Use of these non-compliant hashing algorithms are prohibited in a FIPS-approved mode of operation and shall not be used.

Note: Services marked with a triple asterisk (***) may use non-compliant encryption strengths for EC Diffie-Hellman, Diffie-Hellman and RSA. Use of these non-compliant encryption strengths are prohibited in a FIPS-approved mode of operation and shall not be used. Refer to section 2.6 for descriptions of the minimum required encryption strengths for compliance.

## 2.4 Physical Security

The module obtains its physical security from any platform running Cisco IOS with production grade components as allowed by FIPS 140-2 level 1.

## 2.5 Cryptographic Key Management

Keys that reside 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, and on demand by the calling process using the module provided API function calls provided for that purpose.

Zeroization consists of overwriting the memory that store the key or refreshing the volatile memory. Keys can also be zeroized by cycling the power.

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

| Key/CSP Name | Generation/ Algorithm | Description | Storage and Zeroization |
|---|---|---|---|
| Asymmetric private key | RSA, ECDSA | RSA: 2048-4096 bits<br>ECDSA: P-256, P-384<br>Used for signature generation | Stored and zeroized outside the module in the host OS |
| Diffie-Hellman private key | DH | DH: 2048-4096 bits<br>Used for key agreement | Stored and zeroized outside the module in the host OS |
| EC Diffie-Hellman private key | ECDH | ECDH:, P-256, P-384<br>Used for key agreement | Stored and zeroized outside the module in the host OS |
| Firmware integrity key | HMAC-SHA-256 | Integrity test at power-on. Key embedded within module | Stored in plaintext and zeroized by uninstalling the module |
| HMAC-SHS Key | FIPS 198 | SHA-1, 256, 384 and 512<br>Message authentication code key | Stored and zeroized outside the module in the host OS |
| Symmetric Key | AES, Triple-DES | AES: 128, 192, 256 bits<br>Triple-DES: 168 bits<br>Used for symmetric encryption/decryption | Stored and zeroized outside the module in the host OS |
| DRBG entropy input | SP 800-90A CTR_DRBG | This is the entropy for SP 800-90A RNG. | Zeroized with generation of new seed |
| DRBG seed | SP 800-90A CTR_DRBG | This is the seed for SP 800-90A RNG. | Zeroized with generation of new seed |
| DRBG V | SP 800-90A CTR_DRBG | Internal V value used as part of SP 800-90A CTR_DRBG | Zeroized with generation of new seed |
| DRBG Key | SP 800-90A CTR_DRBG | Internal Key value used as part of SP 800-90A CTR_DRBG | Zeroized with generation of new seed |
| Diffie-Hellman public key | DH | DH: 2048-4096 bits<br>Used for key agreement | Stored and zeroized outside the module in the host OS |
| EC Diffie-Hellman public key | ECDH | ECDH:, P-256, P-384<br>Used for key agreement | Stored and zeroized outside the module in the host OS |
| Asymmetric public key | RSA, ECDSA | RSA: 2048-4096 bits<br>ECDSA: P-256, P-384<br>Used for signature verification.<br>RSA: Also used for key transport | Stored and zeroized outside the module in the host OS |

| | | | |
|---|---|---|---|
| skeyid | HMAC-SHA-1 (160-bits) | Value derived from the shared secret within IKE exchange. Zeroized when IKE session is terminated. | Stored and zeroized outside the module in the host OS |
| skeyid_d | HMAC-SHA-1 (160-bits) | The IKE key derivation key for non ISAKMP security associations. | Stored and zeroized outside the module in the host OS |
| IKE session encrypt key | Triple-DES (168-bits/AES (128/192/256-bits) | The IKE session encrypt key. | Stored and zeroized outside the module in the host OS |
| IKE session authentication key | HMAC-SHA-1 (160-bits) | The IKE session authentication key. | Stored and zeroized outside the module in the host OS |
| IPsec encryption key | Triple-DES (168-bits/AES (128/192/256-bits) | The IPSec encryption key. | Stored and zeroized outside the module in the host OS |
| IPsec authentication key | HMAC-SHA-1 (160-bits) | The IPSec authentication key. | Stored and zeroized outside the module in the host OS |
| SSH Session Key | Triple-DES (168-bits/AES (128/192/256-bits) | This is the SSH v2 session key. | Stored and zeroized outside the module in the host OS |
| TLS pre-master secret | Shared Secret (384-bits) | Shared Secret created using asymmetric cryptography from which new TLS session keys can be created | Stored and zeroized outside the module in the host OS |
| TLS session encryption key | Triple-DES (168-bits/AES (128/192/256-bits) | Key used to encrypt TLS session data | Stored and zeroized outside the module in the host OS |
| TLS session integrity key | HMAC-SHA-1 (160-bits) | HMAC-SHA-1 used for TLS data integrity protection | Stored and zeroized outside the module in the host OS |
| SNMPv3 Password | Shared Secret ( 8 – 25 characters) | The password use to setup SNMP v3 connection. | Stored and zeroized outside the module in the host OS |
| snmpEngineID | Shared Secret (32-bits) | A unique string used to identify the SNMP engine. | Stored and zeroized outside the module in the host OS |
| SNMP session key | AES (128 bits) | Encryption key used to protect SNMP traffic. | Stored and zeroized outside the module in the host OS |
| sRTP master key | AES (128 bits) | Key used to generate sRTP session keys | Stored and zeroized outside the module in the host OS |
| sRTP encryption key | AES (128 bits) | Key used to encrypt/decrypt sRTP packets | Stored and zeroized outside the module in the host OS |
| sRTP authentication key | HMAC | Key used to authenticate sRTP packets | Stored and zeroized outside the module in the host OS |

**Table 5 - Cryptographic Keys and CSPs**

*Note: The minimum 256-bits of random data from application will be required to be loaded. Failure to do this will result in an error when the DRBG is instantiated. No assurance of the minimum strength of generated key.*

All cryptographic 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.

## 2.6 Cryptographic Algorithms

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

**Approved Cryptographic Algorithms**

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

| Algorithm | Algorithm Certificate Number |
|---|---|
| AES | 2817, 2783, 3278, and 4583 |
| KTS (800-38F)[1] | 3278 and 4583 |
| DRBG | 481 and 1529 |
| CVL (800-135 KDF) | 253 and 1258 |
| CVL (KAS ECC/FFC) | 252 and 1257 |
| ECDSA | 493 and 1122 |
| HMAC | 1764 and 3034 |
| RSA | 1471 and 2500 |
| SHS | 2361, 2338 and 3760 |
| Triple-DES | 1670, 1671, 1688 and 2436 |
| KBKDF (SP 800-108) | 49 and 139 |
| CKG [2] | Vendor affirmed |

**Table 6 - Approved Cryptographic Algorithms**

Note: Per NIST SP 800-131A, Two-key Triple-DES is no longer be allowed for use in an approved mode of operation after 1 January, 2015.

[1] KTS (AES Cert. #3278; key establishment methodology provides 128 bits of encryption strength)
KTS (AES Cert. #4583; key establishment methodology provides between 128 and 256 bits of encryption strength)
[2] The resulting symmetric key or a generated seed is an unmodified output from the DRBG.

**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 (CVL Certs. #252 and #1257, key agreement; key establishment methodology provides between 112 and 150 bits of encryption strength; non-compliant less than 112 bits of encryption strength)
- EC Diffie-Hellman (CVL Certs. #252 and #1257, key agreement; key establishment methodology provides 128 or 192 bits of encryption strength; non-compliant less than 128 bits of encryption strength)
- RSA (key wrapping; key establishment methodology provides 112 or 128 bits of encryption strength; non-compliant less than 112 bits of encryption strength)

**Non-Approved Cryptographic Algorithms**

- DES
- HMAC-MD5
- MD2
- MD5
- RC2
- RC4
- SEAL

## 2.7 Self-Tests

The modules include an array of self-tests that are run automatically during startup and periodically when called during operations to prevent any secure data from being released and to insure all components are functioning correctly.

### *Self-tests performed*

- IC2M Self Tests
  - POSTs - IOS Common Crypto Module algorithm implementation
    - Firmware Integrity Test (HMAC SHA-256)
    - AES-CBC (encrypt/decrypt) KATs
    - AES-GCM (encrypt/decrypt) KATs
    - AES-CMAC KAT
    - DRBG KAT
    - ECDSA Sign/Verify PWCT
    - HMAC-SHA-1 KAT
    - HMAC-SHA-256 KAT

    - HMAC-SHA-384 KAT
    - HMAC-SHA-512  KAT
    - ECC Primitive “Z” KAT
    - FFC Primitive “Z” KAT
    - RSA KAT
    - SHA-1 KAT
    - SHA-256 KAT
    - SHA-384 KAT
    - SHA-512 KAT
    - Triple-DES (encrypt/decrypt) KATs
    - KBKDF KAT

  - POSTs - IOS Common Crypto Module-Extended algorithm implementation
    - AES (encrypt/decrypt) KATs
    - SHA-1 KAT
    - SHA-256 KAT
    - SHA-384 KAT
    - SHA-512 KAT
    - Triple-DES (encrypt/decrypt) KATs

  - POSTs - IOS Common Crypto Module-Extended2 algorithm implementation
    - Triple-DES (encrypt/decrypt) KATs

- Conditional tests
    - Pairwise consistency test for RSA Sign/Verify
    - Pairwise consistency test for RSA Key Wrapping
    - Pairwise consistency test for ECDSA
    - Continuous random number generation test for approved DRBG

The module inhibits all access to cryptographic algorithms during initialization and self-tests due to the process architecture in use. Additionally, the power-on self-tests are performed after the cryptographic systems are initialized but prior to the underlying OS initialization of external interfaces; this prevents the security appliances from passing any data before completing self-tests and entering FIPS mode. In the event of a power-on self-test failure, the cryptographic module will force the IOS platform to reload and reinitialize the operating system and cryptographic module.  This operation ensures no cryptographic algorithms can be accessed unless all power on self-tests are successful.

In addition to the automatic operation at cryptographic module initialization time, self-tests can also be initiated on demand by the Crypto Officer or User by issuing the operating system command which invokes the module’s “crypto_engine_nist_run_self_tests() function.”

## 2.8 AES GCM IV Generation

The module's AES-GCM implementation conforms to IG A.5, scenario #3: When operating in a FIPS approved mode of operation, the IV is constructed in its entirety internally deterministically, consisting of 96 bits as specified in SP800-38D, section 8.2.1.[3]

# 3 Secure Operation of the IC2M

The module is completely and permanently embedded into the greater IOS operating system. There are no installation considerations besides the typical loading of the larger IOS system. That is, the imbedded IC2M firmware module cannot be modified, replaced or upgraded except by loading a new IOS version in its entirety.

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 following policy must always be followed in order to achieve a FIPS 140-2 mode of operation:

• Calling the function ic2m_init() initializes the cryptographic module and place the module in the FIPS-approved mode of operation. During system bring up the O/S(IOS) will call init functions for each of its subsystems, in this case ic2m_init(). ic2m_init() is the default entry point for IC2M. ic2m_init() then calls the POST and does not return to the O/S until the POST completes. No other tasks are being run at this time, so no data is passed.

• Only FIPS approved or allowed algorithms and key sizes shall be used. Please refer to section 2.6 for more information.

• In the event that the Module power is lost and restored, a new key for use with the AES GCM encryption/decryption shall be established.

• In accordance with CMVP IG A.13, when operating in a FIPS approved mode of operation, the same Triple-DES key shall not be used to encrypt more than 2^20 64-bit data blocks. Each of TLS, SSH and IPSec protocols governs the generation of the respective Triple- DES keys. Refer to RFC 5246 (TLS), RFC 4253 (SSH) and RFC 6071 (IPSec) for details relevant to the generation of the individual Triple-DES encryption keys. The user is responsible for ensuring the module limits the number of encryptions with the same key to 2^20.

Upon power-up the Module, the module will run its power-up self-tests. Successful completion of the power-up self-tests indicates the module has passed the self-tests and is ready within the IOS. If an error occurs during the self-test the module outputs the following message:

---

[3] The module uses 32 bits of the IV field as a name and uses 64 bits as a deterministic non-repetitive counter for combined IV length of 96 bits.

requesting a reload of the OS.  %CRYPTO-0-SELF_TEST_FAILURE: Encryption self-test failed **(**<*failing test description*>**)** where <*failing test description*> identifies the name of the self-test that failed.

## 4 Considerations for using IC2M

The IC2M cryptographic module will function the same way and provide the same security services on any Cisco product using IOS or IOS-XE software, including:

- Cisco 1000 Series Aggregated Services Routers (ASR)
- Cisco 900 Series Aggregated Services Routers (ASR)
- Cisco 901 Series Aggregated Services Routers (ASR)
- Cisco 800 Series Integrated Services Routers (ISR)
- Cisco 1900 Series Integrated Services Routers (ISR)
- Cisco 2900 Series Integrated Services Routers (ISR)
- Cisco 3900 Series Integrated Services Routers (ISR)
- Cisco 4000 Series Integrated Services Routers (ISR)
- Cisco 5900 Series Embedded Services Routers (ESR)
- Cisco Catalyst 2960 Series Switches
- Cisco Catalyst 3560 Series Switches
- Cisco Catalyst 3650 Series Switches
- Cisco Catalyst 3750 Series Switches
- Cisco Catalyst 3850 Series Switches
- Cisco Catalyst 4500 Series Switches
- Cisco IE 2000 Series Switches
- Cisco IE 3000 Series Switches
- Cisco IE 4000 Series Switches
- Cisco IE 5000 Series Switches
- Cisco Aironet 3700 Series Access Points
- Cisco Aironet 2700 Series Access Points
- Cisco Aironet 1700 Series Access Points
- Cisco Aironet 3600 Series Access Points
- Cisco Aironet 2600 Series Access Points
- Cisco Aironet 1600 Series Access Points
- Cisco Aironet 700w Series Access Points
- Cisco Aironet 700 Series Access Points
- VG300 Series Analog Voice Gateway

 