Cisco Systems, Inc.

# FIPS 140-3 and ISO/IEC 19790 Non-Proprietary Security Policy

For

# Linux Kernel FIPS Object Module Cryptographic Module

Last Updated: May 31, 2024, Version 1.1

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

## Table of Content

1 General ........ 3
2 Cryptographic Module Specification ........ 3
3 Cryptographic Module Interfaces ........ 9
4 Roles, Services, and Authentication ........ 10
5 Software/Firmware Security ........ 13
6 Operational Environment ........ 14
7 Physical Security ........ 14
8 Non-Invasive Security ........ 14
9 Sensitive Security Parameters Management ........ 14
10 Self-Tests ........ 17
11 Life-Cycle Assurance ........ 20
12 Mitigation of Other Attacks ........ 20

## List of Tables

Table 1 - Security Levels ........ 3
Table 2 - Tested Operational Environments ........ 4
Table 3 - Vendor Affirmed Operational Environments ........ 5
Table 4 - Approved Algorithms ........ 8
Table 5 - Ports and Interfaces ........ 10
Table 6 - Roles, Service Commands, Input and Output ........ 10
Table 7 - Approved Services ........ 13
Table 8 - SSPs ........ 16
Table 9 - Non-Deterministic Random Number Generation Specification ........ 17

## List of Figures

Figure 1 - UCS C220 M5 Front View ........ 4
Figure 2 - UCS C220 M5 Rear View ........ 4
Figure 3 - MX68CW Front View ........ 4
Figure 4 - MX68CW Rear View ........ 4
Figure 5 - MX68CW Side View ........ 5
Figure 6 - Intel Xeon Gold 6138 ........ 5
Figure 7 - ARMv8 Cortex-A53 ........ 5
Figure 8 - Block Diagram ........ 6

# 1 General

This is Cisco Systems, Inc.'s non-proprietary security policy for Linux Kernel FIPS Object Module Cryptographic Module (hereinafter referred to as KFOM or Module) firmware version 1.0. The following details how this module meets the security requirements of FIPS 140-3, SP 800-140 and ISO/IEC 19790 for a Security Level 1 Firmware hybrid cryptographic module.

The security requirements cover areas related to the design and implementation of a cryptographic module. These areas include cryptographic module specification; cryptographic module interfaces; roles, services, and authentication; software/firmware security; operational environment; physical security; non-invasive security; sensitive security parameter management; self-tests; life-cycle assurance; and mitigation of other attacks. The following table indicates the actual security levels for each area of the cryptographic module.

| ISO/IEC 24759 Section 6. | FIPS 140-3 Section Title | Security Level |
|---|---|---|
| 1 | General | 1 |
| 2 | Cryptographic Module Specification | 1 |
| 3 | Cryptographic Module Interfaces | 1 |
| 4 | Roles, Services, and Authentication | 1 |
| 5 | Software/Firmware Security | 1 |
| 6 | Operational Environment | 1 |
| 7 | Physical Security | 1 |
| 8 | Non-Invasive Security | N/A |
| 9 | Sensitive Security Parameter Management | 1 |
| 10 | Self-Tests | 1 |
| 11 | Life-Cycle Assurance | 1 |
| 12 | Mitigation of Other Attacks | N/A |
| Overall Level | | 1 |

**Table 1 - Security Levels**

# 2 Cryptographic Module Specification

The Cisco Linux Kernel FIPS Object Module (KFOM) Cryptographic Module is a firmware hybrid cryptographic library in a multi-chip standalone embodiment that allows for Linux kernel applications to use approved algorithms. It does not implement any security protocols nor create any cryptographic keys. Instead, it only provides Linux kernel applications access to approved algorithms.

The module is intended to run on the UCS C220 M5 and MX68W host platforms or any general-purpose computer, so the physical perimeter of the module is the tested platforms. The cryptographic module comprises the Cisco Linux Kernel FIPS Object Module (KFOM) Cryptographic Module (Firmware Version: 1.0) which is a kernel object file linux_kfom_1_0_0.ko and the processor (Hardware Version: ARMv8 Cortex-A53, Intel Xeon Gold 6138) (only for algorithm acceleration) which only operates in the approved mode of operation which is set at manufacture. The module is validated according to FIPS 140-3 at overall security level 1. Please refer to Table 1 above for the individual areas.

Cisco UCS C220 M5 unifies computing, networking, management, virtualization, and storage access into a single integrated architecture, enabling end-to-end server visibility, management, and control in both bare metal and virtualized environments.

The module has been tested on the following Operational Environments.

| # | Operating System | Hardware Platform | Processor | PAA/Acceleration |
|---|---|---|---|---|
| 1 | Linux 4.9 | MX68CW | ARMv8 Cortex-A53 | With PAA |
| 2 | Ubuntu 18.04 | UCS C220 M5 | Intel Xeon Gold 6138 | With PAA |

**Table 2 - Tested Operational Environments**

**Figure 1 - UCS C220 M5 Front View**

**Figure 2 - UCS C220 M5 Rear View**

**Figure 3 - MX68CW Front View**

**Figure 4 - MX68CW Rear View**

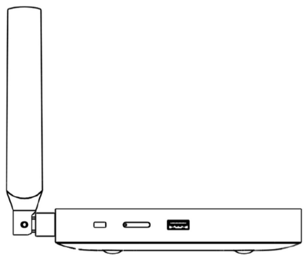
**Figure 5 - MX68CW Side View**

Please note that Figures 1-5 are the tested platforms and not the module itself.

**Figure 6 - Intel Xeon Gold 6138**

**Figure 7 - ARMv8 Cortex-A53**

The following table lists the Vendor affirmed operational environment:

| # | Operating System | Hardware Platform |
|---|---|---|
| 1 | Linux 4 | Z3 |
| 2 | Linux 4 | Z3C |
| 3 | Linux 4 | MX67C |
| 4 | Linux 4 | MX67W |
| 5 | Linux 4 | MX75 |
| 6 | Linux 4 | MX85 |
| 7 | Linux 4 | MX95 |
| 8 | Linux 4 | MX105 |
| 9 | Linux 4 | MX250 |
| 10 | Linux 4 | MX450 |

**Table 3 - Vendor Affirmed Operational Environments**

The cryptographic module maintains validation compliance when operating on the operating system/mode specified in Table 2 above and on the validation certificate as well as for those specified in Table 3 above (vendor affirmed). For the latter, the CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when ported to an operational environment which is not listed on the validation certificate.

Cryptographic Boundary

The KFOM cryptographic module (red box) is a non-modifiable, multi-chip standalone firmware hybrid cryptographic module providing cryptographic support to the Kernel which takes data in and out from the host application via the API (libkcapi) feeding into the kernel. All processing is done on the listed processors in Table 2 above. The KFOM performs no communications other than with the consuming application. The block diagram below shows the Tested Operational Environment's Physical Perimeter (TOEPP) being defined as the physical perimeter of the tested platform enclosure around which everything runs. The cryptographic boundary is the KFOM (red box) and the processors (ARMv8 Cortex-A53, Intel Xeon Gold 6138) with PAA.

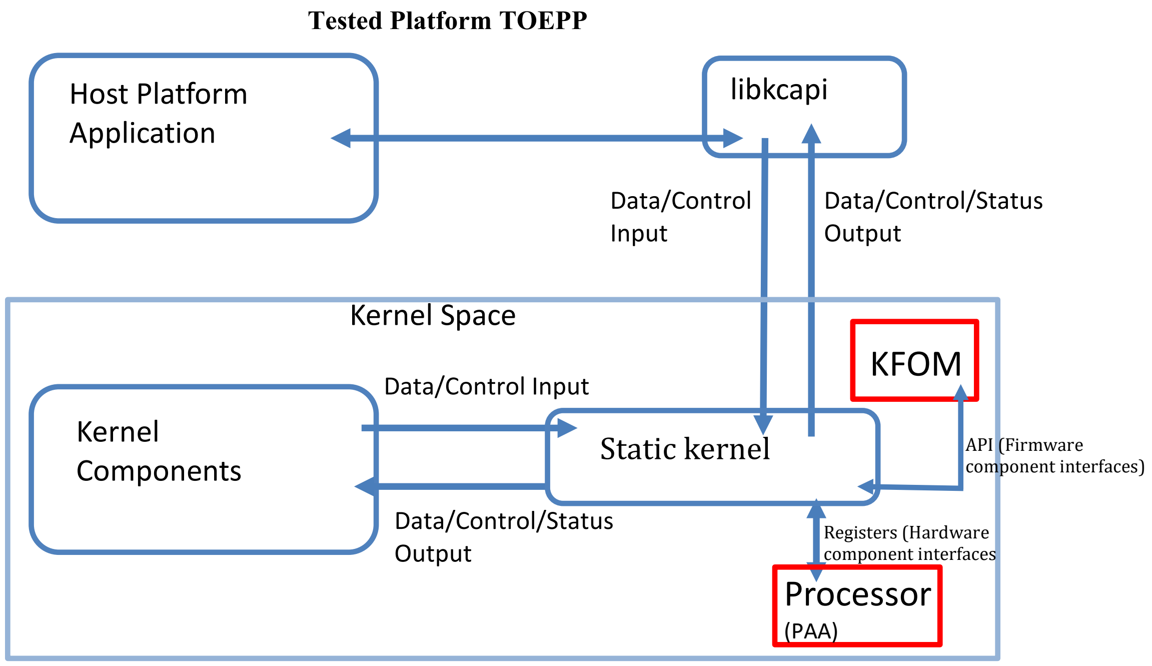
Tested Platform TOEPP
Host Platform Application
libkcapi
Data/Control Input
Data/Control/Status Output
Kernel Space
KFOM
Data/Control Input
Kernel Components
Static kernel
API (Firmware component interfaces)
Data/Control/Status Output
Registers (Hardware component interfaces
Processor (PAA)

**Figure 8 - Block Diagram**

Firmware component interfaces include: Data Input, Data Output, Control Input and Status Output interfaces.

Hardware component interfaces include: Input, Output, Control Input and Status registers.
Modes of Operation

By design, the module is only able to support approved mode of operation. Once the module is configured in the approved mode of operation by following the steps in Section 11 of this document, the module will be ready for approved mode of operation. The module doesn't claim the implementation of a degraded mode operation.

Approved security functions:

| CAVP Cert | Algorithm and Standard | Mode/Method | Description / Key Size(s) / Key Strength(s) | Use / Function |
|---|---|---|---|---|
| A1182 and A1185 | AES [FIPS 197, SP800-38A] | CBC, ECB, CTR | Key length: 128, 192 and 256 bits | Block cipher providing encryption/decryption with data confidentiality from the modes of operation |
| A1182 and A1185 | AES [FIPS 197, SP800-38C] | CCM | Key length: 128, 192 and 256 bits | Block cipher providing confidentiality an authentication through Counter with Cipher Block Chaining-Message Authentication Code |
| A1182 and A1185 | AES [FIPS 197, SP800-38B] | CMAC | Key length: 128, 192 and 256 bits | A cipher (AES) based MAC providing authentication, encryption and decryption |
| A1182 and A1185 | AES [FIPS 197, SP800-38D] | GCM, GMAC | Key length: 128, 192 and 256 bits | Authentication and encryption. Providing confidentiality of data through authentication, encryption and decryption |
| A1182 and A1185 | AES [FIPS 197, SP800-38A(addendum)] | CBC-CS3 | Key length: 128, 192 and 256 bits | Block cipher providing encryption/decryption with variant on padding |
| A1182 and A1185 | AES [FIPS 197, SP800-38E] | XTS | Key length: 128 and 256 bits | Authenticated symmetric encryption and decryption; XTS in approved mode can only be used for cryptographic |

| | | | | |
|---|---|---|---|---|
| | | | | protection of data on storage devices |
| A1182 and A1185 | SHS [FIPS 180-4] | SHA-1, SHA2-224/256/384/512 | N/A | Message digest; In approved mode, SHA-1 can only be used for non-digital-signature and legacy use. All other SHAs acceptable for hash functions applications |
| A1182 and A1185 | HMAC [FIPS 198-1] | HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384 and HMAC-SHA2-512 | Key length: 112 bits or greater | Integrity based on secret key. Using standard SHA HASH with secret key for calculations and verification |
| A1182 and A1185 | DRBG-CTR [SP800-90Arev1] | AES-128, AES-192, AES-256 Derivation Function Enabled; Prediction Resistance: Yes | N/A | Deterministic Random Bit Generators (DRBG); uses an algorithm to produce random output |
| A1182 and A1185 | DRBG_HASH [SP800-90Arev1] | SHA-1, SHA2-224/256/384/512 | N/A | Deterministic Random Bit Generators (DRBG); uses an algorithm to produce random output |
| A1182 and A1185 | DRBG_HMAC [SP800-90Arev1] | HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384 and HMAC-SHA2-512 | N/A | Deterministic Random Bit Generators (DRBG); uses an algorithm to produce random output |

**Table 4 - Approved Algorithms**

Notes:

- Algorithm Cert. #A1182 was tested for the OE with PAA
- Algorithm Cert. #A1185 was tested for the OE with PAA
- There are some algorithm modes that were tested but not implemented by the module. Only the algorithms, modes, and key sizes that are implemented by the module are shown in this table.
- Use of external IV with GCM is exclusively permitted for decryption operations or where the GCM is used to support the protocol specific implementation used to protect connections to the calling applications. The GCM IV is generated per FIPS 140-3 IG C.H technique #3. The counter portion of the IV is set by the module within its cryptographic boundary. When the IV exhausts the maximum number of possible values for a given session key, the first party, client or server, to encounter this condition will trigger a handshake to establish a new encryption key. In case the module's power is lost and then restored, a new key for use with the AES GCM encryption/decryption shall be established.

- The check for Key_1 ≠ Key_2 is done before using the keys in the XTS-AES algorithm to process data and is in accordance with IG C.I requirements.

## 3 Cryptographic Module Interfaces

The module's physical perimeter encompasses the case of the tested platform mentioned in Table 2. The module provides its logical interfaces via Application Programming Interface (API) calls. The logical interfaces provided by the module are mapped onto the FIPS 140-3 interfaces (data input, data output, control input, control output and status output) as follows.

| **Physical Port** | **Logical Interface** | **Data that passes over port/interface** |
|---|---|---|
| Input registers [Hardware component only] | N/A | • Data to be encrypted, decrypted or hashed<br>• Keys to be used in cryptographic services |
| Output registers [Hardware component only] | N/A | • Data that has been encrypted or decrypted |
| Control registers [Hardware component only] | N/A | • Instructions to invoke the PAA operation |
| Status registers [Hardware component only] | N/A | • Return values |
| N/A | Data Input Interface [Firmware component only] | Arguments for an API call that provide the data to be used or processed by the module:<br><br>• Data to be encrypted, decrypted or hashed<br>• Keys to be used in cryptographic services<br>• Random seed material for the module's DRBG |
| N/A | Data Output Interface [Firmware component only] | Arguments output from an API call:<br><br>• Data that has been encrypted or decrypted<br>• Hashes |
| N/A | Control Input Interface [Firmware component only] | Arguments for an API call used to control and configure module operation:<br>• Modes, key sizes, etc. used with cryptographic services |
| N/A | Status Output Interface [Firmware component only] | • Return values<br>• Status information regarding the module<br>• Status information regarding the invoked service/operation |

**Table 5 - Ports and Interfaces**

Please note that the module does not support control output interface and is not applicable for this module.

## 4 Roles, Services, and Authentication

The module supports Crypto Officer (CO) role. The cryptographic module does not provide any authentication methods. The module does not allow concurrent operators. The Crypto Officer is implicitly assumed based on the service requested. The module provides the following services to the Crypto Officer.

| Role | Service | Input | Output |
|---|---|---|---|
| Crypto Officer | Show Status | API call parameters | Successful output of module's name and version denotes that the module is active |
| Crypto Officer | Perform Self-Tests | Automatically executed, or host platform's shutdown command | Output on each algorithm running self-test and pass or fail |
| Crypto Officer | Show Version | API call parameters | Output the version |
| Crypto Officer | Configure Symmetric Encryption | API call parameters, encryption key, plaintext data | LINUX KERNEL FOM followed by the encryption in use and ciphertext data |
| Crypto Officer | Configure Symmetric Decryption | API call parameters, decryption key, ciphertext data | LINUX KERNEL FOM followed by the decryption in use and plaintext data |
| Crypto Officer | Configure Keyed Hash | API call parameters, authentication key, message | LINUX KERNEL FOM followed by the hash in use and keyed hash output |
| Crypto Officer | Configure Message Digest | API call parameters, message to be hashed | LINUX KERNEL FOM followed by the digest in use and hashed output |
| Crypto Officer | Configure Random Number Generation | API call parameters | LINUX KERNEL FOM followed by the random strings in use |
| Crypto Officer | Perform Zeroisation | API call parameters (for temporary SSPs) and host platform's shutdown command | N/A |

**Table 6 - Roles, Service Commands, Input and Output**

The table below lists all approved services that can be used in the approved mode of operation. The abbreviations of the access rights to keys and SSPs have the following interpretation:

**G = Generate**: The module generates or derives the SSP.

**R = Read**: The SSP is read from the module (e.g. the SSP is output).
**W = Write**: The SSP is updated, imported, or written to the module.
**E = Execute**: The module uses the SSP in performing a cryptographic operation.
**Z = Zeroise**: The module zeroises the SSP.
**N/A** = The service does not access any SSP during its operation.

| Service | Description | Approved Security Functions | Keys and/or SSPs | Roles | Access Rights to Keys and/or SSPs | Indicator[1] |
|---|---|---|---|---|---|---|
| Show Status | Provide Module's status | N/A | N/A | Crypto Officer | N/A | Global indicator API output as designed by HOST system using the KFOM and output of `echo $?` |
| Perform Self-Tests | Execute the CAST and Health tests | None | AES Key, Authentication, DRBG entropy input, DRBG Seed, DRBG V and C, DRBG Key | Crypto Officer | E | Global indicator API output as designed by HOST system using the KFOM and output of `echo $?` |
| Show Version | Provide module's name and version information | N/A | N/A | Crypto Officer | N/A | Global indicator API output as designed by HOST system using the KFOM and output of `echo $?` |
| Configure Symmetric Encryption | Configure AES algorithm | AES CBC, ECB, CTR, CCM, GCM, CBC-CS3, XTS 128, 192 (except XTS), 256 bits | AES Key | Crypto Officer | W,E,Z | Global indicator API output as designed by HOST system using the KFOM and output of `echo $?` |
| | | | AES GCM IV | Crypto Officer | W, G, E, Z | |
| Configure Symmetric Decryption | Configure AES algorithm | AES CBC, ECB, CTR, CCM, GCM, CBC-CS3, XTS 128, 192 (except XTS), 256 bits | AES Key | Crypto Officer | W,E,Z | Global indicator API output as designed by HOST system using the KFOM and |
| | | | AES GCM IV | Crypto Officer | W, G, E, Z | |

[1] If `echo $?` returns 0, the approved service executed successfully. Any other value = not successful.

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
|  |  |  |  |  |  | output of `echo $?` |
| Configure Keyed Hash | Configure HMAC usage | HMAC SHA-1, HMAC-SHA2-224/256/384/512 | Authentication | Crypto Officer | W,E,Z | Global indicator API output as designed by HOST system using the KFOM and output of `echo $?` |
| Configure Message Digest | Configure SHS usage | SHA-1, SHA2-224/256/384/512 | N/A | Crypto Officer | N/A | Global indicator API output as designed by HOST system using the KFOM and output of `echo $?` |
| Configure Random Number Generation | Configure DRBG usage | DRBG (Hash, HMAC or AES CTR) | DRBG entropy input | Crypto Officer | W, E, Z | Global indicator API output as designed by HOST system using the KFOM and output of `echo $?` |
|  |  |  | DRBG Seed, DRBG V and C, DRBG Key | Crypto Officer | G, E, Z |  |
| Perform Zeroization | Perform Zeroization | N/A | All SSPs | Crypto Officer | Z | None |

**Table 7 - Approved Services**

## 5 Software/Firmware Security

**Integrity Technique**
The Module is provided in the form of binary executable code. To ensure firmware security, the Module is protected by HMAC-SHA2-512 (HMAC Certs. #A1182 or #A1185) algorithm. At Module's initialization, the integrity of the runtime executable is verified using a HMAC-SHA2-512 digest which is compared to a value computed at build time. If at the load time the MAC does not match the stored, known MAC value, the Module would enter an Error state with all crypto functionality inhibited.

**Integrity Test On-Demand**
The integrity test is performed as part of the Pre-Operational Self-Tests. It is automatically executed at power-on. The operator can initiate the integrity test on demand by power cycling the host platform. The module is a kernel object file linux_kfom_1_0_0.ko

## 6 Operational Environment

The module is operated in a non-modifiable operational environment per FIPS 140-3 level 1 specifications. The module is installed within the product for use with the kernel, and the linux kernel cannot be modified. The operating systems and tested platforms can be found in Table 2. The application that makes calls to the module is the single user of the module, even when the application is serving multiple clients. The module's firmware version running on each tested platform is 1.0.

## 7 Physical Security

Per ISO/IEC 19790 and FIPS 140-3 classification, this is a multi-chip standalone cryptographic module. KFOM 1.0 is a firmware hybrid module and runs on a production grade chassis.

## 8 Non-Invasive Security

This section is not applicable as the cryptographic module does not implement any non-invasive attack mitigation techniques.

## 9 Sensitive Security Parameters Management

The following table summarizes the Sensitive Security Parameters (SSPs) that are used by the cryptographic services implemented in the module.

| Key/SSP Name/ Type | Strength | Security Function and Cert. Number | Generation | Import/ Export | Establish-ment | Storage | Zeroisation | Use & Related Keys |
|---|---|---|---|---|---|---|---|---|
| DRBG Entropy Input | >112 bits | N/A | Obtained from the Entropy Source within TOEPP (GPS INT Pathways) | Import to the module via Module's API Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage | Zeroized when the tested platform is powered down | Random number generation |
| DRBG Output | 128 to 512 bits | SP800-90Arev1 Counter DRBG, Hash DRBG, HMAC DRBG Cert A1182/ A1185 | Generated using SP800-90Arev1 DRBG | Import: No Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage | crypto_free_rng() or power cycle the device | Random bits provided for the calling application |
| DRBG Seed | 384 bits | SP800-90Arev1 Counter DRBG, Hash DRBG, | Generated using DRBG derivation function that includes the | Import: No Export: No | N/A | N/A: The module does not provide persistent | crypto_free_rng() or power cycle the device | Internal state of the DRBG |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | HMAC DRBG Cert A1182/ A1185 | entropy input | | | keys/SSPs storage | | |
| DRBG V | 128<br><br>440/888 bits<br><br>160/256/384/512 bits) | SP800-90Arev1 Counter DRBG, Hash DRBG, HMAC DRBG Cert A1182/ A1185 | Generated first during DRBG instantiation and then subsequently updated using the DRBG update function | Import: No<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage | crypto_free_rng() or power cycle the device | Internal state of the DRBG |
| DRBG C | 128/192/256<br><br>440/888 bits | SP800-90Arev1 Counter DRBG, Hash DRBG Cert A1182/ A1185 | Generated first during DRBG instantiation and then subsequently updated using the DRBG update function | Import: No<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage | crypto_free_rng() or power cycle the device | Internal state of the DRBG |
| DRBG Key | 128/192/256 bits<br><br>160/256/384/512 bits | SP800-90Arev1 Counter DRBG, HMAC DRBG Cert A1182/ A1185 | Established per SP 800-90Arev1 Counter DRBG and HMAC DRBG | Import: Yes<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage | crypto_free_rng() or power cycle the device | Internal state of the DRBG |
| AES Key | 128,192,256 bits | AES (GCM, GMAC, XTS, CMAC, CCM, CTR, CBC, ECB) Cert A1182/ A1185 | Generated externally and passed into the module | Import: Yes<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage | crypto_free_cipher() crypto_free_ablkcipher() crypto_free_blkcipher() crypto_free_skcipher() crypto_free_aead() or power cycle the device | AES session key |

|  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|
| AES GCM IV | 96-bit | AES GCM Cert A1182/ A1185 | Generated externally and passed into the module<br><br>Generated internally (FIPS 140-3 IG C.H #3) | Import: Yes<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage | Power cycle the device | AES GCM session key and decryption |
| Authentication | 160-512 bits | HMAC (SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512) Cert A1182/ A1185 | Generated externally and passed into the module | Import: Yes<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage | crypto_free_shash() crypto_free_ahash() or power cycle the device | Integrity assurance |
| Firmware Integrity Key (not a SSP) | 160 bits | HMAC-SHA2-512 Cert A1182/ A1185 | Pre-loaded at the factory (in the module's binary) | Import: No<br><br>Export: No | N/A | Stored in the module binary computed during build | This key is used for firmware integrity test and not subject to key zeroization requirements according to FIPS140-3 IG 9.7.B | Used for firmware integrity test. This is not an SSP |
| Integrity test calculation value | N/A | HMAC-SHA2-512 Cert A1182/ A1185 | Calculated during integrity test | Import: No<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage | memzero_explicit() call before exiting the integrity test function | Calculated during integrity test |

**Table 8 - SSPs**

Even though the module does implement an approved random number generator, it is not used by the module for the generation of the cryptographic keys. All Cryptographic keys are externally generated and imported to the working space assigned by the kernel. The module uses approved DRBG for the generation of random strings and passes them to the calling application only upon their request. The cryptographic module is passed a pointer to the cryptographic keys as API parameters, associated by memory location. The application calling the cryptographic module passes keys in plaintext within the physical perimeter. The module does not perform storage of keys. All SSPs can be zeroized by power cycling the host.

| Entropy sources | Minimum number of bits of entropy | Details |
|---|---|---|
| Entropy within the TOEPP was passively load into the Module to seed the 800-90Arev1 DRBG by the operating system | At least 112 bits | The entropy and seeding material for the entropy source is provided to it by the external calling application (and not by the module) which is outside the module's boundary. The minimum effective strength of the SP 800-90Arev1 DRBG seed is required to be at least 112 bits when used in the approved mode of operation, therefore the minimum number of bits of entropy requested when the calling application makes a call to the SP 800-90Arev1 DRBG is 112. Hence the caveat "No assurance of the minimum strength of generated SSPs (e.g., keys)" is applicable.<br><br>The module does not generate any cryptographic keys but the SP800-90Arev1 DRBGs are used to provide the random strings requested by the calling application. These random strings are not used by the module and is passed to the calling application when requested.<br><br>The module users (the external calling applications) shall use entropy source which meets the security strength required for the random number generation mechanism as shown in SP 800-90Arev1 based on Hash DRBG, HMAC DRBG, and Counter DRBG. |

**Table 9 - Non-Deterministic Random Number Generation Specification**

## 10 Self-Tests

When the Linux kernel is setup properly the self-test solution is that only the kernel module, which has been signed and verified can load in cryptographic algorithms. Once the KFOM is loaded, first the self-test for the algorithm used in the firmware integrity test is run and then the actual firmware integrity test of the module runs for the KFOM, then all the self-tests for the algorithms are run. If a self-test fails, the result from the Linux Crypto Test Manager is a failure. This means the algorithm will not have the self-test flag set. The presence of self-test flag allows the Linux Crypto Framework to utilize the algorithm.

A successful log message is provided by the module after the successful completion of the self-tests.

If an error occurs to a valid approved algorithm during the self-test, the module enters hard error state, and the Linux kernel will print an error message to the console and data output from the data output interface is inhibited. This results in the system shutting down.

**Pre-Operational Self-tests**

- Pre-operational firmware integrity test
  - Firmware Integrity Test (HMAC SHA2-512)

The Module conducts HMAC-SHA2-512 KAT self-test before the integrity test is performed.

**Conditional Self-Test**

- Conditional Cryptographic Algorithm Self-Tests (CASTs)
  - AES CBC 128-bit Encrypt KAT
  - AES CBC 192-bit Encrypt KAT
  - AES CBC 256-bit Encrypt KAT
  - AES CBC 128-bit Decrypt KAT
  - AES CBC 192-bit Decrypt KAT
  - AES CBC 256-bit Decrypt KAT
  - AES CTR 128-bit Encrypt KAT
  - AES CTR 192-bit Encrypt KAT
  - AES CTR 256-bit Encrypt KAT
  - AES CTR 128-bit Decrypt KAT
  - AES CTR 192-bit Decrypt KAT
  - AES CTR 256-bit Decrypt KAT
  - AES ECB 128-bit Encrypt KAT
  - AES ECB 192-bit Encrypt KAT
  - AES ECB 256-bit Encrypt KAT
  - AES ECB 128-bit Decrypt KAT
  - AES ECB 192-bit Decrypt KAT
  - AES ECB 256-bit Decrypt KAT
  - AES CCM 128-bit Encrypt KAT
  - AES CCM 192-bit Encrypt KAT
  - AES CCM 256-bit Encrypt KAT
  - AES CCM 128-bit Decrypt KAT
  - AES CCM 192-bit Decrypt KAT
  - AES CCM 256-bit Decrypt KAT
  - AES XTS 128-bit Encrypt KAT
  - AES XTS 256-bit Encrypt KAT
  - AES XTS 128-bit Decrypt KAT
  - AES XTS 256-bit Decrypt KAT
  - AES CBC CS 128-bit Encrypt KAT
  - AES CBC CS 128-bit Decrypt KAT
  - AES GCM 128-bit Encrypt KAT
  - AES GCM 192-bit Encrypt KAT
  - AES GCM 256-bit Encrypt KAT
  - AES GCM 128-bit Decrypt KAT
  - AES GCM 192-bit Decrypt KAT
  - AES GCM 256-bit Decrypt KAT
  - AES CMAC 128-bit Encrypt KAT
  - AES CMAC 128-bit Decrypt KAT
  - DRBG AES CTR 128-bit KAT (Health Tests: Generate, Reseed, Instantiate functions per Section 11.3 of SP 800-90Arev1)
  - DRBG AES CTR 192-bit KAT(Health Tests: Generate, Reseed, Instantiate functions

per Section 11.3 of SP 800-90Arev1)

- DRBG AES CTR 256-bit KAT(Health Tests: Generate, Reseed, Instantiate functions per Section 11.3 of SP 800-90Arev1)
- DRBG HMAC SHA-1 KAT(Health Tests: Generate, Reseed, Instantiate functions per Section 11.3 of SP 800-90Arev1)
- DRBG HMAC SHA2-256 KAT(Health Tests: Generate, Reseed, Instantiate functions per Section 11.3 of SP 800-90Arev1)
- DRBG HMAC SHA2-384 KAT(Health Tests: Generate, Reseed, Instantiate functions per Section 11.3 of SP 800-90Arev1)
- DRBG HMAC SHA2-512 KAT(Health Tests: Generate, Reseed, Instantiate functions per Section 11.3 of SP 800-90Arev1)
- DRBG Hash SHA-1 KAT(Health Tests: Generate, Reseed, Instantiate functions per Section 11.3 of SP 800-90Arev1)
- DRBG Hash SHA2-256 KAT(Health Tests: Generate, Reseed, Instantiate functions per Section 11.3 of SP 800-90Arev1)
- DRBG Hash SHA2-384 KAT(Health Tests: Generate, Reseed, Instantiate functions per Section 11.3 of SP 800-90Arev1)
- DRBG Hash SHA2-512 KAT(Health Tests: Generate, Reseed, Instantiate functions per Section 11.3 of SP 800-90Arev1)
- HMAC-SHA-1 KAT
- HMAC-SHA2-224 KAT
- HMAC-SHA2-256 KAT
- HMAC-SHA2-384 KAT
- HMAC-SHA2-512 KAT
- SHA-1 KAT
- SHA2-224 KAT
- SHA2-256 KAT
- SHA2-384 KAT
- SHA2-512 KAT

**Periodic/On-Demand Self-Tests**

The module performs on-demand self-tests initiated by the operator, by powering off and powering the module back on through power cycling the host. The full suite of self-tests is then executed. The same procedure may be employed by the operator to perform periodic self-tests.

**Self-Test Failure Handling**

KFOM reaches the critical error state when any self-test fails. This failure results in kernel panic. Upon test failure, the module will set an internal flag and enter a critical error state, and a message will be logged to dmesg. In this state, the module will no longer perform cryptographic services or output data over the data output interface.

To recover, the module must be re-instantiated by the calling application (through power cycling the host platform). If the pre-operational self-tests complete successfully, then the module can resume normal operations. If the module continues to experience self-test failures after reinitializing, then the module will not be able to resume normal operation.

## 11 Life-Cycle Assurance

Secure Operations

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 firmware and hardware. The module functions entirely within the process space of the process that invokes it, and thus the module runs on a single user mode of operation.

The module is copied into the tested operational environments prior to shipping. The operator needs to load the module per the instructions below, and power cycle the platform. At this point, the pre-operational tests are run, followed by the conditional cryptographic algorithm self-tests. Now KFOM will be in approved mode.

KFOM Loading:

```
insmod linux_kfom_1_0_0.ko config_file=kfom_priorities ko_file=linux_kfom_1_0_0.ko
```

The operator can verify that the module is in approved mode by executing the “Show Status” service command. If the module outputs name and version, it can be considered that the module is running in approved mode.

The guidance document – readme.rtf version 1.0, can be obtained by contacting the vendor using the contact information (posted on the validation certificate).

## 12 Mitigation of Other Attacks

The requirements under INCITS+ISO+IEC 19790+2012[2014], section 7.12 “Mitigation of other attacks”, are not applicable to the module since the module currently doesn’t support any mitigation of other attacks services.