# Cradlepoint, Inc.

## Cradlepoint Cryptographic Module

Software Version: 1.0

# FIPS 140-3 Non-Proprietary Security Policy

**FIPS Security Level: 1**
**Document Version: 0.4**

| **Prepared for:** | **Prepared by:** |
|---|---|
| **Cradlepoint, Inc.**<br>1100 W. Idaho Street, Floor 8<br>Boise, ID 83702-5389<br>United States<br><br>Phone: +1 888-331-2968<br>www.cradlepoint.com | **Corsec Security, Inc.**<br>12600 Fair Lakes Circle, Suite 210<br>Fairfax, VA 22033<br>United States of America<br><br>Phone: +1 703 267 6050<br>www.corsec.com |

# Abstract

This is a non-proprietary Cryptographic Module Security Policy for the Cradlepoint Cryptographic Module (version: 1.0) from Cradlepoint, Inc. (Cradlepoint). This Security Policy describes how the Cradlepoint Cryptographic Module meets the security requirements of Federal Information Processing Standards (FIPS) Publication 140-3, which details the U.S. and Canadian government requirements for cryptographic modules. More information about the FIPS 140-3 standard and validation program is available on the National Institute of Standards and Technology (NIST) and the Canadian Centre for Cyber Security (CCCS) Cryptographic Module Validation Program (CMVP) website at http://csrc.nist.gov/groups/STM/cmvp.

This document also describes how to run the module in its Approved mode of operation. This policy was prepared as part of the Level 1 FIPS 140-3 validation of the module. The Cradlepoint Cryptographic Module is referred to in this document as Cradlepoint Cryptographic Module or the module.

# References

This document deals only with operations and capabilities of the module in the technical terms of a FIPS 140-3 cryptographic module security policy. More information is available on the module from the following sources:

- The Cradlepoint website www.cradlepoint.com contains information on the full line of services and solutions from Cradlepoint.

- The search page on the CMVP website (https://csrc.nist.gov/Projects/cryptographic-module-validation-program/Validated-Modules/Search) can be used to locate and obtain vendor contact information for technical or sales-related questions about the module.

# Document Organization

*ISO/IEC 19790* Annex B uses the same section naming convention as *ISO/IEC 19790* section 7 - Security requirements. For example, Annex B section B.2.1 is named “General” and B.2.2 is named “Cryptographic module specification,” which is the same as *ISO/IEC 19790* section 7.1 and section 7.2, respectively. Therefore, the format of this Security Policy is presented in the same order as indicated in Annex B, starting with “General” and ending with “Mitigation of other attacks.” If sections are not applicable, they have been marked as such in this document.

# Table of Contents

# List of Tables

# List of Figures

# 1. General

Cradlepoint is a global leader in cloud-delivered 4G and 5G wireless network edge solutions. Cradlepoint's NetCloud™ platform and cellular routers deliver a pervasive, secure, and software-defined Wireless WAN[1] edge to connect people, places, and things – anywhere. More than 28,500 businesses and government agencies worldwide, including many Global 2000 organizations and top public sector agencies, rely on Cradlepoint to keep mission-critical sites, points of commerce, field forces, vehicles, and IoT[2] devices always connected.

Cradlepoint NetCloud for Branch makes it easy to accelerate connecting to the Internet and critical applications from anywhere. Designed for traditional medium branches or locations requiring flexible connectivity, reliable performance, and simplified management, this all-in-one, compact endpoint includes full-featured routing, security, and Wi-Fi without needing extra hardware or complicated configurations.

The Cradlepoint Cryptographic Module version 1.0 is a cryptographic library as part of the NetCloud operating system (OS) kernel that provides cryptographic services for Cradlepoint endpoints. The module offers symmetric encryption/decryption, digital signature generation/verification, hashing, cryptographic key generation, random number generation, message authentication, and SSP establishment functions to secure data-at-rest/data-in-flight and to support secure communications protocols (including TLS[3] 1.2).

Cradlepoint Cryptographic Module is validated at the FIPS 140-3 section levels shown in Table 1.

**Table 1 – Security Level per FIPS 140-3 Section**

| ISO/IEC 24759 Section 6. [Number Below] | 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 | N/A |
| 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 |

The module has an overall security level of 1.

[1] WAN – Wide Area Network
[2] IoT – Internet of Things
[3] TLS – Transport Layer Security

# 2. Cryptographic Module Specification

The Cradlepoint Cryptographic Module (version 1.0) is a software module with a multi-chip standalone embodiment. The module is designed to operate within a modifiable operational environment.

## 2.1 Operational Environments

The module was tested and found to be compliant with FIPS 140-3 requirements on the environment listed in Table 2.

**Table 2 – Tested Operational Environments**

| # | Operating System | Hardware Platform | Processor | PAA/Acceleration |
|---|---|---|---|---|
| 1 | NetCloud OS 7 | Cradlepoint E3000 | ARM Cortex-A (ARMv8-A) | With |
| 2 | NetCloud OS 7 | Cradlepoint E3000 | ARM Cortex-A (ARMv8-A) | Without |

Each test environment includes a Qualcomm 802.11ax SoC[4] with four ARM Cortex-A processing cores. These processing cores support the ARMv8 Cryptography Extensions. The Cryptography Extensions include A64, A32, and T32 instructions for accelerating AES and SHA implementations. The module is designed to utilize the extended instruction sets when available.

The vendor affirms the module's continued validation compliance when operating on the environments listed in Table 3.

**Table 3 – Vendor-Affirmed Operational Environments**

| # | Operating System | Hardware Platform |
|---|---|---|
| 1 | NetCloud OS 7 | Cradlepoint R920 |
| 2 | NetCloud OS 7 | Cradlepoint R2105/R2155 |
| 3 | NetCloud OS 7 | Cradlepoint R1900 |
| 4 | NetCloud OS 7 | Cradlepoint E300 |
| 5 | NetCloud OS 7 | Cradlepoint S700/S750 |
| 6 | NetCloud OS 7 | Cradlepoint R980 |

The cryptographic module maintains validation compliance when operating on any general-purpose computer (GPC) provided that the GPC uses any single-user operating system/mode specified on the validation certificate, or another compatible single-user operating system. 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 not listed on the validation certificate.

[4] SoC – System on a Chip

The sections below describe the module boundary, modes of operation, and algorithm implementations.

## 2.2 Algorithm Implementations

Validation certificates for each Approved security function are listed in Table 4 below.

**Table 4 – Approved Algorithm Validation Certificates**

| CAVP Certificate[5] | Algorithm and Standard | Mode / Method | Description / Key Size(s) / Key Strengths | Use / Function |
|---|---|---|---|---|
| A2584 | **AES**<br>*FIPS PUB[6] 197*<br>*NIST SP 800-38A* | CBC[7], CTR[8], ECB[9] | 128, 192, 256 | Encryption/decryption |
| Vendor Affirmed | **CKG**[10]<br>*NIST SP 800-133rev2* | - | - | Cryptographic key generation |
| A2584 | **CVL**[11]<br>*NIST SP 800-135rev1* | TLS[12] 1.2 KDF | - | Key derivation[13] |
| A2584 | **CVL**<br>*RFC 7627* | TLS 1.2 KDF RFC7627 | - | Key derivation[14] |
| A2584 | **DRBG**[15]<br>*NIST SP 800-90Arev1* | Counter-based | AES-128, AES-192, AES-256 | Deterministic random bit generation |
| A2584 | **DSA**[16]<br>*FIPS PUB 186-4* | - | 2048/224, 2048/256 | Key pair generation |
| | | SHA2-256 | 2048/224, 2048/256 | Domain parameter generation |
| | | SHA2-256 | 2048/224, 2048/256 | Domain parameter verification |
| | | SHA2-256 | 2048/224, 2048/256 | Digital signature generation |
| | | SHA-1, SHA2-256 | 1024/160, 2048/224, 2048/256 | Digital signature verification |
| A2584 | **ECDSA**[17]<br>*FIPS PUB 186-4* | Secret generation method: Testing candidates | P-224, P-256, P-384 | Key pair generation |
| | | - | P-224, P-256, P-384 (SHA2-256, SHA2-384, SHA2-512) | Public key verification |
| | | - | P-224, P-256, P-384 (SHA2-256, SHA2-384, SHA2-512) | Digital signature generation |

[5] This table includes vendor-affirmed algorithms that are approved but CAVP testing is not yet available.
[6] PUB – Publication
[7] CBC – Cipher Block Chaining
[8] CTR – Counter
[9] ECB – Electronic Code Book
[10] CKG – Cryptographic Key Generation
[11] CVL – Component Validation List
[12] TLS – Transport Layer Security
[13] No part of the TLS protocol, other than the KDF, has been tested by the CAVP and CMVP.
[14] No part of the TLS protocol, other than the KDF, has been tested by the CAVP and CMVP.
[15] DRBG – Deterministic Random Bit Generator
[16] DSA – Digital Signature Algorithm
[17] ECDSA – Elliptic Curve Digital Signature Algorithm

| CAVP Certificate[5] | Algorithm and Standard | Mode / Method | Description / Key Size(s) / Key Strengths | Use / Function |
|---|---|---|---|---|
| | | - | P-224, P-256, P-384 (SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512) | Digital signature verification |
| A2584 | **HMAC**<br>*FIPS PUB 198-1* | SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512 | 112 (minimum) | Message authentication |
| A2584 | **KAS**[18]<br>*NIST SP 800-56Arev3*<br>*NIST SP 800-135rev1*<br>*RFC 7627* | KAS-ECC-SSC with TLS 1.2 KDF RFC7627 | P-224, P-256, P-384 | Key agreement[19]<br>*SSP establishment methodology provides between 112 and 256 bits of encryption strength.* |
| | | KAS-FFC-SSC with TLS 1.2 KDF RFC7627 | FB, FC, MODP-2048, MODP-3072, MODP-4096 | Key agreement[20]<br>*SSP establishment methodology provides 112 bits of encryption strength.* |
| A2584 | **KAS-ECC-SSC**[21]<br>*NIST SP 800-56Arev3* | EphemeralUnified | P-224, P-256, P-384 | Shared secret computation |
| A2584 | **KAS-FFC-SSC**[22]<br>*NIST SP 800-56Arev3* | dhEphem | FB, FC, MODP-2048, MODP-3072, MODP-4096 | Shared secret computation |
| A2584 | **KTS**[23]<br>*FIPS PUB 197*<br>*FIPS PUB 198-1* | AES with HMAC | 128, 192, 256 | Key wrap/unwrap (encryption//decryption with message authentication)[24]<br>*SSP establishment methodology provides 112 bits of encryption strength.* |
| A2584 | **KTS**<br>*NIST SP 800-67rev2*<br>*FIPS PUB 198-1* | Triple-DES[25] with HMAC | 168 (KO1) | Key unwrap (decryption with message authentication)[26]<br>*SSP establishment methodology provides 168 bits of encryption strength.* |
| A2584 | **PBKDF2**[27]<br>*NIST SP 800-132* | Section 5.4, option 1a | SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512 | Password-based key derivation |
| A2584 | **RSA**[28]<br>*FIPS PUB 186-4* | Key generation mode: B.3.3 | 2048, 3072, 4096 | Key pair generation |
| | | ANSI X9.31 | 2048, 3072, 4096 (SHA2-256, SHA2-384, SHA2-512) | Digital signature generation |

[18] KAS – Key Agreement Scheme
[19] Key agreement method complies with *FIPS 140-3 Implementation Guidance* D.F, scenario 2(1).
[20] Key agreement method complies with *FIPS 140-3 Implementation Guidance* D.F, scenario 2(1).
[21] KAS-ECC-SSC – Key Agreement Scheme - Elliptic Curve Cryptography - Shared Secret Computation
[22] KAS-FFC-SSC – Key Agreement Scheme - Finite Field Cryptography - Shared Secret Computation
[23] KTS – Key Transport Scheme
[24] Per *FIPS 140-3 Implementation Guidance* D.G, AES in any Approved mode with HMAC is an Approved key transport technique.
[25] DES – Data Encryption Standard
[26] Per *FIPS 140-3 Implementation Guidance* D.G, Triple DES in any Approved mode with HMAC is an Approved key transport technique.
[27] PBKDF2 – Password-based Key Derivation Function 2
[28] RSA – Rivest Shamir Adleman

| CAVP Certificate[5] | Algorithm and Standard | Mode / Method | Description / Key Size(s) / Key Strengths | Use / Function |
|---|---|---|---|---|
| | | | 1024, 2048, 3072, 4096 (SHA-1, SHA2-256, SHA2-384, SHA2-512) | Digital signature verification |
| | | PKCS#1 v1.5 | 2048, 3072, 4096 (SHA2-224, SHA2-256, SHA2-384, SHA2-512) | Digital signature generation |
| | | | 1024, 2048, 3072, 4096 (SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512) | Digital signature verification |
| | | PSS[29] | 2048, 3072, 4096 (SHA2-224, SHA2-256, SHA2-384, SHA2-512) | Digital signature generation |
| | | | 1024, 2048, 3072, 4096 (SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512) | Digital signature verification |
| A2584 | **Safe Primes**<br>*NIST SP 800-56Arev3, Appendix D* | - | MODP-2048, MODP-3072, MODP-4096 | Key generation |
| A2584 | **SHS[30]**<br>*FIPS PUB 180-4* | SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512 | - | Message digest |
| A2584 | **Triple-DES**<br>*NIST SP 800-67rev2*<br>*NIST SP 800-38A* | CBC, ECB | 168 (KO1) | Encryption/decryption<br>*The encrypt function is used only to support self-testing. During operation, it is not available in the Approved mode.* |

The vendor affirms the following cryptographic security methods:

- Cryptographic key generation – In compliance with section 4 of *NIST SP 800-133rev2*, the module uses its Approved DRBG to generate random values and seeds used for asymmetric key generation. The generated seed is an unmodified output from the DRBG.

The module implements the non-Approved but allowed algorithms shown in Table 5 below.

**Table 5 – Non-Approved Algorithms Allowed in the Approved Mode of Operation**

| Algorithm | Caveat | Use / Function |
|---|---|---|
| AES (Cert. A2584) | SSP establishment methodology provides between 128 and 256 bits of encryption strength. | Key unwrapping (using any approved mode) |
| Triple-DES (Cert. A2584) | SSP establishment methodology provides between 112 and 168 bits of encryption strength. | Key unwrapping (using any approved mode with two-key or three-key) |

[29] PSS – Probabilistic Signature Scheme
[30] SHS – Secure Hash Standard

The module does not implement any non-Approved algorithms allowed in the Approved mode of operation for which no security is claimed.

The module does not implement any non-Approved algorithms not allowed in the Approved mode of operation.

## 2.3 Cryptographic Boundary

As a software cryptographic module, the module has no physical components. Therefore, the physical perimeter of the cryptographic module is defined by each host device on which the module is installed. Figure 1 below illustrates a block diagram of a typical general-purpose computer (GPC) and the module's physical perimeter.

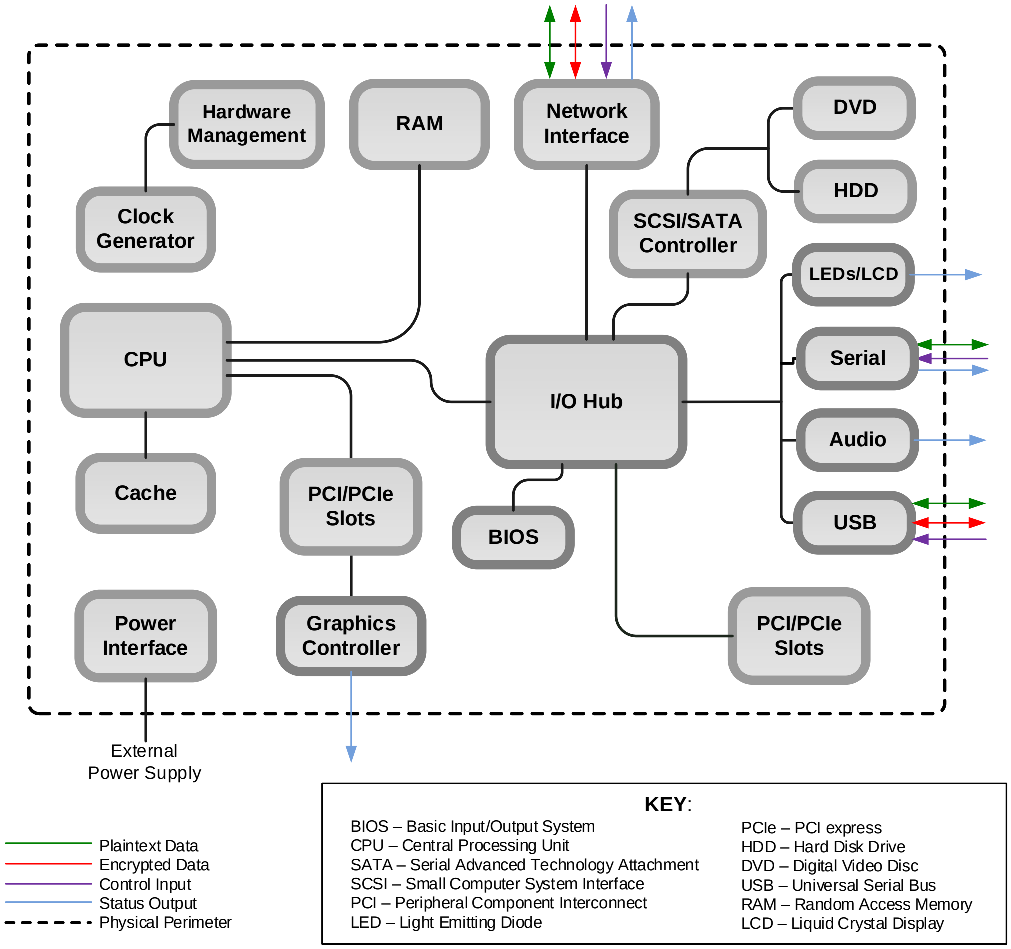

**Figure 1 – GPC Block Diagram**

The module's cryptographic boundary consists of all functionalities contained within the module's compiled source code. This comprises:

- libcrypto.so (cryptographic primitives library file)
- libssl.so (TLS protocol library file)

- libcrypto.hmac (an HMAC[41] digest file for libcrypto integrity checking)
- libssl.hmac (an HMAC digest file for libssl integrity checking)

The cryptographic boundary is the contiguous perimeter that surrounds all memory-mapped functionality provided by the module when loaded and stored in the host device's memory. The module is entirely contained within the physical perimeter.

Figure 2 shows the logical block diagram of the module executing in memory, its location with respect to the operating system and other supporting applications, and its interactions with surrounding software components, as well as the host platform's physical perimeter and module's cryptographic boundary .

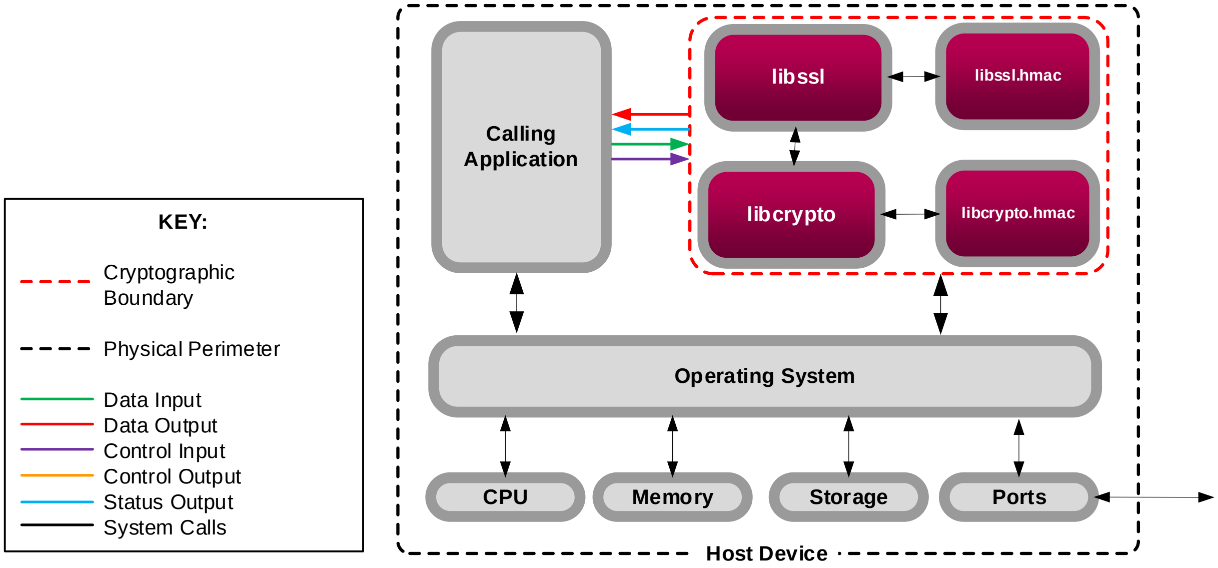

**Figure 2 – Module Block Diagram (with Cryptographic Boundary)**

## 2.4 Modes of Operation

module only implements one mode of operation, the Approved mode, in which the Approved and allowed cryptographic functions are available. The module transitions to the Approved mode of operation automatically after the module completes its pre-operational self-tests. No configuration is necessary for the module to operate and remain in the Approved mode.

[41] HMAC – Keyed-Hash Message Authentication Code

# 3. Cryptographic Module Interfaces

FIPS 140-3 defines the following logical interfaces for cryptographic modules:

- Data Input
- Data Output
- Control Input
- Control Output
- Status Output

As a software library, the cryptographic module has no direct access to any of the host platform's physical ports, as it communicates only to the calling application via its well-defined API. A mapping of the FIPS-defined interfaces and the module's interfaces can be found in Table 6. Note that the module does not output control information, and thus has no specified control output interface.

**Table 6 – Ports and Interfaces**

| Physical Port | Logical Interface | Data That Passes Over Port/Interface |
|---|---|---|
| Physical data input port(s) of the tested platforms | Data Input<br>• API input arguments that provide input data for processing | • Data to be encrypted, decrypted, signed, verified, or hashed<br>• Keys to be used in cryptographic services<br>• Random seed material for the module's DRBG<br>• Keying material to be used as input to key establishment services |
| Physical data output port(s) of the tested platforms | Data Output<br>• API output arguments that return generated or processed data back to the caller | • Data that has been encrypted, decrypted, or verified<br>• Digital signatures<br>• Hashes<br>• Random values generated by the module's DRBG<br>• Keys established using module's key establishment methods |
| Physical control input port(s) of the tested platforms | Control Input<br>• API input arguments that are used to initialize and control the operation of the module | • API commands invoking cryptographic services<br>• Modes, key sizes, etc. used with cryptographic services |
| Physical status output port(s) of the tested platforms | Status Output<br>• API call return values | • Status information regarding the module<br>• Status information regarding the invoked service/operation |

# 4. Roles, Services, and Authentication

The sections below describe the module's authorized roles, services, and operator authentication methods.

## 4.1 Authorized Roles

The module supports a Crypto Officer that authorized operators can assume. The CO role performs cryptographic initialization or management functions and general security services.

The module also supports the following role(s):

- User – The User role performs general security services, including cryptographic operations and other approved security functions.

The module does not support multiple concurrent operators. The calling application that loaded the module is its only operator.

Table 7 below lists the supported roles, along with the services (including input and output) available to each role.

**Table 7 – Roles, Service Commands, Input and Output**

| Role | Service | Input | Output |
|---|---|---|---|
| CO | Show Status | API call | Current operational status |
| CO | Perform self-tests on-demand | Re-instantiate module; API call parameters | Status |
| CO | Zeroize | Restart calling application; reboot or power-cycle host platform | None |
| CO | Show versioning information | API call parameters | Module name, version |
| User | Perform symmetric encryption | API call parameters, key, plaintext | Status, ciphertext |
| User | Perform symmetric decryption | API call parameters, key, ciphertext | Status, plaintext |
| User | Generate random number | API call parameters, entropy input | Status, random bits |
| User | Perform keyed hash operations | API call parameters, key, message | Status, MAC[42] |
| User | Perform hash operation | API call parameters, message | Status, hash |
| User | Generate DSA domain parameters | API call parameters | Status, domain parameters |
| User | Verify DSA domain parameters | API call parameters | Status, domain parameters |
| User | Generate asymmetric key pair | API call parameters | Status, key pair |
| User | Verify ECDSA public key | API call parameters, key | Status |
| User | Generate digital signature | API call parameters, key, message | Status, signature |
| User | Verify digital signature | API call parameters, key, signature, message | Status |

[42] MAC – Message Authentication Code

| Role | Service | Input | Output |
|---|---|---|---|
| User | Perform key wrap | API call parameters, wrapping key, plaintext key | Status, encrypted key |
| User | Perform key unwrap | API call parameters, wrapping key, wrapped key | Status, decrypted key |
| User | Compute shared secret | API call parameters | Status, shared secret |
| User | Derive TLS keys | API call parameters, TLS pre-master secret | Status, TLS keys |
| User | Derive key via PBKDF2 | API call parameters, password | Status, key |

## 4.2 Authentication Methods

The module does not support authentication methods; operators implicitly assume an authorized role based on the service selected.

## 4.3 Services

Descriptions of the Approved services available to the authorized roles are provided in Table 8 below.

The module is an integrated component of Cradlepoint's NetCloud OS and offers crypto functions to applications installed on Cradlepoint devices. While the module includes implementations of non-Approved security functions that can be called by other Cradlepoint modules, all such invocations will return failure codes to the caller. This effectively limits the service offerings to Approved services only.

As allowed for this scenario per section C.H of *FIPS 140-3 Implementation Guidance*, the module provides indicators for the use of Approved services through a combination of an explicit indication (via a global FIPS mode indicator) and an implicit indication (via the API return value of the service).

The keys and Sensitive Security Parameters (SSPs) listed in the table indicate the type of access required using the following notation:

- 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 = Zeroize: The module zeroizes the SSP.

**Table 8 – Approved Services**

| Service | Description | Approved Security Function(s) | Keys and/or SSPs | Roles | Access Rights to Keys and/or SSPs | Indicator |
|---|---|---|---|---|---|---|
| Show status | Return mode status | None | None | CO | N/A | N/A |
| Perform self-tests on-demand | Perform pre-operational self-tests | HMAC (Cert. A2584)<br>SHA2-256 (Cert. A2584) | None | CO | N/A | API return value |
| Zeroize | Zeroize and de-allocate memory containing sensitive data | None | All SSPs | CO | All SSPs – Z | N/A |

| Service | Description | Approved Security Function(s) | Keys and/or SSPs | Roles | Access Rights to Keys and/or SSPs | Indicator |
|---|---|---|---|---|---|---|
| Show versioning information | Return module versioning information | None | None | CO | N/A | N/A |
| Perform symmetric encryption | Encrypt plaintext data | AES (Cert. A2584) | AES key | User | AES key – WE | API return value |
| Perform symmetric decryption | Decrypt ciphertext data | AES (Cert. A2584)<br>Triple-DES (Cert. A2584) | AES key<br>Triple-DES key | User | AES key – WE<br>Triple-DES key – WE | API return value |
| Generate random number | Generate random bits using DRBG | DRBG (Cert. A2584) | DRBG entropy input<br>DRBG seed<br>DRBG 'V' value<br>DRBG 'Key' value | User | DRBG entropy input – WE<br>DRBG seed – GE<br>DRBG 'V' value – GE<br>DRBG 'Key' value – GE | API return value |
| Perform keyed hash operations | Compute a message authentication code | HMAC (Cert. A2584)<br>SHS (Cert. A2584) | HMAC key | User | HMAC key – WE | API return value |
| Perform hash operation | Compute a message digest | SHS (Cert. A2584) | None | User | N/A | API return value |
| Generate DSA domain parameters | Generate DSA domain parameters | DSA (Cert. A2584) | None | User | N/A | API return value |
| Verify DSA domain parameters | Verify DSA domain parameters | DSA (Cert. A2584) | None | User | N/A | API return value |
| Generate asymmetric key pair | Generate a public/private key pair | CKG<br>DSA (Cert. A2584)<br>ECDSA (Cert. A2584)<br>RSA (Cert. A2584)<br>Safe Primes (Cert. A2584) | DSA public key<br>DSA private key<br>ECDSA public key<br>ECDSA private key<br>RSA public key<br>RSA private key | User | DSA public key – GR<br>DSA private key – GR<br>ECDSA public key – GR<br>ECDSA private key – GR<br>RSA public key – GR<br>RSA private key – GR | API return value |
| Verify ECDSA public key | Verify an ECDSA public key | ECDSA (Cert. A2584) | ECDSA public key | User | ECDSA public key – W | API return value |
| Generate digital signature | Generate a digital signature | DSA (Cert. A2584)<br>ECDSA (Cert. A2584)<br>RSA (Cert. A2584)<br>SHS (Cert. A2584) | DSA private key<br>ECDSA private key<br>RSA private key | User | DSA private key – WE<br>ECDSA private key – WE<br>RSA private key – WE | API return value |
| Verify digital signature | Verify a digital signature | DSA (Cert. A2584)<br>ECDSA (Cert. A2584)<br>RSA (Cert. A2584)<br>SHS (Cert. A2584) | DSA public key<br>ECDSA public key<br>RSA public key | User | DSA public key – WE<br>ECDSA public key – WE<br>RSA public key – WE | API return value |
| Perform key wrap | Perform key wrap | KTS (Cert. A2584) | AES key<br>HMAC key | User | AES key – WE<br>HMAC key – WE | API return value |
| Perform key unwrap | Perform key unwrap | KTS (Cert. A2584) | AES key<br>HMAC key<br>Triple-DES key | User | AES key – WE<br>HMAC key – WE<br>Triple-DES key – WE | API return value |
| Compute shared secret | Compute DH/ECDH shared secret suitable for use as input to an internal TLS KDF or an external IKE KDF | KAS-ECC-SSC (Cert. A2584)<br>KAS-FFC-SSC (Cert. A2584) | DH public component<br>DH private component<br>ECDH public component<br>ECDH private component<br>TLS pre-master secret<br>IKE shared secret | User | DH public component – WE<br>DH private component – WE<br>ECDH public component – WE<br>ECDH private component – WE<br>TLS pre-master secret – GE<br>IKE shared secret – GR | API return value |
| Derive keys via TLS KDF | Derive TLS session and integrity keys | KDF (TLS) (Cert. A2584) | TLS pre-master secret<br>TLS master secret<br>AES key<br>HMAC key | User | TLS pre-master secret – WE<br>TLS master secret – GE<br>AES key – GR<br>HMAC key – GR | API return value |
| Derive key via PBKDF2 | Derive key from PBKDF2 | PBKDF (Cert. A2584) | Password<br>AES key | User | Password – WE<br>AES key – GR | API return value |

**Per FIPS 140-3 Implementation Guidance 2.4.C, the **Show Status**, **Zeroize**, and **Show Versioning Information** services do not require a service indicator.*

The module does not support a non-Approved mode of operation and offers no non-Approved services.

# 5. Software/Firmware Security

All software components within the cryptographic boundary are verified using an Approved integrity technique implemented within the cryptographic module itself. The module implements an HMAC SHA2-256 for the integrity test of each library file; failure of the integrity check for either library file will cause the module to enter a critical error state.

The module's integrity check is performed automatically at module instantiation (i.e., when the module is loaded into memory for execution) without action from the module operator. The CO can initiate the pre-operational tests and conditional CASTs on demand by re-instantiating the module or issuing the `FIPS_selftest()` API command.

The Cradlepoint Cryptographic Module is not delivered to end-users as a standalone offering. Rather, it is a pre-built integrated component of Cradlepoint's solutions. Cradlepoint does not provide end-users with any mechanisms to directly access the module, its source code, its APIs, or any information sent to/from the module. Thus, end-users have no ability to independently load the module onto target platforms. No configuration steps are required to be performed by end-users, and no end-user action is required to initialize the module for operation.

# 6. Operational Environment

The Cradlepoint Cryptographic Module comprises a software cryptographic library that executes in a modifiable operational environment.

The cryptographic module has control over its own SSPs. The process and memory management functionality of the host device's OS prevents unauthorized access to plaintext private and secret keys, intermediate key generation values and other SSPs by external processes during module execution. The module only allows access to SSPs through its well-defined API. The operational environment provides the capability to separate individual application processes from each other by preventing uncontrolled access to CSPs and uncontrolled modifications of SSPs regardless of whether this data is in the process memory or stored on persistent storage within the operational environment. Processes that are spawned by the module are owned by the module and are not owned by external processes/operators.

Please refer to section 2.1 of this document for a list/description of the applicable operational environments.

# 7. Physical Security

The cryptographic module is software module and does not include physical security mechanisms. Therefore, per *ISO/IEC 19790:2021* section 7.7.1, requirements for physical security are not applicable.

# 8. Non-Invasive Security

This section is not applicable. There are currently no approved non-invasive mitigation techniques referenced in *ISO/IEC 19790:2021* Annex F.

# 9. Sensitive Security Parameter Management

## 9.1 Keys and Other SSPs

The module supports the keys and other SSPs listed Table 9 and Table 10 below.

**Table 9 – Keys**

| Key/SSP Name/Type | Strength | Security Function and Cert. Number | Generation | Import / Export | Establishment | Storage | Zeroization | Use & Related Keys |
|---|---|---|---|---|---|---|---|---|
| AES key (CSP) | Between 128 and 256 bits | AES (CBC, CTR, ECB) (Cert. A2584)<br>KTS (Cert. A2584) | - | Imported in plaintext via API parameter<br>Never exported | Derived via TLS KDF | Not persistently stored by the module | Unload module; Remove power | Symmetric encryption, decryption |
| Triple-DES key (CSP) | - | Triple-DES (Cert. A2584)<br>KTS (Cert. A2584) | - | Imported in plaintext via API parameter<br>Never exported | - | Not persistently stored by the module | Unload module; Remove power | Symmetric decryption |
| HMAC key (CSP) | 160 bits (minimum) | HMAC (Cert. A2584)<br>KTS (Cert. A2584) | - | Imported in plaintext via API parameter<br>Never exported | Derived via TLS KDF | Not persistently stored by the module | Unload module; Remove power | Keyed hash |
| DSA private key (CSP) | 112 or 128 bits | DSA (Cert. A2584) | Generated internally via Approved DRBG | Imported in plaintext via API parameter<br>Exported in plaintext via API parameter | - | Not persistently stored by the module | Unload module; Remove power | Digital signature generation |
| DSA public key (PSP) | 112 or 128 bits | DSA (Cert. A2584) | Generated internally via approved DRBG | Imported in plaintext via API parameter<br>Exported in plaintext via API parameter | - | Not persistently stored by the module | Unload module; Remove power | Digital signature verification |
| ECDSA private key (CSP) | Between 112 and 256 bits | ECDSA (Cert. A2584) | Generated internally via approved DRBG | Imported in plaintext via API parameter<br>Exported in plaintext via API parameter | - | Not persistently stored by the module | Unload module; Remove power | Digital signature generation |
| ECDSA public key (PSP) | Between 112 and 256 bits | ECDSA (Cert. A2584) | Generated internally via approved DRBG | Imported in plaintext via API parameter<br>Exported in plaintext via API parameter | - | Not persistently stored by the module | Unload module; Remove power | Digital signature verification |

| Key/SSP Name/Type | Strength | Security Function and Cert. Number | Generation | Import / Export | Establishment | Storage | Zeroization | Use & Related Keys |
|---|---|---|---|---|---|---|---|---|
| RSA private key (CSP) | Between 80 and 150 bits | RSA (Cert. A2584) | Generated internally via approved DRBG | Imported in plaintext via API parameter<br>Exported in plaintext via API parameter | - | Not persistently stored by the module | Unload module; Remove power | Digital signature generation |
| RSA public key (PSP) | Between 80 and 150 bits | RSA (Cert. A2584) | Generated internally via approved DRBG | Imported in plaintext via API parameter<br>Exported in plaintext via API parameter | - | Not persistently stored by the module | Unload module; Remove power | Digital signature verification |
| DH private component (CSP) | 112 bits | KAS-FFC-SSC (Cert. A2584) | Generated internally via approved DRBG | Imported in plaintext via API parameter<br>Exported in plaintext via API parameter | - | Not persistently stored by the module | Unload module; Remove power | DH shared secret computation |
| DH public component (PSP) | 112 bits | KAS-FFC-SSC (Cert. A2584) | Generated internally via approved DRBG | Imported in plaintext via API parameter<br>Exported in plaintext via API parameter | - | Not persistently stored by the module | Unload module; Remove power | DH shared secret computation |
| ECDH private component (CSP) | Between 112 and 256 bits | KAS-ECC-SSC (Cert. A2584) | Generated internally via approved DRBG | Imported in plaintext via API parameter<br>Exported in plaintext via API parameter | - | Not persistently stored by the module | Unload module; Remove power | ECDH shared secret computation |
| ECDH public component (PSP) | Between 112 and 256 bits | KAS-ECC-SSC (Cert. A2584) | Generated internally via approved DRBG | Imported in plaintext via API parameter<br>Exported in plaintext via API parameter | - | Not persistently stored by the module | Unload module; Remove power | ECDH shared secret computation |

**Table 10 – Other SSPs**

| Key/SSP Name/Type | Strength | Security Function and Cert. Number | Generation | Import / Export | Establishment | Storage | Zeroization | Use & Related Keys |
|---|---|---|---|---|---|---|---|---|
| Passphrase (PSP) | - | PBKDF (Cert. A2584) | - | Imported in plaintext via API parameter<br>Never exported | - | Not persistently stored by the module | Unload module; Remove power | Input to PBKDF for key derivation |
| IKE shared secret (CSP) | - | KAS-FFC-SSC (Cert. A2584) | - | Never imported<br>Exported in plaintext via API parameter | Derived internally via DH shared secret computation | Not persistently stored by the module | Unload module; Remove power | Keying material suitable for use as input to an external IKE KDF |
| TLS pre-master secret (CSP) | - | KDF (TLS) (Cert. A2584) | - | Imported in plaintext via API parameter<br>Never exported | - | Not persistently stored by the module | Unload module; Remove power | Derivation of the TLS master secret |

| Key/SSP Name/Type | Strength | Security Function and Cert. Number | Generation | Import / Export | Establishment | Storage | Zeroization | Use & Related Keys |
|---|---|---|---|---|---|---|---|---|
| TLS master secret (CSP) | - | KDF (TLS) (Cert. A2584) | - | Never imported<br>Never exported | Derived internally using the TLS pre-master secret via TLS KDF | Not persistently stored by the module | Unload module; Remove power | Derivation of the AES/AES-GCM key and HMAC key used for securing TLS connections |
| DRBG entropy input (CSP) | - | DRBG (Cert. A2584) | - | Imported in plaintext via API parameter[43];<br>Never exported | - | Not persistently stored by the module | Unload module; Remove power | Entropy material for DRBG |
| DRBG seed (CSP) | - | DRBG (Cert. A2584) | Generated internally using nonce along with DRBG entropy input | Never imported<br>Never exported | - | Not persistently stored by the module | Unload module; Remove power | Seeding material for DRBG |
| DRBG 'V' value (CSP) | - | DRBG (Cert. A2584) | Generated internally | Never imported<br>Never exported | - | Not persistently stored by the module | Unload module; Remove power | State value for DRBG |
| DRBG 'Key' value (CSP) | - | DRBG (Cert. A2584) | Generated internally | Never imported<br>Never exported | - | Not persistently stored by the module | Unload module; Remove power | State value for DRBG |

## 9.2 DRBGs

The module implements the following Approved DRBG:

- Counter-based DRBG

This DRBG is used to generate random values at the request of the calling application. Outputs from this DRBG are also used as seeds in the generation of asymmetric key pairs.

The module implements the following non-Approved DRBGs (which are only available in the non-Approved mode of operation):

- Hash-based DRBG (non-compliant)
- HMAC-based DRBG (non-compliant)
- ANSI X9.31 RNG (non-Approved)

## 9.3 SSP Storage Techniques

There is no mechanism within the module's cryptographic boundary for the persistent storage of SSPs. The module stores DRBG state values for the lifetime of the DRBG instance. The module uses SSPs passed in on the stack by the calling application and does not store these SSPs beyond the lifetime of the API call.

[43] The module relies on entropy input received from the calling application, which is outside of the cryptographic boundary. Thus, there is no assurance of the minimum strength of generated keys.

## 9.4 SSP Zeroization Methods

As a software cryptographic module, there is no mechanism within the module boundary for the persistent storage of keys and CSPs. Maintenance, including protection and zeroization, of any keys and CSPs that exist outside the module's cryptographic boundary are the responsibility of the end-user. For the zeroization of keys in volatile memory, module operators can unload the module from memory or reboot/power-cycle the host device.

## 9.5 RGB Entropy Sources

The cryptographic module's entropy scheme follows the scenario given in *FIPS 140-3 Implementation Guidance* 9.3.A, section 2(b).

The module invokes a GET command to obtain entropy for random number generation (the module requests 256 bits of entropy from the calling application per request), and then passively receives entropy from the calling application while having no knowledge of the entropy source and exercising no control over the amount or the quality of the obtained entropy.

The calling application and its entropy sources are located within the physical perimeter of the module's operational environment but outside its cryptographic boundary. Thus, there is no assurance of the minimum strength of the generated SSPs.

# 10. Self-Tests

Both pre-operational and conditional self-tests are performed by the module. Pre-operational tests are performed between the time the cryptographic module is instantiated and before the module transitions to the operational state. Conditional self-tests are performed by the module during module operation when certain conditions exist. The following sections list the self-tests performed by the module, their expected error status, and the error resolutions.

## 10.1 Pre-Operational Self-Tests

The module performs the following pre-operational self-test(s):

- Software integrity test for libcrypto (using an HMAC SHA2-256 digest)
- Software integrity test for libssl (using an HMAC SHA2-256 digest)

If both pre-operational self-tests complete successfully, then the module sets an internal flag to indicate that it is in the Approved mode.

## 10.2 Conditional Self-Tests

The module performs the following conditional self-tests:

- Conditional cryptographic algorithm self-tests (CASTs)
    - AES ECB encrypt KAT[45] (128-bit length)
    - AES ECB decrypt KAT (128-bit length)
    - Triple-DES ECB encrypt KAT (3-Key)
    - Triple-DES ECB decrypt KAT (3-Key)
    - CTR_DRBG generate/instantiate/reseed KAT (256-bit AES, with derivation function)
    - PBKDF2 KAT
    - DSA sign KAT (2048-bit; SHA2-256)
    - DSA verify KAT (2048-bit; SHA2-256)
    - ECDSA sign KAT (P-224 and K-233 curves; SHA2-256)
    - ECDSA verify KAT (P-224 and K-233 curves; SHA2-256)
    - RSA verify KAT (2048-bit; SHA2-256; PKCS#1.5 scheme)
    - RSA sign KAT (2048-bit; SHA2-256; PKCS#1.5 scheme)
    - HMAC KATs (SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512)
    - SHA-1 KAT
    - SHA-2 KATs (SHA2-224, SHA2-256, SHA2-384, SHA2-512)
    - FFC DH Shared Secret “Z” Computation KAT (2048-bit)
    - ECC CDH Shared Secret “Z” Computation KAT (P-224 curve)
    - TLS 1.2 KDF KAT

[45] KAT – Known Answer Test

To ensure all CASTs are performed prior to the first operational use of the associated algorithm, all CASTs are performed during the module's initial power-up sequence. The SHA and HMAC KATs are performed prior to the pre-operational software integrity test; all other CASTs are executed after the successful completion of the software integrity test.

- Conditional pair-wise consistency tests (PCTs)
  - DSA sign/verify PCT
  - ECDSA sign/verify PCT
  - RSA sign/verify PCT
  - DH key generation PCT
  - ECDH key generation PCT

## 10.3 Self-Test Failure Handling

The module reaches the critical error state when any self-test fails. Upon test failure, the module immediately terminates the calling application's API call with a returned error code and sets an internal flag, signaling the error condition. For any subsequent request made by the calling application for cryptographic services, the module will return a failure indicator, thereby disabling all access to its cryptographic functions, sensitive security parameters (SSPs), and data output services while the error condition persists.

To recover, the module must be re-instantiated by the calling application. 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 operations, and the CO should contact Cradlepoint, Inc. for assistance.

# 11. Life-Cycle Assurance

The sections below describe how to ensure the module is operating in its validated configuration, including the following:

- Procedures for secure installation, initialization, startup, and operation of the module
- Maintenance requirements
- Administrator and non-Administrator guidance

**Operating the module without following the guidance herein (including the use of undocumented services) will result in non-compliant behavior and is outside the scope of this Security Policy**.

## 11.1 Secure Installation

The module is distributed as a package containing the binaries and HMAC digest files that the Crypto Officer is to install onto a target platform specified in section 6 or one where portability is maintained.

## 11.2 Initialization

This module is designed to support vendor applications, and these applications are the sole consumers of the cryptographic services provided by the module. No end-user action is required to initialize the module for operation; the calling application performs any actions required to initialize the module.

The pre-operational integrity test and cryptographic algorithm self-tests are performed automatically via a DEP when the module is loaded for execution, without any specific action from the calling application or the end-user. End-users have no means to short-circuit or bypass these actions. Failure of any of the initialization actions will result in a failure of the module to load for execution.

## 11.3 Startup

No startup steps are required to be performed by end-users.

## 11.4 Administrator Guidance

There are no specific management activities required of the CO role to ensure that the module runs securely. However, if any irregular activity is noticed or the module is consistently reporting errors, then Cradlepoint Customer Support should be contacted.

The following list provides additional guidance for the CO:

- The `fips_post_status()` API can be used to determine the module's operational status. A non-zero return value indicates that the module has passed all pre-operational self-tests and is currently in the Approved mode.

- The CO can initiate the pre-operational self-tests and conditional CASTs on demand for periodic testing of the module by re-instantiating the module or issuing the `FIPS_selftest()` API command.

- The `OpenSSL_version()` API can be used to obtain the module's versioning information. This information will include the module name and version, which can be correlated with the module's validation record.

## 11.5 Non-Administrator Guidance

The following list provides additional policies for non-administrators:

- The module uses PBKDF2 option 1a from section 5.4 of *NIST SP 800-132*. The iteration count shall be selected as large as possible, as long as the time required to generate the resultant key is acceptable for module operators. The minimum iteration count shall be 1000.

  The length of the password/passphrase used in the PBKDF shall be of at least 20 characters, and shall consist of lower-case, upper-case, and numeric characters. The upper bound for the probability of guessing the value is estimated to be $1/62^{20} = 10^{-36}$, which is less than $2^{-112}$.

  Keys derived from passwords/passphrases, as shown in *NIST SP 800-132*, may only be used in storage applications.

- The cryptographic module's services are designed to be provided to a calling application. Excluding the use of the NIST-defined elliptic curves as trusted third-party domain parameters, all other assurances from *FIPS PUB 186-4* (including those required of the intended signatory and the signature verifier) are outside the scope of the module and are the responsibility of the calling application.

- The module performs assurances for its key agreement schemes as specified in the following sections of *NIST SP 800-56Arev3:*

  - Section 5.5.2 (for assurances of domain parameter validity)
  - Section 5.6.2.1 (for assurances required by the key pair owner)

  The module includes the capability to provide the required recipient assurance of ephemeral public key validity specified in section 5.6.2.2.2 of *NIST SP 800-56Arev3*. However, since public keys from other modules are not received directly by this module (those keys are received by the calling application), the module has no knowledge of when a public key is received. Invocation of the proper module services to validate another module's public key is the responsibility of the calling application.

# 12. Mitigation of Other Attacks

This section is not applicable. The module does not claim to mitigate any attacks beyond the FIPS 140-3 Level 1 requirements for this validation.

# Appendix A. Acronyms and Abbreviations

Table 11 provides definitions for the acronyms and abbreviations used in this document.

**Table 11 – Acronyms and Abbreviations**

| Term | Definition |
|---|---|
| AES | Advanced Encryption Standard |
| ANSI | American National Standards Institute |
| API | Application Programming Interface |
| CAST | Cryptographic Algorithm Self-Test |
| CBC | Cipher Block Chaining |
| CCCS | Canadian Centre for Cyber Security |
| CCM | Counter with Cipher Block Chaining - Message Authentication Code |
| CFB | Cipher Feedback |
| CKG | Cryptographic Key Generation |
| CMAC | Cipher-Based Message Authentication Code |
| CMVP | Cryptographic Module Validation Program |
| CO | Cryptographic Officer |
| CPU | Central Processing Unit |
| CSP | Critical Security Parameter |
| CTR | Counter |
| CVL | Component Validation List |
| DEP | Default Entry Point |
| DES | Data Encryption Standard |
| DH | Diffie-Hellman |
| DRBG | Deterministic Random Bit Generator |
| DSA | Digital Signature Algorithm |
| ECB | Electronic Code Book |
| ECC | Elliptic Curve Cryptography |
| CDH | Elliptic Curve Cryptography Cofactor Diffie-Hellman |
| ECDH | Elliptic Curve Diffie-Hellman |
| ECDSA | Elliptic Curve Digital Signature Algorithm |
| EMI/EMC | Electromagnetic Interference /Electromagnetic Compatibility |
| FFC | Finite Field Cryptography |
| FIPS | Federal Information Processing Standard |
| GCM | Galois/Counter Mode |

| Term | Definition |
|---|---|
| GMAC | Galois Message Authentication Code |
| GPC | General-Purpose Computer |
| HMAC | (keyed-) Hash Message Authentication Code |
| KAS | Key Agreement Scheme |
| KAT | Known Answer Test |
| KTS | Key Transport Scheme |
| KW | Key Wrap |
| KWP | Key Wrap with Padding |
| MD | Message Digest |
| NIST | National Institute of Standards and Technology |
| OCB | Offset Codebook |
| OFB | Output Feedback |
| OS | Operating System |
| PBKDF | Password-Based Key Derivation Function |
| PCT | Pairwise Consistency Test |
| PKCS | Public Key Cryptography Standard |
| PSS | Probabilistic Signature Scheme |
| PUB | Publication |
| RC | Rivest Cipher |
| RNG | Random Number Generator |
| RSA | Rivest Shamir Adleman |
| SHAKE | Secure Hash Algorithm KECCAK |
| SHA | Secure Hash Algorithm |
| SHS | Secure Hash Standard |
| SP | Special Publication |
| SSC | Shared Secret Computation |
| TDES | Triple Data Encryption Standard |
| TLS | Transport Layer Security |
| XEX | XOR Encrypt XOR |
| XTS | XEX-Based Tweaked-Codebook Mode with Ciphertext Stealing |

Prepared by:
**Corsec Security, Inc.**

12600 Fair Lakes Circle, Suite 210
Fairfax, VA 22033
United States of America

Phone: +1 703 267 6050
Email: info@corsec.com
http://www.corsec.com