# Nutanix Cryptographic Module for OpenSSL

Version 6.0

# FIPS 140-2 Non-Proprietary Security Policy

Document Version 1.7

May 26, 2022

**Prepared for:**

**Prepared by:**

**Nutanix, Inc.**
1740 Technology Drive, Suite 150
San Jose, CA 95110
nutanix.com
+1 855.NUTANIX

**KeyPair Consulting Inc.**
987 Osos Street
San Luis Obispo, CA 93401
keypair.us
+1 805.316.5024

# Table of Contents

# List of Tables

# List of Figures

# References

| Ref | Full Specification Name |
|---|---|
| *References used in Approved Algorithms Table* | |
| [38A] | NIST SP 800-38A, *Recommendation for Block Cipher Modes of Operation: Methods and Techniques,* Dec 2001 |
| [38B] | NIST SP 800-38B, *Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication*, Oct 2016 |
| [38C] | NIST SP 800-38C, *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality*, Jul 2007 |
| [38D] | NIST SP 800-38D, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC,* Nov 2007 |
| [38E] | NIST SP 800-38E, *Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices*, Jan 2010 |
| [38F] | NIST SP 800-38F, *Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping,* Dec 2012 |
| [56Ar3] | NIST SP 800-56A Rev. 3, *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography,* Apr 2018 |
| [56B] | NIST SP 800-56B, *Recommendation for Pair-Wise Key-Establishment Schemes Using Integer Factorization Cryptography,* Mar 2019 |
| [57P1] | NIST SP 800-57 Part 1 Rev. 5, *Recommendation for Key Management: Part 1 - General*, May 2020 |
| [67] | NIST SP 800-67 Rev. 2, *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Nov 2017* |
| [90A] | NIST SP 800-90A Rev. 1, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators,* Jun 2015 |
| [133] | NIST SP 800-133 Rev. 2, *Recommendation for Cryptographic Key Generation,* Jun 2020 |
| [135] | NIST SP 800-135 Rev. 1, *Recommendation for Existing Application-Specific Key Derivation Functions*, Dec 2011 |
| [180] | FIPS 180-4, *Secure Hash Standard (SHS),* Aug 2015 |
| [186] | FIPS 186-4, *Digital Signature Standard (DSS),* Jul 2013 |
| [197] | FIPS 197, *Advanced Encryption Standard (AES),* Nov 2001 |
| [198] | FIPS 198-1, *The Keyed Hash Message Authentication Code (HMAC),* Jul 2008 |
| *Other References* | |
| [52] | NIST SP 800-52 Rev. 2, *Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations,* Aug 2019 |
| [140] | FIPS 140-2, *Security Requirements for Cryptographic Modules,* Dec 2002 |
| [140DTR] | FIPS 140-2 *Derived Test Requirements,* Jan 2011 |
| [140IG] | *Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program,*Mar 2022 |
| [131A] | NIST SP 800-131A Rev. 2, *Transitioning the Use of Cryptographic Algorithms and Key Lengths*, Mar 2019 |
| [UG] | *Nutanix Cryptographic Modules for OpenSSL and OpenSSH - User Guide*, Feb 2021 |

## Acronyms and Definitions

| Term | Definition |
|---|---|
| AES | Advanced Encryption Standard [197] |
| AES-NI | Advanced Encryption Standard New Instructions |
| API | Application Programming Interface |
| CAVP | Cryptographic Algorithm Validation Program |
| CMVP | Cryptographic Module Validation Program |
| CO | Cryptographic Officer |
| CSP | Critical Security Parameter [140] |
| DRBG | Deterministic Random Number Generator [90A] |
| DSS | Digital Signature Standard [186] |
| DTR | Derived Test Requirements [140DTR] |
| FIPS | Federal Information Processing Standard [140] |
| HMAC | Keyed-Hash Message Authentication Code [198] |
| IG | Implementation Guidance [140IG] |
| KAT | Known Answer Test |
| KDF | Key Derivation Function |
| KW | Key Wrap |
| NDRNG | Non-Deterministic Random Number Generator |
| NIST | National Institute of Standards and Technology |
| OS | Operating System |
| PAA | Processor Algorithm Accelerators |
| PCT | Pairwise Consistency Test |
| PKI | Public Key Infrastructure |
| PSP | Public Security Parameter |
| RSA | Rivest, Shamir, and Adleman Algorithm [186] |
| SHA/SHS | Secure Hash Algorithm/Standard [180] |
| SP | Special Publication |
| SSP | Sensitive Security Parameter - CSPs and PSPs |
| TLS | Transport Layer Security |

# 1   Overview

This document defines the non-proprietary Security Policy for the Nutanix Cryptographic Module for OpenSSL version 6.0; hereafter denoted the Module. The Module is a cryptographic software library, designated as a multi-chip standalone embodiment in [140] terminology, used in Nutanix, Inc. (Nutanix) solutions to provide FIPS 140-2 Approved cryptographic algorithms and TLS secure communication.

The Module meets FIPS 140-2 overall Level 1 requirements, with security levels as follows:

*Table 1: Security Level of Security Requirements*

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

In Table 1 above, [140] Section 4.5 *Physical Security* is not applicable, as permitted by [140IG] 1.16 *Software Module* and [140IG] G.3 *Partial Validations and Not Applicable Areas of FIPS 140-2*.

The Module design corresponds to the Module security rules. Security rules enforced by the Module are described in the appropriate context of this document.

The Module and this Security Policy are aligned with [52] *Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations*, although [52] is not enforced by [140] validation.

The Module operates within a general-purpose computer. Figure 1 depicts the Module operational environment, with the logical boundary highlighted in red inclusive of all Module entry points (API calls), conformant with [140IG] 14.3 *Logical Diagram for Software, Firmware and Hybrid Modules*.
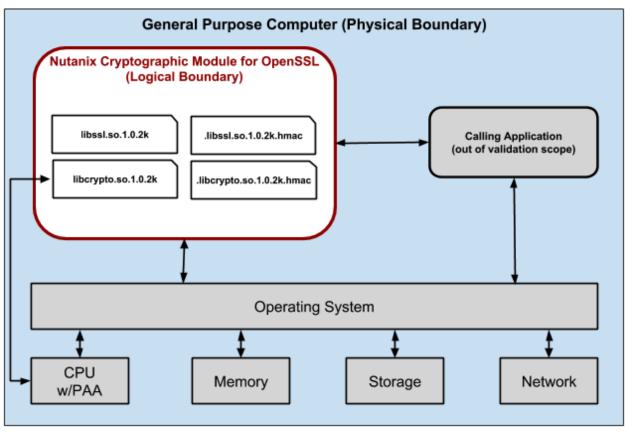
*Figure 1: Module Physical and Logical Boundary*

The Module conforms to [140IG] 1.16 *Software Module*:

- The physical cryptographic boundary is the general-purpose computer which wholly contains the Module and operating system.
- The logical cryptographic boundary is the set of shared library files and associated HMAC files:
  - o   libcrypto.so.1.0.2k
  - o   .libcrypto.so.1.0.2k.hmac
  - o   libssl.so.1.0.2k
  - o   .libssl.so.1.0.2k.hmac
- All components are defined in accordance with [140DTR] AS01.08; no components are excluded from [140] requirements.
- The power-up approved integrity test is performed over all components of the logical boundary.
- Updates to the Module are provided as a complete replacement in accordance with [140IG] 9.7 S*oftware/Firmware Load Test*.
- The Module does not map any interfaces to physical ports. Table 2 defines the Module's [140] logical interfaces.

*Table 2: Ports and Interfaces*

| Description | Logical Interface Type |
|---|---|
| API function calls or configuration files on filesystem | Control input |
| API input parameters, kernel I/O - network or files on filesystem | Data input |
| API return value | Status output |
| API output parameters, kernel I/O - network or files on filesystem | Data output |

Operational testing was performed on the Operating Environments listed in Table 3. The Nutanix product runs an industry-standard hypervisor (ESXi, AHV, or Hyper-V) and the Nutanix Controller VM (CVM). The Nutanix CVM is what runs the Nutanix software and serves all of the I/O operations for the hypervisor and all VMs running on that host. Acropolis Hypervisor (AHV) is the default hypervisor shipped with AOS. AHV is based upon CentOS KVM. Full hardware virtualization is used for guest VMs.

*Table 3: Tested Operating Environments*

| Operating System | Processor | Platform |
|---|---|---|
| CentOS 7.9 on Nutanix Acropolis Hypervisor (AHV) 7.1.1 | Intel Xeon Gold 6234 with PAA | Nutanix NX-3360-G7 (CVM) |
| CentOS 7.9 on Nutanix Acropolis Hypervisor (AHV) 7.1.1 | Intel Xeon Gold 6234 without PAA | Nutanix NX-3360-G7 (CVM) |
| CentOS 7.9 | Intel Xeon Gold 6234 with PAA | Nutanix NX-3360-G7 (AHV) |
| CentOS 7.9 | Intel Xeon Gold 6234 without PAA | Nutanix NX-3360-G7 (AHV) |

The Module conforms to [140IG] 6.1 *Single Operator Mode and Concurrent Operators*. The tested environments place user processes into segregated spaces. A process is logically removed from all other processes by the hardware and Operating System. Since the Module exists inside the process space of the application, this environment implicitly satisfies the requirement for a single user mode.

The Module conforms to [140IG] 1.21 *Processor Algorithm Accelerators (PAA) and Processor Algorithm Implementation (PAI).* The Intel Processor AES-NI functions are identified by [140IG] 1.21 as a known PAA.

## 2   Cryptographic Functionality

The Module implements the FIPS Approved cryptographic functions listed in Table 4. [57P1] notation is used throughout this document to describe key sizes and security strength.

*Table 4: Approved CAVP Validated Cryptographic Functions*

| Cert | Algorithm | Mode | Description | Functions, Caveats |
|---|---|---|---|---|
| A1403 | AES [197] | [38A]: CBC, ECB, CFB-1, CFB-8, CFB-128, CTR, OFB | Key sizes: 128, 192, 256 (bits) | Encryption, Decryption |
| | | [38C]: CCM [38D]: GCM,GMAC | Key sizes: 128, 192, 256 (bits) | Authenticated encryption and decryption; message authentication. |
| | | [38B]: CMAC | Key sizes: 128, 192, 256 (bits) | Generation, Verification |
| | | [38E]: XTS | Key sizes: 128, 256 (bits)[1] | Encryption, Decryption |
| | | [38F]: KW, KWP | Key sizes: 128, 192, 256 (bits) | Key Establishment |
| Vendor Affirmed | CKG [133] | [133] Section 5.1 Asymmetric signature key generation using unmodified DRBG output | | Key Generation |
| | | [133] Section 5.2 Asymmetric key establishment key generation using unmodified DRBG output | | |
| | | [133] Section 6.1 Direct symmetric key generation using unmodified DRBG output | | |
| A1403 | CVL [135] | TLS 1.0/1.1 KDF | | Key derivation |
| | | TLS 1.2 KDF | SHA-256 and SHA-384 | |
| A1403 | CVL [56B] | RSADP $k$ = 2048 | | Key Transport |
| A1403 | DRBG [90A] | CTR_DRBG | AES: 128, 192, 256 (bits) | Random bit generation. The default DRBG (used by the module for secure communications) is an AES-256 CTR_DRBG. |
| | | Hash_DRBG | SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 | |
| | | HMAC_DRBG | SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 | |
| A1403 | DSA [186] | L = 2048 N = 224; SHA-2 (-224, -256, -384, -512) L = 2048 N = 256; SHA-2 (-256, -384, -512) L = 3072 N = 256; SHA-2 (-256, -384, -512) Legacy use (PQG verify and signature verify only): L = 1024 N = 160; SHA-1, SHA-2 (-224, -256, -384, -512) | | Domain parameter (PQG) generate and verify; key generate; signature generate and verify |
| A1403 | ECDSA [186] | P-256, P-384, P-521 with SHA-2 (-224, -256, -384, -512) Legacy use: P-256, P-384, P-521 with SHA-1 (signature verify only) | | Key generate and verify; signature generate and verify |
| A1403 | HMAC [198] | HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 | | Generate and verify |
| A1403 | KAS [56Ar3] (KAS-SSC [56Ar3] | KAS ECC SSC (ephemeralUnified) | P-256, P-384, P-521[2, 3] | Per [140IG] D.8 Scenario X1 path (2), [56Ar3] compliant key agreement scheme where testing is performed separately for the |
| | | KAS FFC SSC (dhEphem) | FB: $L \geq 2048$ $N$ = 224 FC: $L \geq 2048$ $N$ = 256[4] | |

---

[1] The module provides a check and enforcement that key_1 and key_2 are not equal.

[2] See the Security Policy, Section 3, item 13.

[3] EC Diffie-Hellman (key agreement; key establishment methodology provides between 128 and 256 bits of encryption strength)

[4] Diffie-Hellman (key agreement; key establishment methodology provides 112 bits of encryption strength)

| Cert | Algorithm | Mode | Description | Functions, Caveats |
|------|-----------|------|-------------|--------------------|
| with CVL [135]) | | TLS 1.2 KDF | SHA-256 and SHA-384 | shared secret computation and a KDF compliant with [135]. No key confirmation. |
| A1403 | KAS-SSC [56Ar3] | KAS ECC SSC (ephemeralUnified) | P-256, P-384, P-521[2, 3] | Key agreement: Shared secret calculations. |
| | | KAS FFC SSC (dhEphem) | FB: $L \geq 2048$ $N = 224$<br>FC: $L \geq 2048$ $N = 256$[4] | |
| A1403 | KTS [38F] | AES KW, KWP Applicable also to AES-GCM and AES-CBC with HMAC [38F] Section 3.3 | Key sizes: 128, 192, 256 (bits) | Key establishment methodology provides between 128 and 256 bits of encryption strength |
| A1403 | RSA [186] | $k$ = 2048, 3072 (9.31, PKCS1.5, PSS); SHA-2 (-224, -256, -384, -512)<br>Legacy use (signature verify only): $k$ = 1024; SHA-1, SHA-2 (-224, -256, -384, -512) | | Key generate; signature generate and verify |
| A1403 | SHS [180] | SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 | | Message Digest |
| A1403 | Triple-DES [67] | CBC, CFB1, CFB-8, CFB-64, ECB, OFB [38A] | Key size: 168 (3-Key)[5, 6] | Encryption, Decryption |
| | | CMAC [38B] | Key size: 168 (3-key)[8] | Generation, Verification |

The Module conforms to [140IG] D.11 *References to the Support of Industry Protocols* (Resolution scenario 2) by providing CAVP validated [56A] and [56B] components along with the CAVP validated [135] Section 4.2 KDF for TLS. In accordance with [140IG] D.11, the remainder of the TLS protocol has not been reviewed or tested by the CAVP and CMVP.

The Module implements the non-Approved but allowed cryptographic functions listed in Table 5.

*Table 5: Non-Approved but Allowed Cryptographic Functions*

| Cryptographic Function | Description / Usage |
|------------------------|---------------------|
| MD5 | Message Digest used only in the TLS 1.0 / 1.1 KDF per [140IG] 1.23 (No Security Claimed) |
| RSA Key Transport | RSA key transport, using the validated CVL RSADP with PKCS 1.5 padding and $k$ = 2048.[7] |

Entropy is provided by an entropy source outside the Module logical boundary which loads 2048 bits of entropy into the module per invocation. Per [140IG] 7.14 Scenario 2, the following caveat applies:

*No assurance of the minimum strength of generated keys.*

---

[5] The Triple-DES key size is 168 bits; however, the bit strength of the key is 112 bits.

[6] See the Security Policy, Section 3, item 7.

[7] RSA (CVL Cert. #A1403, key wrapping; key establishment methodology provides 112 bits of encryption strength; non-compliant less than 112 bits of encryption strength)

The Module supports the following non-FIPS 140-2 Approved cryptographic functions, which shall not be used in the FIPS Approved mode of operation. Any use of the non-Approved functions will cause the Module to transition to the non-FIPS mode of operation.

*Table 6: Non-Approved Cryptographic Functions*

| Cryptographic Function | Description / Usage |
|---|---|
| Camellia | Encryption, Decryption. |
| CAST | Encryption, Decryption. |
| DES | Encryption, Decryption. |
| Diffie-Hellman | Key Agreement with key sizes not listed in Table 4. |
| DSA | Key Generation, Signature Generation, Signature Verification with $L$ and $N$ not consistent with Table 4. |
| ECDSA | Key Generation, Signature Generation, Signature Verification with secp256k1. The module does not implement any predefined curves other than the NIST curves listed in Table 4 and secp256k1. |
| MD2, MD4 | Message Digest. |
| MD5 | Message Digest used outside of the TLS 1.0 / 1.1 KDF. |
| RC2, RC4, RC5 | Encryption, Decryption. |
| RIPEMD | Message Digest. |
| RNG | ANSI X9.31 AES-128 Random Number Generation. |
| RSA | Key Wrapping (encrypt/decrypt) with $k$ < 2048 bits. |
| RSA | Key Generation, Signature Generation, Signature Verification with $k$ not listed in Table 4. |
| Whirlpool | Message Digest. |

## 3   Modes of Operation, Security Rules and Guidance

The Module supports a FIPS Approved mode of operation and a non-FIPS Approved mode of operation and conforms to [140IG] 1.2 *FIPS Approved Mode of Operation* and 1.19 *non-Approved Mode of Operation*. The value 1 in the file /proc/sys/crypto/fips_enabled confirms operation of the module in the approved mode.

The conditions for using Module cryptographic primitives in the [140] Approved mode of operation are:
1.  The Module is a cryptographic library used by a calling application. The calling application is responsible for the usage of the primitives in the correct sequence, and interpretation of return codes.
2.  With the exception of CSPs managed within the Module boundary (default DRBG state, TLS shared secrets and TLS KDF derived session keys), the keys used by the Module are managed by the calling application, provided on the caller's stack. The calling application is required to provide keys in accordance with FIPS 140-2 requirements. CSPs are zeroized when released by the appropriate API function calls. CSPs managed within the Module boundary are established using only Approved or allowed methods.
3.  The OPENSSL_ENFORCE_MODULUS_BITS must be set to disable generation of RSA and DSA key sizes not listed in Table 4.
4.  The memory occupied by keys is allocated using utility function OPENSSL_alloc. The application is responsible for calling the corresponding OPENSSL_clear_free or OPENSSL_clear_realloc, which overwrites the memory occupied by keys with predefined values before memory free or reallocation. In case of abnormal termination, or swap in/out of a physical memory page of a process, the keys in physical memory are overwritten by the Linux kernel before the physical memory is allocated to another process.
5.  Only the approved and allowed cryptographic functions listed in Table 4 and Table 5 are to be used, along with the guidance detailed in this section. Any use of Table 6 non-approved services transitions the Module to the non-Approved mode of operation.
6.  Use of the ENGINE_register_*, ENGINE_set_default_* function calls, or explicitly setting the module to the non-Approved mode by calling FIPS_mode_set(FALSE) transitions the Module to the non-Approved mode of operation.
7.  Use of Triple-DES is being phased out by NIST, and is treated as deprecated in this security policy, meaning that ciphersuites based on Triple-DES are not cited in the Table 7 list of recommended ciphersuites. [140IG] A.13 *SP 800-67rev1 Transition* requires the calling application to limit encryption with a Triple-DES key used in a recognized IETF protocol to $2^{20}$ 64-bit blocks of data – in this case, the TLS protocols: IETF RFC 2246 (TLS 1.0), RFC 4346 (TLS 1.1) and RFC 5246 (TLS 1.2) define the key derivation method for the corresponding TLS protocol version, including Triple-DES (denoted "3DES" in these documents) keys.
    The calling application must limit encryption with a Triple-DES key used in any other scenario to $2^{16}$ blocks of data.
8.  MD5 is called by code within the Module boundary only for use in the TLS 1.0 / 1.1 KDF (No Security Claimed). Any other use of MD5 must be consistent with [140IG] 1.23 *Definition and Use of a non-Approved Security Function*.
9.  The OpenSSL API call of RAND_cleanup must not be used, as it will clean and free the default DRBG state and replace the default DRBG with the non-FIPS Approved SSLeay Deterministic Random Number Generator when using the RAND_* API calls.
10. The length of a single data unit encrypted with the XTS-AES shall not exceed $2^{20}$ AES blocks (16MB).
11. XTS-AES keys shall only be used to encrypt/decrypt data in storage. The module enforces the [140IG] A.9 requirement to assure k1 ≠ k2 in the key structure initialization.
12. CSPs defined in an Approved mode of operation are not to be accessed or shared while in a non-Approved mode of operation. CSPs shall not be generated while in a non-approved mode.

The conditions for using the Module's *Secure Communication* service in the [140] Approved mode of operation are:

13. Only the NIST P-256, P-384 and P-521 curves shall be used in the approved mode of operation.

14. All certificates used for S*ecure Communications* services must adhere to [131] requirements.

15. In case the Module's power is lost and then restored, the key used for the AES GCM encryption/decryption shall be re-distributed.

    The nonce_explicit part of the IV does not exhaust the maximum number of possible values for a given session key. The design of the TLS protocol in this module implicitly ensures that the nonce_explicit, or counter portion of the IV will not exhaust all of its possible values.

    The module complies with IG A.5, provision 1a.

Additionally, although [52] is not within the scope of [140] validation, this Security Policy is aligned with [52] and FedRAMP controls to assist users in configuration and secure use of systems for compliance with the full suite of applicable standards. Table 7 lists the ciphersuites consistent with a subset of [52] §3.3.1, using only those cryptographic functions available in the approved mode.

Ciphersuites marked with * correspond to the TLS 1.2 ciphersuites used by Nutanix products that incorporate the Module. However, since the mechanism for selecting the set of ciphersuites used by Nutanix products is outside the boundary, the full capability of the Module at the boundary is represented in this Security Policy.

*Table 7: Recommended TLS Cipher Suites*

| ID | IANA Enumeration | OpenSSL Enumeration |
|---|---|---|
| 002F | TLS_RSA_WITH_AES_128_CBC_SHA | AES128-SHA |
| 0035 | TLS_RSA_WITH_AES_256_CBC_SHA | AES256-SHA |
| 0030 | TLS_DH_DSS_WITH_AES_128_CBC_SHA | DH-DSS-AES128-SHA |
| 0031 | TLS_DH_RSA_WITH_AES_128_CBC_SHA | DH-RSA-AES128-SHA |
| 0032 | TLS_DHE_DSS_WITH_AES_128_CBC_SHA | DHE-DSS-AES128-SHA |
| 0033 | TLS_DHE_RSA_WITH_AES_128_CBC_SHA | DHE-RSA-AES128-SHA |
| 0036 | TLS_DH_DSS_WITH_AES_256_CBC_SHA | DH-DSS-AES256-SHA |
| 0037 | TLS_DH_RSA_WITH_AES_256_CBC_SHA | DH-RSA-AES256-SHA |
| 0038 | TLS_DHE_DSS_WITH_AES_256_CBC_SHA | DHE-DSS-AES256-SHA |
| 0039 | TLS_DHE_RSA_WITH_AES_256_CBC_SHA | DHE-RSA-AES256-SHA |
| 003E | TLS_DH_DSS_WITH_AES_128_CBC_SHA256 | DH-DSS-AES128-SHA256 * |
| 003F | TLS_DH_RSA_WITH_AES_128_CBC_SHA256 | DH-RSA-AES128-SHA256 * |
| 0040 | TLS_DHE_DSS_WITH_AES_128_CBC_SHA256 | DHE-DSS-AES128-SHA256 * |
| 0067 | TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 | DHE-RSA-AES128-SHA256 * |
| 0068 | TLS_DH_DSS_WITH_AES_256_CBC_SHA256 | DH-DSS-AES256-SHA256 * |
| 0069 | TLS_DH_RSA_WITH_AES_256_CBC_SHA256 | DH-RSA-AES256-SHA256 * |
| 006A | TLS_DHE_DSS_WITH_AES_256_CBC_SHA256 | DHE-DSS-AES256-SHA256 * |
| 006B | TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 | DHE-RSA-AES256-SHA256 * |
| 009C | TLS_RSA_WITH_AES_128_GCM_SHA256 | AES128-GCM-SHA256 |
| 009D | TLS_RSA_WITH_AES_256_GCM_SHA384 | AES256-GCM-SHA384 |
| 009E | TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 | DHE-RSA-AES128-GCM-SHA256 * |
| 009F | TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 | DHE-RSA-AES256-GCM-SHA384 * |
| 00A0 | TLS_DH_RSA_WITH_AES_128_GCM_SHA256 | DH-RSA-AES128-GCM-SHA256 * |

| ID | IANA Enumeration | OpenSSL Enumeration |
|---|---|---|
| 00A1 | TLS_DH_RSA_WITH_AES_256_GCM_SHA384 | DH-RSA-AES256-GCM-SHA384 * |
| 00A2 | TLS_DHE_DSS_WITH_AES_128_GCM_SHA256 | DHE-DSS-AES128-GCM-SHA256 * |
| 00A3 | TLS_DHE_DSS_WITH_AES_256_GCM_SHA384 | DHE-DSS-AES256-GCM-SHA384 * |
| 00A4 | TLS_DH_DSS_WITH_AES_128_GCM_SHA256 | DH-DSS-AES128-GCM-SHA256 * |
| 00A5 | TLS_DH_DSS_WITH_AES_256_GCM_SHA384 | DH-DSS-AES256-GCM-SHA384 * |
| C004 | TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA | ECDH-ECDSA-AES128-SHA |
| C005 | TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA | ECDH-ECDSA-AES256-SHA |
| C009 | TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA | ECDHE-ECDSA-AES128-SHA |
| C00A | TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA | ECDHE-ECDSA-AES256-SHA |
| C00E | TLS_ECDH_RSA_WITH_AES_128_CBC_SHA | ECDH-RSA-AES128-SHA |
| C00F | TLS_ECDH_RSA_WITH_AES_256_CBC_SHA | ECDH-RSA-AES256-SHA |
| C013 | TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA | ECDHE-RSA-AES128-SHA |
| C014 | TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA | ECDHE-RSA-AES256-SHA |
| C023 | TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 | ECDHE-ECDSA-AES128-SHA256 * |
| C024 | TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 | ECDHE-ECDSA-AES256-SHA384 * |
| C025 | TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256 | ECDH-ECDSA-AES128-SHA256 * |
| C026 | TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384 | ECDH-ECDSA-AES256-SHA384 * |
| C027 | TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 | ECDHE-RSA-AES128-SHA256 * |
| C028 | TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 | ECDHE-RSA-AES256-SHA384 * |
| C029 | TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256 | ECDH-RSA-AES128-SHA256 * |
| C02A | TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384 | ECDH-RSA-AES256-SHA384 * |
| C02B | TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 | ECDHE-ECDSA-AES128-GCM-SHA256 * |
| C02C | TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 | ECDHE-ECDSA-AES256-GCM-SHA384 * |
| C02D | TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256 | ECDH-ECDSA-AES128-GCM-SHA256 * |
| C02E | TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384 | ECDH-ECDSA-AES256-GCM-SHA384 * |
| C02F | TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 | ECDHE-RSA-AES128-GCM-SHA256 * |
| C030 | TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 | ECDHE-RSA-AES256-GCM-SHA384 * |
| C031 | TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256 | ECDH-RSA-AES128-GCM-SHA256 * |
| C032 | TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384 | ECDH-RSA-AES256-GCM-SHA384 * |
| C09E | TLS_DHE_RSA_WITH_AES_128_CCM | DHE-RSA-AES128-CCM |
| C09F | TLS_DHE_RSA_WITH_AES_256_CCM | DHE-RSA-AES256-CCM |
| C0A2 | TLS_DHE_RSA_WITH_AES_128_CCM_8 | DHE-RSA-AES128-CCM8 |
| C0A3 | TLS_DHE_RSA_WITH_AES_256_CCM_8 | DHE-RSA-AES256-CCM8 |
| C0AC | TLS_ECDHE_ECDSA_WITH_AES_128_CCM | ECDHE-ECDSA-AES128-CCM |
| C0AD | TLS_ECDHE_ECDSA_WITH_AES_256_CCM | ECDHE-ECDSA-AES256-CCM |
| C0AE | TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 | ECDHE-ECDSA-AES128-CCM8 |
| C0AF | TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8 | ECDHE-ECDSA-AES256-CCM8 |

## 4    Critical Security Parameters and Public Keys

All CSPs and public keys used by the Module are described in this section. Note that the term SSP refers collectively to critical security parameters (CSPs) and public security parameters (e.g., public keys). The list of SSPs is arranged for consistency with Table 11, which in turn is organized for ease of review.

Prefixes:

DRBG = Deterministic random bit generation (random number generation service);
DS = Digital signature service; GKP = Generate key pair service; KAS = Key agreement service;
KH = Keyed hash service; KTS = Key transport service;
SED = symmetric encrypt/decrypt service; TLS = TLS secure communications service.

*Table 8: SSPs Used for Cryptographic Primitives*

| SSP | Description/Usage |
|---|---|
| DRBG_Seed | DRBG seed, inclusive of entropy input from the registered callback function, sized as (8 * security_strength) + 64 bits; 128 bit nonce and optional personalization string (as defined by caller) for instantiation; optional additional input for reseed (as defined by caller). |
| DRBG_State | Default DRBG use for key generation:<br>AES-256 CTR_DRBG state: 256-bit K, 128-bit V.<br>CTR_DRBG: V (128 bits) and Key (128, 192 or 256 bits)<br>Hash_DRBG: V (440 to 880 bits) and C (440 to 880 bits)<br>HMAC_DRBG:  V (440 to 880 bits) and Key (160 to 512 bits)<br>The libcrypto.so.1.0.2k API provides DRBG functions to calling applications, with the DRBG structure memory allocated, stored and managed by the calling application. |
| DS_Private | Private component of an ECC[8], DSA[9] or RSA[10] key pair used by the Digital Signature service. |
| DS_Public | Public component of an ECC[11], DSA[12] or RSA[13] key pair used by the Digital Signature service. |
| GKP_Private | Private component of an ECC[11], DSA/FFC[12] or RSA[13] key pair generated by the Generate Key Pair service, or managed by the CM service. |
| GKP_Public | Public component of an ECC[11], DSA/FFC[12] or RSA[13] key pair generated by the Generate Key Pair service, or managed by the CM service. |
| KAS_SS | The Elliptic Curve Diffie-Hellman ([56A] Section 5.7.1.2 ECC CDH) or Diffie-Hellman ([56A] Section 5.7.1.1 FFC DH) shared secret. |
| KAS_Private | Private component of an ECC[3] or FFC[4] key pair used by the Key Agreement service. |
| KAS_Public | Public component of an ECC[3] or FFC[4] key pair used by the Key Agreement service. |
| KH_Key | CMAC or GMAC: (AES-128, AES-192, AES-256, 3-Key Triple-DES), or HMAC (160-bit, 256-bit or 512-bit) key used for generating or verifying keyed hashes. |
| KTS_SS | The RSA key transport shared secret (112-bit security strength). |
| KTS_Private | Private component of an RSA key pair (k = 2048) used for RSA key transport. |
| KTS_Public | Public component of an RSA key pair (k = 2048) used for RSA key transport. |
| SED_EDK | AES (128-bit, 192-bit, 256-bit) or Triple-DES (168-bit 3-Key, 112-bit equivalent strength) encrypt/decrypt key. |

---

[8] The approved mode of the Module permits the curves as shown in Table 4 for ECC key generation, ECDSA signature generation and verification and EC Diffie-Hellman key agreement primitives.

[9] The approved mode of the Module permits the parameter as shown in Table 4 for DSA/FFC key generation, DSA signature generation and verification and Diffie-Hellman key agreement.

[10] The approved mode of the Module permits the parameters as shown in Table 4 for RSA key generation, RSA signature generation and verification and RSADP.

*Table 9: SSPs Used for TLS Secure Communications*

| SSP / Public | Description/Usage |
|---|---|
| TLS-DH-Priv | FFC n=2048 or ECDSA P-256/P-384/P-521 Private Key used for TLS Diffie-Hellman key agreement |
| TLS-DH-Pub | FFC n=2048 or ECDSA P-256/P-384/P-521 Public Key used for TLS Diffie-Hellman key agreement (provided to peer in handshake) |
| TLS-DH-Peer | FFC n=2048 or ECDSA P-256/P-384/P-521 Public Key used for TLS Diffie-Hellman key agreement (provided by peer in handshake) |
| TLS-Host-Priv | RSA n=2048 and ECDSA P-256/P-384/P-521 Private Key |
| TLS-Host-Pub | RSA n=2048 and ECDSA P-256/P-384/P-521 Public Key |
| TLS-MS | TLS Master Secret: 384-bit secret key material |
| TLS-PMS | TLS Pre-Master Secret: 2048/384-bit secret key material |
| TLS-SENC | AES CBC, CCM, GCM 128-bit, 256-bit key or 3-key Triple-DES used for TLS secure communications session encryption |
| TLS-SMAC | HMAC-SHA-1 (160 bit) or HMAC-SHA-256 (256 bit). Note that although CCM and GCM accomplish the same message integrity purpose, those algorithms do not require separate keys for cipher and integrity. |

## 5    Roles and Services

The Module supports two distinct operator roles, User and Cryptographic Officer (CO), and does not support multiple concurrent operators, a maintenance role or bypass capability. The Module does not provide an authentication or identification method of its own.  The CO and the User roles are implicitly identified by the service requested.

All services implemented by the Module are listed in Table 10. Keys are provided to the Module by the calling application; manual key entry is not supported. Data output is inhibited during self-tests, zeroization, and error states. The module runs in a single-threaded process that inhibits data output when in the self-test state by preventing invocation of any other cryptographic functions via the API. The module responds to specific calls through the API which are deliberate in their function; and thus, the module does not indiscriminately output data. Status information does not contain CSPs or sensitive data that if misused could lead to Module compromise.

*Table 10: Authorized Services Available in FIPS Mode*

| Service | Description | Role |
|---|---|---|
| Certificate Management | Parse and format certificates. | User |
| Digital signature | Generate or verify DSA, ECDSA or RSA digital signatures. | User |
| Generate key pair | Generate DSA (FFC), ECDSA, or RSA key pairs. | User |
| Initialize | Initialize the Module, inclusive of Self-test. | User |
| Install | Install and configure the Module. | CO |
| Key agreement | DH (FFC), ECDH key agreement primitives. | User |
| Key transport | RSA key transport primitives. | User |
| Keyed hash | Generate or verify data integrity (CMAC, GMAC or HMAC). | User |
| Message digest | Generate a SHA-1 or SHA-2 message digest. | User |
| Random number generation | [90A] DRBG for random number generation. | User |
| Secure communications | Establish and use TLS secure communications. | User |
| Show status | Functions that give Module status feedback. | User |
| Symmetric encrypt/decrypt | Symmetric data encrypt and decrypt. | User |
| Zeroize | Functions that destroy CSPs. | User |
| Utility | Support functions, e.g. number conversion, compression. | User |

Table 11 describes Module service access to CSPs and public keys. In each cell below, the following annotations indicate the type of access by the Module service:

- E (Execute): The service uses the CSP or public key for service execution. All CSPs are provided by the calling application as positional parameters on the stack; the calling application owns the stack; the Module zeroizes all local copies of a CSP before returning.
- G (Generate): The Module generates or derives the cryptographic keys/CSPs internally. The Module does not retain copies of the key after call completion. In the case of key transport decapsulation, the KTS_SS CSP is produced through decryption of the wrapped key. DSA and ECDSA signature services also require a random (the DRBG Generate primitive corresponds to DRBG_State "EG"). The Initialize service initializes and instantiates the default DRBG, and DRBG_REI may also be used for default DRBG reseed or instantiation or reseed of a user requested DRBG instance.
- I (Input): The Module receives the CSP on the stack, or obtains data from the entropy callback function.
- O (Output): The Module outputs a CSP/cryptographic key to the calling application through the logical interface. The Module does not output CSPs through a physical port.
- Z (Zeroize): The Module zeroizes the CSP.

*Table 11: CSP Access Rights within Services*

| Service \ Key | DRBG_Seed | DRBG_State | DS_Private | DS_Public | GKP_Private | GKP_Public | KAS_SS | KAS_Private | KAS_Public | KH_Key | KTS_SS | KTS_Private | KTS_Public | SED_EDK | TLS-DH-Priv | TLS-DH-Pub | TLS-DH-Peer | TLS-Host-Priv | TLS-Host-Pub | TLS-MS | TLS-PMS | TLS-SENC | TLS-SMAC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Certificate Management | -- | -- | -- | -- | EIO | EIO | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Deterministic random number generation | EIZ | EG | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Digital signature | -- | EG | EI | EI | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Generate key pair | -- | EG | -- | -- | GO | GO | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Initialize | EGZ | EG | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Install | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Key agreement | -- | -- | -- | -- | -- | -- | GO | EI | EI | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Key transport | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | GO | EI | EI | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Keyed hash | -- | -- | -- | -- | -- | -- | -- | -- | -- | EI | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Message digest | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Secure communications | -- | EG | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | EI | O | EI | EI | EI | GEZ | GEZ | GEZ | GEZ |
| Show status | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Symmetric encrypt/decrypt | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | EI | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Utility | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Zeroize | -- | Z | Z | -- | Z | -- | Z | Z | -- | Z | Z | Z | -- | Z | Z | -- | -- | Z | -- | -- | -- | -- | -- |

## 6   Self-Tests

Each time the Module is powered up, it tests that the cryptographic algorithms still operate correctly and that sensitive data have not been damaged. The Module provides a default entry point to automatically run the power on self-tests without operator intervention compliant with [140IG] 9.10 *Power-Up Tests for Software Module Libraries*. Power on self–tests are available on demand by reloading the Module.

On power-on or reset, the Module performs the self-tests described in Table 12. All KATs must complete successfully prior to any other use of cryptography by the Module. Algorithms that rely on random values (DSA, ECDSA, DRBG) are tested using a fixed value.

*Table 12: Power-on Self-Tests*

| Test Target | Description |
|---|---|
| Software Integrity | HMAC-SHA-256 with a 256-bit key. |
| AES | Separate encryption and decryption KATs; ECB 128-bit key. |
| AES CCM | Separate encryption and decryption KATs; 192-bit key. |
| AES GCM | Separate encryption and decryption KATs; 256-bit key. |
| AES XTS | Separate encryption and decryption KATs; 128-bit key and 256-bit key. |
| CMAC | KATs using AES-128, AES-192, AES-256 and 3-Key Triple-DES. |
| DSA | Pairwise consistency test (PCT), signature generation and verification, L = 2048 N = 256 and SHA-256. |
| DRBG | Separate KATs covering [90A] Section 11.3 health test (Instantiate, Generate Reseed) for:<br>- CTR_DRBG (AES-128, AES-192, AES-256) all with and without prediction resistance, and all with and without derivation function.<br>- HASH_DRBG (SHA-1, SHA-256, SHA-384, SHA-512) all with and without prediction resistance.<br>- HMAC_DRBG (SHA-1, SHA-256, SHA-384, SHA-512) all with and without prediction resistance, using a 160-bit HMAC key. |
| ECDSA | Pairwise consistency test (PCT), signature generation and verification; P-256 curve, SHA-256. |
| HMAC | HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 KATs. |
| KAS-ECC-SSC | [56A] Section 5.7.1.2 primitive "Z" computation KAT per [140IG] 9.6; P-256 curve. |
| KAS-FFC-SSC | [56A] Section 5.7.1.1 primitive "Z" computation KAT per [140IG] 9.6; L = 2048 N = 256. |
| RSA | Separate signature generation and signature verification KATs. Tested with k = 2048, SHA-256. |
| SHS | SHA-1, SHA-256 (inclusive of SHA-224), SHA-512 KATs (inclusive of SHA-384) |
| Triple-DES | Separate encryption and decryption KATs; 3-Key. |

*Table 13: Conditional Self-Tests*

| Test Target | Description |
|---|---|
| CRNGT | AS09.42 continuous RNG test performed on entropy source (callback function output). |
| DSA | PCT, signature generation and verification |
| ECDSA | PCT, signature generation and verification |
| RSA | PCT, signature generation and verification, encryption and decryption |

PCTs are performed in accordance with [140IG] 9.9 *Pair-Wise Consistency Self-Test When Generating a Key Pair*.