Corsec Security, Inc.

CorSSL™ FIPS Object Module

Software Version: 2.0.16.001

FIPS 140-3 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 0.3

Prepared by:

Corsec Security, Inc.
12600 Fair Lakes Circle, Suite 210
Fairfax, VA 22033
United States of America

Phone: +1 703 267 6050
www.corsec.com

Table of Contents

1. General 5
1.1 Overview 5
1.2 Security Levels 5

2. Cryptographic Module Specification 7
2.1 Description 7
2.2 Tested and Vendor Affirmed Module Version and Identification 9
2.3 Excluded Components 10
2.4 Modes of Operation 10
2.5 Algorithms 11
2.6 Security Function Implementations 14
2.7 Algorithm Specific Information 17
2.8 RNG and Entropy 18
2.9 Key Generation 19
2.10 Key Establishment 19
2.11 Industry Protocols 20
2.12 Additional Information 20

3. Cryptographic Module Interfaces 21
3.1 Ports and Interfaces 21

4. Roles, Services, and Authentication 22
4.1 Authentication Methods 22
4.2 Roles 22
4.3 Approved Services 22
4.4 Non-Approved Services 26
4.5 External Software/Firmware Loaded 27

5. Software/Firmware Security 28
5.1 Integrity Techniques 28
5.2 Initiate on Demand 28

6. Operational Environment 29
6.1 Operational Environment Type and Requirements 29

7. Physical Security 30

8. Non-Invasive Security 31

9. Sensitive Security Parameters Management 32
9.1 Storage Areas 32
9.2 SSP Input-Output Methods 32
9.3 SSP Zeroization Methods 32
9.4 SSPs 33
9.5 Transitions 37
9.6 Additional Information 38

10. Self-Tests 39

10.1 Pre-Operational Self-Tests 39
10.2 Conditional Self-Tests 39
10.3 Periodic Self-Test Information 42
10.4 Error States 43
10.5 Operator Initiation of Self-Tests 43

11. Life-Cycle Assurance 44
11.1 Installation, Initialization, and Startup Procedures 44
11.2 Administrator Guidance 44
11.3 Non-Administrator Guidance 45
11.4 Design and Rules 45
11.5 End of Life 45

12. Mitigation of Other Attacks 46

Appendix A. Acronyms and Abbreviations 47

List of Tables

Table 1: Security Levels 6
Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets) 10
Table 3: Tested Operational Environments - Software, Firmware, Hybrid 10
Table 4: Modes List and Description 10
Table 5: Approved Algorithms 14
Table 6: Vendor-Affirmed Algorithms 14
Table 7: Non-Approved, Allowed Algorithms 14
Table 8: Security Function Implementations 17
Table 9: Ports and Interfaces 21
Table 10: Roles 22
Table 11: Approved Services 26
Table 12: Storage Areas 32
Table 13: SSP Input-Output Methods 32
Table 14: SSP Zeroization Methods 33
Table 15: SSP Table 1 35
Table 16: SSP Table 2 37
Table 17: Pre-Operational Self-Tests 39
Table 18: Conditional Self-Tests 41
Table 19: Pre-Operational Periodic Information 42
Table 20: Conditional Periodic Information 43
Table 21: Error States 43
Table 22. Acronyms and Abbreviations 47

List of Figures

Figure 1. Module Block Diagram (with Cryptographic Boundary) 8
Figure 2. GPC Block Diagram 9

1. General

1.1 Overview

1.1.1 Abstract

This is a non-proprietary Cryptographic Module Security Policy for the CorSSL™ FIPS Object Module (version: 2.0.16.001) from Corsec Security, Inc. (Corsec). This Security Policy describes how the CorSSL™ FIPS Object 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 a secure FIPS-Approved mode of operation. This policy was prepared as part of the Level 1 FIPS 140-3 validation of the module. The CorSSL™ FIPS Object Module is referred to in this document as CorSSL FOM or the module.

1.1.2 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 Corsec website www.corsec.com contains information on the full line of services and solutions from Corsec.
- 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.

1.1.3 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.

1.2 Security Levels

The CorSSL™ FIPS Object Module is validated at the FIPS 140-3 section levels shown in the table below.

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
	Overall Level	1

Table 1: Security Levels

The module has an overall security level of 1.

2. Cryptographic Module Specification

2.1 Description

2.1.1 Purpose and Use

Corsec Security, Inc is a privately owned company dedicated to assisting organizations through the security certification and validation process. Over the past 22 years, Corsec has grown significantly, becoming a global leader in product and corporate security, offering critical guidance and expertise to meet important business challenges in product security and third-party certifications and security validations, including FIPS 140-2, FIPS 140-3, Common Criteria, and the DoDIN APL[1].

Corsec's certification methodology helps open doors to new markets and increase revenue for clients with products ranging from mobile phones to satellites. Corsec's broad knowledge safeguards against common pitfalls and thwarts delays, translating to a swift and seamless path to certification. Corsec has created the benchmark for providing business leaders with fast, flexible access to industry knowledge on security certifications and validations.

The CorSSL™ FIPS Object Module (also called "CorSSL FOM") v2.0.16.001 is a software library providing a C language API[2] for use by other applications requiring cryptographic functionality. The CorSSL FOM offers symmetric encryption/decryption, digital signature generation/verification, hashing, cryptographic key generation, random number generation, message authentication, and key establishment functions to secure data-at-rest/data-in-flight and to support industry-standard secure communications protocols.

The module is a software module based on the OpenSSL FIPS Object Module (FOM) 2.0.16. The module is compiled into object form and then linked to a "FIPS-capable" instance of the OpenSSL cryptographic library at build-time. The larger library can then be linked to a calling application. The module provides engineering teams with a completely compatible cryptographic engine, allowing quick "drop-in" replacement into any existing OpenSSL FOM-based solutions. The CorSSL FOM does not modify the OpenSSL interface, maintaining complete compatibility, and eliminating engineering development time to meet FIPS 140-3 requirements.

2.1.2 Module Type

The module is a Software module.

2.1.3 Module Embodiment

The module has a Multi-Chip Standalone embodiment.

[1] DoDIN APL – Department of Defense Information Network Approved Product List
[2] API – Application Programming Interface

2.1.4 Cryptographic Boundary

The cryptographic boundary is the contiguous perimeter that surrounds all memory-mapped functionality provided by the module when loaded and stored in the host platform's memory. The module's cryptographic boundary consists of all functionalities contained within the module's compiled source code. This comprises a cryptographic primitives library file called fipscanister.o. The module image includes an embedded HMAC SHA-1 MAC value for verifying the module's integrity at runtime.

Figure 1 below shows the logical block diagram of the module executing in memory and its interactions with surrounding software components, as well as the module's physical perimeter and cryptographic boundary.

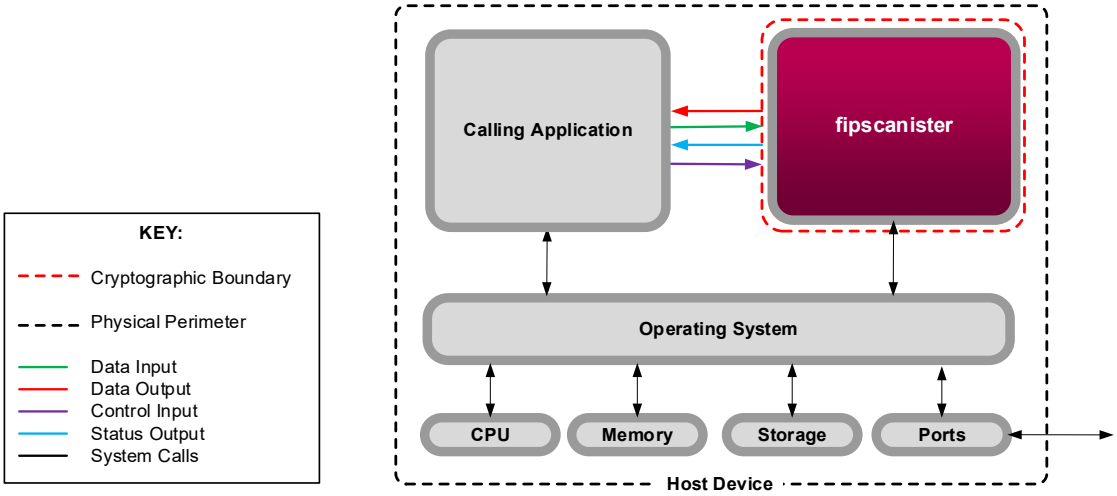

Figure 1. Module Block Diagram (with Cryptographic Boundary)

2.1.5 Tested Operational Environment's Physical Perimeter (TOEPP)

As a software cryptographic module, the module has no physical components. The physical perimeter of the cryptographic boundary is defined by the hard enclosure of each host platform on which the module is installed. Figure 2 illustrates a block diagram of a typical GPC and the module's physical perimeter.

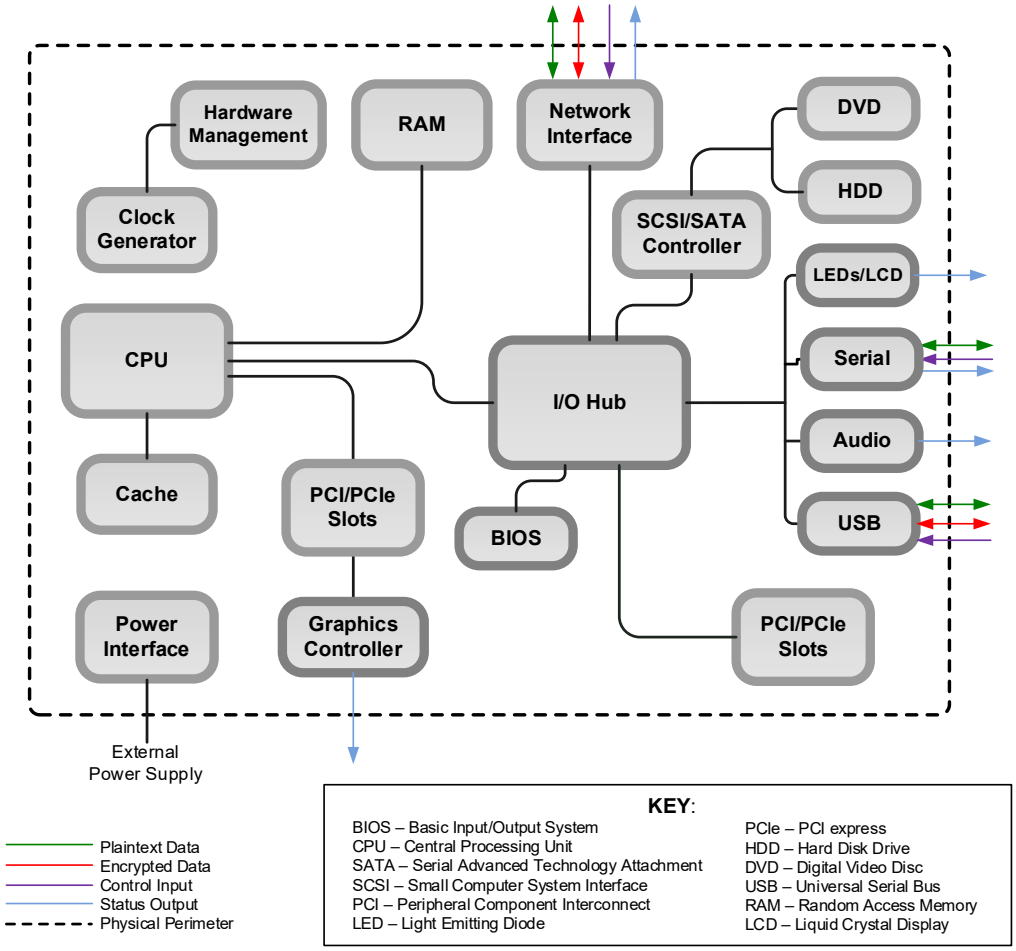

Figure 2. GPC Block Diagram

The module is entirely contained within the physical perimeter.

2.2 Tested and Vendor Affirmed Module Version and Identification

2.2.1 Tested Module Identification – Hardware

The module does not include any hardware.

2.2.2 Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)

The table below lists the executable code sets of the module.

Package or File Name	Software/ Firmware Version	Features	Integrity Test
fipscanister.o	2.0.16.001		Yes

Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)

2.2.3 Tested Module Identification – Hybrid Disjoint Hardware

This section is only applicable to hybrid modules.

2.2.4 Tested Operational Environments – Software, Firmware, Hybrid

The module was tested and found to be compliant with FIPS 140-3 requirements on the environments listed in the table below.

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Debian 9	Dell PowerEdge R440	Intel Xeon Silver 4214R	Yes		2.0.16.001
Debian 9	Dell PowerEdge R440	Intel Xeon Silver 4214R	No		2.0.16.001

Table 3: Tested Operational Environments - Software, Firmware, Hybrid

The module is designed to utilize the AES-NI extended instruction set when available by the host platform's CPU for the processor-based algorithm acceleration (PAA) of its AES and SHA implementations.

2.2.5 Vendor-Affirmed Operational Environments – Software, Firmware, Hybrid

There are no vendor-affirmed operational environments claimed.

2.3 Excluded Components

The module does not exclude any components from the requirements.

2.4 Modes of Operation

2.4.1 Modes List and Description

The module supports only the Approved mode of operation. This operational mode is described in the table below.

Mode Name	Description	Type	Status Indicator
Approved	Once all pre-operational self-tests have completed successfully, the module supports an Approved mode of operation only.	Approved	FIPS_mode() returns non-zero value

Table 4: Modes List and Description

2.5 Algorithms

2.5.1 Approved Algorithms

The module employs cryptographic algorithm implementations from the following source:

- CorSSL FOM (Cert. A3356)

Validation certificates for each Approved algorithm are listed in the table below.

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A3356	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A3356	Key Length - 128, 192, 256 Tag Length - 112, 128, 32, 48, 64, 80, 96 IV Length - IV Length: 56-104 Increment 8 Payload Length - Payload Length: 0-256 Increment 8 AAD Length - AAD Length: 0-524288 Increment 8	SP 800-38C
AES-CFB1	A3356	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB128	A3356	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A3356	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A3356	Direction - Generation, Verification Key Length - 128, 192, 256 MAC Length - MAC Length: 16-128 Increment 8 Message Length - Message Length: 0-65536 Increment 8	SP 800-38B
AES-CTR	A3356	Direction - Decrypt, Encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 8-128 Increment 8 Supports Counter larger than maximum value - Yes Incremental Counter - Yes Counter Tests Performed - Yes	SP 800-38A
AES-ECB	A3356	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A3356	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 Payload Length - Payload Length: 504, 512, 1016, 1024 AAD Length - AAD Length: 0, 504, 512, 1016, 1024	SP 800-38D
AES-OFB	A3356	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-XTS Testing Revision 2.0	A3356	Direction - Decrypt, Encrypt Key Length - 128, 256 Payload Length - Payload Length: 128-65536 Increment 8 Tweak Mode - Hex Data Unit Length Matches Payload Length - Yes	SP 800-38E

Algorithm	CAVP Cert	Properties	Reference
Counter DRBG	A3356	Prediction Resistance - No, Yes Supports Reseed - Yes Mode - AES-128, AES-192, AES-256 Derivation Function Enabled - Yes Additional Input - Additional Input: 0-256 Increment 256 Entropy Input - Entropy Input: 128-256 Increment 128, Entropy Input: 256-512 Increment 128 Nonce - Nonce: 128 Personalization String Length - Personalization String Length: 0-256 Increment 256 Returned Bits - 256	SP 800-90A Rev. 1
DSA KeyGen (FIPS186-4)	A3356	L - 2048, 3072 N - 224, 256	FIPS 186-4
DSA PQGGen (FIPS186-4)	A3356	P/Q Generation Methods - Probable G Generation Methods - Canonical L - 2048, 3072 N - 224, 256 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
DSA PQGVer (FIPS186-4)	A3356	P/Q Generation Methods - Probable G Generation Methods - Canonical L - 1024, 2048, 3072 N - 160, 224, 256 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
DSA SigGen (FIPS186-4)	A3356	L - 2048, 3072 N - 224, 256 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
DSA SigVer (FIPS186-4)	A3356	L - 1024, 2048, 3072 N - 160, 224, 256 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
ECDSA KeyGen (FIPS186-4)	A3356	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Secret Generation Mode - Testing Candidates	FIPS 186-4
ECDSA KeyVer (FIPS186-4)	A3356	Curve - B-163, B-233, B-283, B-409, B-571, K-163, K-233, K-283, K-409, K-571, P-192, P-224, P-256, P-384, P-521	FIPS 186-4
ECDSA SigGen (FIPS186-4)	A3356	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A3356	Curve - B-163, B-233, B-283, B-409, B-571, K-163, K-233, K-283, K-409, K-571, P-192, P-224, P-256, P-384, P-521 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
Hash DRBG	A3356	Prediction Resistance - No, Yes Supports Reseed - Yes Mode - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512 Entropy Input - Entropy Input: 128-256 Increment 64, Entropy Input: 192-256 Increment 64, Entropy Input: 256-320 Increment 64 Nonce - Nonce: 128-160 Increment 32, Nonce: 96-128 Increment 32 Personalization String Length - Personalization String Length: 0-256 Increment 128 Additional Input - Additional Input: 0-256 Increment 128 Returned Bits - 160, 224, 256, 384, 512	SP 800-90A Rev. 1

Algorithm	CAVP Cert	Properties	Reference
HMAC DRBG	A3356	Prediction Resistance - No, Yes Supports Reseed - Yes Mode - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512 Entropy Input - Entropy Input: 160-256 Increment 32, Entropy Input: 192-256 Increment 64, Entropy Input: 256-512 Increment 64, Entropy Input: 384-512 Increment 64, Entropy Input: 512-1024 Increment 64 Nonce - Nonce: 128, Nonce: 64, Nonce: 96 Personalization String Length - Personalization String Length: 0-256 Increment 128 Additional Input - Additional Input: 0-256 Increment 128 Returned Bits - 160, 224, 256, 384, 512	SP 800-90A Rev. 1
HMAC-SHA-1	A3356	MAC - MAC: 32-160 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A3356	MAC - MAC: 32-224 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A3356	MAC - MAC: 32-256 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A3356	MAC - MAC: 32-384 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A3356	MAC - MAC: 32-512 Increment 8 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A3356	Domain Parameter Generation Methods - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KTS-IFC	A3356	Function - partialVal IUT ID - CAFECAFE Modulo - 2048, 3072, 4096, 6144, 8192 Key Generation Methods - rsakpg1-basic Fixed Public Exponent - 010001 Scheme - KTS-OAEP-basic - KAS Role - initiator, responder Key Transport Method - Hash Algorithms - SHA2-224, SHA2-256, SHA2-384, SHA2-512 Supports Null Associated Data - Yes Associated Data Pattern - uPartyInfo||vPartyInfo Associated Data Encoding - concatenation Key Length - 768	SP 800-56B Rev. 2
RSA KeyGen (FIPS186-4)	A3356	Key Generation Mode - B.3.3 Modulo - 2048, 3072, 4096 Primality Tests - Table C.2 Info Generated By Server - No Public Exponent Mode - Fixed Fixed Public Exponent - 010001 Private Key Format - Standard	FIPS 186-4
RSA SigGen (FIPS186-4)	A3356	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA2-256	FIPS 186-4
RSA SigVer (FIPS186-4)	A3356	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA-1 Public Exponent Mode - Fixed Fixed Public Exponent - 010001	FIPS 186-4

Algorithm	CAVP Cert	Properties	Reference
SHA-1	A3356	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-224	A3356	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-256	A3356	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-384	A3356	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-512	A3356	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4

Table 5: Approved Algorithms

2.5.2 Vendor Affirmed Algorithms

The vendor affirms the following cryptographic security methods in the table below:

Name	Properties	Implementation	Reference
CKG Section 4 Example 1	Key type:Asymmetric	N/A	SP 800-133 Rev. 2 Section 4 Example 1

Table 6: Vendor-Affirmed Algorithms

2.5.3 Non-Approved, Allowed Algorithms

Name	Properties	Implementation	Reference
AES-ECB	Direction:Unwrapping	CorSSL FOM	FIPS 140-3 Implementation Guidance D.G Key Transport Methods
AES-CBC	Direction:Unwrapping	CorSSL FOM	FIPS 140-3 Implementation Guidance D.G Key Transport Methods
AES-OFB	Direction:Unwrapping	CorSSL FOM	FIPS 140-3 Implementation Guidance D.G Key Transport Methods
AES-CFB1	Direction:Unwrapping	CorSSL FOM	FIPS 140-3 Implementation Guidance D.G Key Transport Methods
AES-CFB8	Direction:Unwrapping	CorSSL FOM	FIPS 140-3 Implementation Guidance D.G Key Transport Methods
AES-CFB128	Direction:Unwrapping	CorSSL FOM	FIPS 140-3 Implementation Guidance D.G Key Transport Methods
AES-CTR	Direction:Unwrapping	CorSSL FOM	FIPS 140-3 Implementation Guidance D.G Key Transport Methods

Table 7: Non-Approved, Allowed Algorithms

2.5.4 Non-Approved, Allowed Algorithms with No Security Claimed

N/A for this module.

2.5.5 Non-Approved, Not Allowed Algorithms

N/A for this module.

2.6 Security Function Implementations

The table below lists the security function implementations for this module.

Name	Type	Description	Properties	Algorithms
AES for Symmetric Encryption/Decryption	BC-UnAuth	Uses any approved mode of unauthenticated AES for symmetric encryption and decryption.	Property:Publication: SP 800-38A	AES-ECB: (A3356) AES-CBC: (A3356) AES-OFB: (A3356) AES-CFB1: (A3356) AES-CFB8: (A3356) AES-CFB128: (A3356) AES-CTR: (A3356)
AES-GCM for Authenticated Symmetric Encryption/Decryption	BC-Auth	Uses AES-GCM for authenticated symmetric encryption and decryption.	Property:Publication: SP 800-38D	AES-GCM: (A3356) AES-CTR: (A3356)
AES-CCM for Authenticated Symmetric Encryption/Decryption	BC-Auth	Uses AES-CCM for authenticated symmetric encryption and decryption.	Property:Publication: SP 800-38C	AES-CCM: (A3356) AES-CTR: (A3356)
AES-XTS for Symmetric Encryption/Decryption	BC-UnAuth	Uses AES-XTS for symmetric encryption and decryption.	Property:Publication: SP 800-38E	AES-XTS Testing Revision 2.0: (A3356) AES-ECB: (A3356)
SHA for Message Digest	SHA	Uses SHA for generating message digests.	Property:Publication: FIPS 180-4	SHA-1: (A3356) SHA2-224: (A3356) SHA2-256: (A3356) SHA2-384: (A3356) SHA2-512: (A3356)
DRBG	DRBG	Uses DRBG for generating random bits.	Property:Publication: SP 800-90A	Counter DRBG: (A3356) AES-CTR: (A3356) HMAC DRBG: (A3356) HMAC-SHA-1: (A3356) HMAC-SHA2-224: (A3356) HMAC-SHA2-256: (A3356) HMAC-SHA2-384: (A3356) HMAC-SHA2-512: (A3356) Hash DRBG: (A3356) SHA-1: (A3356) SHA2-224: (A3356) SHA2-256: (A3356) SHA2-384: (A3356) SHA2-512: (A3356)
HMAC for Keyed Hash	MAC	Uses HMAC for generating keyed hash.	Property:Publication: FIPS 198-1	HMAC-SHA-1: (A3356) HMAC-SHA2-224: (A3356) HMAC-SHA2-256: (A3356) HMAC-SHA2-384: (A3356) HMAC-SHA2-512: (A3356) SHA-1: (A3356) SHA2-224: (A3356) SHA2-256: (A3356) SHA2-384: (A3356) SHA2-512: (A3356)

Name	Type	Description	Properties	Algorithms
AES-CMAC for Symmetric Digest	MAC	Uses AES-CMAC for generating symmetric digest.	Property:Publication: SP 800-38B	AES-CMAC: (A3356) AES-CBC: (A3356)
DSA for Parameter Generation	AsymKeyPair-DomPar	Generates DSA domain parameters.	Property:Publication: FIPS 186-4	DSA PQGGen (FIPS186-4): (A3356) Counter DRBG: (A3356) SHA2-224: (A3356) SHA2-256: (A3356) SHA2-384: (A3356) SHA2-512: (A3356)
DSA for Parameter Verification	AsymKeyPair-DomPar	Verifies DSA domain parameters.	Property:Publication: FIPS 186-4	DSA PQGVer (FIPS186-4): (A3356) SHA-1: (A3356) SHA2-224: (A3356) SHA2-256: (A3356) SHA2-384: (A3356) SHA2-512: (A3356)
DSA for Key Generation	AsymKeyPair-KeyGen	Generates DSA private key and DSA public key.	Property:Publication: FIPS 186-4	DSA KeyGen (FIPS186-4): (A3356) Counter DRBG: (A3356)
DSA for Signature Generation	DigSig-SigGen	Uses the DSA private key to generate signatures.	Property:Publication: FIPS 186-4	DSA SigGen (FIPS186-4): (A3356) Counter DRBG: (A3356) SHA2-224: (A3356) SHA2-256: (A3356) SHA2-384: (A3356) SHA2-512: (A3356)
DSA for Signature Verification	DigSig-SigVer	Uses the DSA public key to verify signatures.	Property:Publication: FIPS 186-4	DSA SigVer (FIPS186-4): (A3356) SHA-1: (A3356) SHA2-224: (A3356) SHA2-256: (A3356) SHA2-384: (A3356) SHA2-512: (A3356)
ECDSA for Key Generation	AsymKeyPair-KeyGen	Generates ECDSA private key and ECDSA public key.	Property:Publication: FIPS 186-4	ECDSA KeyGen (FIPS186-4): (A3356) Counter DRBG: (A3356)
ECDSA for Key Verification	AsymKeyPair-KeyVer	Verifies the ECDSA public key.	Property:Publication: FIPS 186-4	ECDSA KeyVer (FIPS186-4): (A3356)
ECDSA for Signature Generation	DigSig-SigGen	Uses the ECDSA private key to generate signatures.	Property:Publication: FIPS 186-4	ECDSA SigGen (FIPS186-4): (A3356) Counter DRBG: (A3356) SHA2-224: (A3356) SHA2-256: (A3356) SHA2-384: (A3356) SHA2-512: (A3356)
ECDSA for Signature Verification	DigSig-SigVer	Uses the ECDSA public key to verify signatures.	Property:Publication: FIPS 186-4	ECDSA SigVer (FIPS186-4): (A3356) SHA2-224: (A3356) SHA2-256: (A3356) SHA2-384: (A3356) SHA2-512: (A3356)
RSA for Key Generation	AsymKeyPair-KeyGen	Generates RSA private key and RSA public key.	Property:Publication: FIPS 186-4	RSA KeyGen (FIPS186-4): (A3356) Counter DRBG: (A3356)

Name	Type	Description	Properties	Algorithms
RSA for Signature Generation	DigSig-SigGen	Uses the RSA private key to generate signatures.	Property:Publication: FIPS 186-4	RSA SigGen (FIPS186-4): (A3356) Counter DRBG: (A3356) SHA2-224: (A3356) SHA2-256: (A3356) SHA2-384: (A3356) SHA2-512: (A3356)
RSA for Signature Verification	DigSig-SigVer	Uses the RSA public key to verify signatures.	Property:Publication: FIPS 186-4	RSA SigVer (FIPS186-4): (A3356) SHA-1: (A3356) SHA2-224: (A3356) SHA2-256: (A3356) SHA2-384: (A3356) SHA2-512: (A3356)
ECDH Shared Secret Computation	KAS-SSC	Uses ECDH to compute shared secret.	Property:Publication: SP 800-56A Rev. 3	KAS-ECC-SSC Sp800-56Ar3: (A3356) ECDSA KeyGen (FIPS186-4): (A3356) ECDSA KeyVer (FIPS186-4): (A3356)
RSA for Asymmetric Encryption/Decryption	AsymKeyPair-KeyGen	Uses RSA for asymmetric encryption and decryption	Property:Publication: SP 800-56B Rev. 2	RSA KeyGen (FIPS186-4): (A3356) Counter DRBG: (A3356) KTS-IFC: (A3356)

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

The module has the following algorithm specific information.

2.7.1 AES-GCM

The module supports internal IV generation using its Approved Counter-based DRBG. The IV is at least 96 bits in length per section 8.2.2 of *NIST SP 800-38D*, and the Approved DRBG generates outputs such that the (key, IV) pair collision probability is less than 2^{-32} per section 8 of *NIST SP 800-38D*. This complies with the AES GCM (key/IV) pair uniqueness requirements specified in scenario 2 of *FIPS 140-3 IG* C.H.

In case the module's power is lost and then restored, the calling application is responsible for ensuring that a new key for use with the AES-GCM encryption/decryption shall be established. This condition is not enforced by the module but is met implicitly. The module does not retain any state across resets or power-cycles, and AES-GCM key/IVs are not stored in non-volatile persistent memory (i.e., disk). Hence, no reconnection can occur without a fresh key establishment operation and the associated SSPs.

When a GCM IV is used for decryption, the responsibility for the IV generation lies with the party that performs the AES GCM encryption.

2.7.2 AES-XTS

The length of a single data unit encrypted or decrypted with the AES-XTS shall not exceed 2^{20} AES blocks; that is, 16 MB of data per AES-XTS instance. An XTS instance is defined in section 4 of *NIST SP 800-38E*.

For Approved use, AES XTS keys (i.e., Key_1 and Key_2) entered into the module shall be generated and/or established independently according to Section 6.3 of *NIST SP 800-133 Rev. 2*. In compliance with *FIPS 140-3 IG* C.I, the module implements a check to ensure that the two keys are not identical.

As specified in *NIST SP 800-132*, AES-XTS mode shall only be used for the cryptographic protection of data on storage devices. The AES-XTS shall not be used for other purposes, such as the encryption of data in transit.

2.7.3 KAS

The module does not establish SSPs using an Approved key agreement scheme (KAS). However, it does offer some or all of the underlying KAS cryptographic functionality to be used by a calling application as part of an Approved KAS. Refer to section 2.10.1 for additional details.

2.7.4 KTS

The module does not establish SSPs using an approved key transport scheme (KTS). However, it does offer approved authentication algorithms that can be used by an external operation/application as part of an approved KTS. Refer to section 2.10.2 below for additional details.

2.7.5 SHA

The module provides ShA-1 for use by a calling application as a stand-alone security function for hashing. Internally, use of SHA-1 is limited to the following:

- as an underlying PRG for HMAC SHA-1
- as an underlying hash function for HASH_DRBG and HMAC_DRBG
- to support legacy DSA, ECDSA, and RSA digital signature verification functions
- to support DSA parameter generation functions

The use of SHA-1 for applying cryptographic protection for non-digital signature applications is deprecated through December 31, 2030. Please refer to section 9.5 below for details regarding algorithm transition dates.

2.8 RNG and Entropy

The cryptographic module supports the Hash_DRBG, HMAC_DRBG, and CTR_DRBG mechanisms and performs the DRBG health tests as defined in section 11.3 of *NIST SP 800-90A Rev. 1*.

The module does not have an internal entropy source. Rather, 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 operational environment inside the module's physical perimeter but outside the cryptographic boundary. **There is no assurance of the minimum strength of generated SSPs (e.g., keys).**

2.9 Key Generation

In compliance with section 4, example 1 of *NIST SP 800-133 Rev. 2*, the cryptographic module uses its Approved DRBGs to generate seeds used for asymmetric key generation. The generated seed is an unmodified output from the DRBG.

2.10 Key Establishment

The cryptographic module receives SSPs from outside of its boundary for use within an Approved algorithm. **There is no assurance of minimum security of SSPs (e.g., keys or bit strings) that are externally loaded, or of SSPs established with externally loaded SSPs.**

Additionally, the module provides the cryptographic primitives necessary to support key agreement schemes and key transport methods utilized by the calling application to establish keys.

2.10.1 Key Agreement Information

The module offers as a service to a calling application the CAVP-tested KAS SSC that maps to *FIPS 140-3 IG* D.F Scenario 2, path (1) without establishing a key/SSP to be used by the module for cryptographic protection:

- KAS-ECC-SSC

The module performs assurances for its key agreement schemes as specified in the following sections of *NIST SP 800-56A Rev. 3:*

- 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 800-56A Rev. 3*. 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.

Key confirmation is not supported by the module.

2.10.2 Key Transport Methods

The module offers as a service to a calling application CAVP-tested algorithms specified in the list below without establishing a key/SSP to be used by the module for cryptographic protection.

- To support authenticated encryption/decryption as a unified service:
  - AES-CCM
  - AES-GCM

- To support unauthenticated encryption/decryption and authentication as separate services:
  - AES + CMAC
  - AES + HMAC

- To support unauthenticated encryption/decryption as a legacy service:
  - AES key unwrap (legacy)[3]
  - RSA key transport

The module performs assurances for its RSA-based key transport scheme as specified in the following sections of *NIST SP 800-56B Rev. 2:*

- Section 6.4.1 (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 6.4.2 of *NIST SP 800-56B Rev. 2*. 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.

2.11 Industry Protocols

The module does not implement any industry protocols.

2.12 Additional Information

Algorithms designated as "legacy" can only be used on data that was generated prior to the associated Legacy Dates specified in *FIPS 140-3* IG C.M.

[3] Per FIPS 140-3 IG D.G, key unwrapping using any Approved mode of AES is an allowed key transport method in the Approved mode.

3. Cryptographic Module Interfaces

3.1 Ports and Interfaces

The module supports the following logical interfaces:

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

As a software library, the cryptographic module has no direct access to any of the host platform's physical ports; it communicates only to the calling application via its well-defined API. The table below contains a mapping of the physical and logical interfaces of the module.

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	Includes data to be encrypted/decrypted/signed/verified/hashed, keys to be used in cryptographic services, random seed material for the module's DRBG, and keying material to be used as input to key establishment services.
N/A	Data Output	Includes data that has been encrypted, digital signatures, hashes, random values generated by the module's DRBG, and key established using module's key establishment methods.
N/A	Control Input	Includes API commands invoking cryptographic services and modes/key sizes/etc. used with cryptographic services.
N/A	Status Output	Includes status information regarding the module and status information regarding the invoked service/operation.

Table 9: Ports and Interfaces

The module does not implement a control output interface. Data output via the data output interface is inhibited when the module is performing pre-operational and conditional tests, zeroization, or when the module is in the error state.

4. Roles, Services, and Authentication

4.1 Authentication Methods

The module does not support authentication mechanisms; operators implicitly assume an authorized role (or set of roles) based on the service selected.

4.2 Roles

The table below lists the supported roles.

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	N/A
User	Role	User	N/A

Table 10: Roles

Roles are implicitly assumed by the calling application based on the service selected. The module does not support multiple concurrent operators. The calling application that loaded the module is its only operator.

4.3 Approved Services

Descriptions of the services available are provided in the table below. 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.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show Status	Returns FIPS mode status.	N/A	Show Status	API call parameters	None	Crypto Officer
Perform self-tests on-demand	Performs pre-operational self-tests.	Global FIPS indicator: FIPS_mode() returns non-zero value.	Re-instantiate module; API call parameters	Status	None	Crypto Officer

Zeroize	Zeroizes and de-allocates memory containing sensitive data.	N/A	Restart calling application; reboot or power-cycle host platform	None	None	Crypto Officer - AES CCM key: Z - AES CMAC key: Z - AES GCM key: Z - AES key: Z - AES XTS key: Z - DRBG 'C' value: Z - DRBG 'Key' value: Z - DRBG 'V' value: Z - DRBG entropy input: Z - DRBG seed: Z - DSA private key: Z - DSA public key: Z - ECDH private key: Z - ECDH public key: Z - ECDH shared secret: Z - ECDSA private key: Z - ECDSA public key: Z - HMAC key: Z - RSA private key: Z - RSA public key: Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show versioning information	Returns the name of the module and versioning information.	N/A	API call parameters	Module name, version	None	Crypto Officer
Perform symmetric encryption	Encrypts plaintext using supplied key and algorithm specification (AES).	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameters, key, plaintext	Status, ciphertext	AES for Symmetric Encryption/Decryption AES-XTS for Symmetric Encryption/Decryption	User - AES key: W,E - AES XTS key: W,E
Perform symmetric decryption	Decrypts ciphertext using supplied key and algorithm specification (AES).	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameters, key, ciphertext	Status, plaintext	AES for Symmetric Encryption/Decryption AES-XTS for Symmetric Encryption/Decryption	User - AES key: W,E - AES XTS key: W,E
Generate message digest	Computes and returns a message digest.	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameters, message	Status, hash	SHA for Message Digest	User
Perform authenticated symmetric encryption	Encrypts plaintext using supplied AES GCM key and IV.	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameters, key, plaintext	Status, ciphertext	AES-GCM for Authenticated Symmetric Encryption/Decryption AES-CCM for Authenticated Symmetric Encryption/Decryption	User - AES CCM key: W,E - AES GCM key: W,E
Perform authenticated symmetric decryption	Decrypts ciphertext using supplied AES GCM key and IV.	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameters, key, ciphertext	Status, plaintext	AES-GCM for Authenticated Symmetric Encryption/Decryption AES-CCM for Authenticated Symmetric Encryption/Decryption	User - AES CCM key: W,E - AES GCM key: W,E
Generate random number	Returns the specified number of random bits to the calling application.	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameters	Status, random bits	DRBG	User - DRBG entropy input: W,E - DRBG seed: G,E - DRBG 'C' value: G,E - DRBG 'V' value: G,E - DRBG 'Key' value: G,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Generate keyed hash (HMAC)	Computes and returns a message authentication code.	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameters, key, message	Status, hash	HMAC for Keyed Hash	User - HMAC key: W,E
Generate symmetric digest (CMAC)	Computes and returns a cipher message authentication code.	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameters, key, message	Status, hash	AES-CMAC for Symmetric Digest	User - AES CMAC key: W,E
Generate asymmetric key pair	Generates and returns the specified type of asymmetric key pair (RSA, DSA, ECDSA).	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameters	Status, key pair	DSA for Key Generation ECDSA for Key Generation RSA for Key Generation	User - DSA private key: G,R - DSA public key: G,R - ECDH private key: G,R - ECDH public key: G,R - ECDSA private key: G,R - ECDSA public key: G,R - RSA private key: G,R - RSA public key: G,R
Verify public key	Verifies the ECDSA public key.	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameters	Status, key pair	ECDSA for Key Verification	User - ECDH private key: W,E - ECDH public key: W,E
Calculate key agreement primitive	Calculates ECDH key agreement primitive.	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameter	Status, key components	ECDH Shared Secret Computation	User - ECDH public key: W,E - ECDH private key: R,W,E - ECDH shared secret: G,R

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Perform RSA encryption	Performs encryption with RSA.	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameters, public key, plaintext	Status, ciphertext	RSA for Asymmetric Encryption/Decryption	User - RSA public key: W,E
Perform RSA decryption	Performs decryption with RSA.	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameters, private key, ciphertext	Status, plaintext	RSA for Asymmetric Encryption/Decryption	User - RSA private key: W,E
Generate domain parameters	Generates DSA domain parameters.	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameter	Status, domain parameters	DSA for Parameter Generation	User
Verify domain parameters	Verifies DSA domain parameters.	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameter, domain parameters	Status	DSA for Parameter Verification	User
Generate signature	Generates a signature for the supplied message using the specified key and algorithm (DSA, ECDSA, RSA).	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameters, key, message	Status, signature	DSA for Signature Generation ECDSA for Signature Generation RSA for Signature Generation	User - DSA private key: W,E - ECDSA private key: W,E - RSA private key: W,E
Verify signature	Verifies the signature on the supplied message using the specified key and algorithm (DSA, ECDSA, RSA).	Global FIPS indicator: FIPS_mode() returns non-zero value.	API call parameters, key, signature, message	Status	DSA for Signature Verification ECDSA for Signature Verification RSA for Signature Verification	User - DSA public key: W,E - ECDSA public key: W,E - RSA public key: W,E

Table 11: Approved Services

The module does not offer any non-Approved services. Thus, as allowed per section 2.4.C 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 Approved mode indicator) and an implicit indication (via the API return indicating the successful completion of the service).

4.4 Non-Approved Services

The module does not offer any non-Approved services.

4.5 External Software/Firmware Loaded

The module does not support the capability to load software from external sources.

5. Software/Firmware Security

5.1 Integrity Techniques

The module's integrity is verified via a pre-operational self-test that uses an Approved integrity technique implemented within the cryptographic module itself. The entirety of the software cryptographic module is verified using a single embedded HMAC SHA-1 digest value. The module computes an HMAC SHA-1 digest at runtime and compares it to the embedded digest value; failure of the integrity test will cause the module to enter a critical error state.

The module's integrity test is performed automatically at module instantiation (i.e., when the module is loaded into memory for execution) without action from the module operator.

5.2 Initiate on Demand

The CO can initiate the pre-operational tests on demand by re-instantiating the module or issuing the `FIPS_selftest()` API command.

6. Operational Environment

6.1 Operational Environment Type and Requirements

The CorSSL™ FIPS Object 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 platform'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 environments provide 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 environments. Processes that are spawned by the module are owned by the module and are not owned by external processes/operators.

Please refer to section 2.2.4 of this document for a list/description of the tested operational environments.

7. Physical Security

This section is not applicable. Per section 7.7.1 of *ISO/IEC 19790:2012*, the requirements of this section are "applicable to hardware and firmware modules, and hardware and firmware components of hybrid modules".

8. Non-Invasive Security

This section is not applicable. There are currently no approved non-invasive mitigation techniques references in Annex F of *ISO/IEC 19790:2012*.

9. Sensitive Security Parameters Management

9.1 Storage Areas

There are no mechanisms within the module's cryptographic boundary for the persistent storage of SSPs. SSPs are stored in volatile RAM during module operation. The table below lists SSP storage areas for this module.

Storage Area Name	Description	Persistence Type
RAM	SSPs stored in RAM.	Dynamic

Table 12: Storage Areas

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.

Section 9.4 selects from the storage areas listed and specifies the appropriate storage area in the "Storage" column if applicable to a specific SSP.

9.2 SSP Input-Output Methods

The table below lists SSP input and output methods for this module.

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
API call parameter input	External	RAM	Plaintext	Manual	Electronic	
API call parameter output	RAM	External	Plaintext	Manual	Electronic	
ECDH key agreement input	External	RAM	Plaintext	Automated	Electronic	ECDH Shared Secret Computation
ECDH key agreement output	RAM	External	Plaintext	Automated	Electronic	ECDH Shared Secret Computation

Table 13: SSP Input-Output Methods

Section 9.4 selects from the input and output methods listed and specifies the appropriate method in the "Inputs/Outputs" column if applicable to a specific SSP.

9.3 SSP Zeroization Methods

The table below lists SSP zeroization methods for this module.

Zeroization Method	Description	Rationale	Operator Initiation
API call	API call zeroizes SSPs.	The operator executes the API call, which zeroizes the SSPs, yielding the SSPs irretrievable.	The operator calls zeroization API.

Zeroization Method	Description	Rationale	Operator Initiation
Remove power	Removing power from the module zeroizes SSPs.	The removal of power zeroizes SSPs, yielding the SSPs irretrievable.	The operator removes power from the module.
Unload module	Unloading the module zeroizes SSPs.	The unloading of the module zeroizes SSPs, making SSPs irretrievable.	The operator unloads the module.

Table 14: SSP Zeroization Methods

Section 9.4 selects from the zeroization methods listed and specifies the appropriate method in the "Zeroization" column if applicable to a specific SSP.

9.4 SSPs

The module supports the keys and other SSPs listed in the tables below. Note that all SSP imports and exports are electronic and performed within the TOEPP.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES key	Used for symmetric encryption/decryption.	Between 128 and 256 bits - Between 128 and 256 bits	Symmetric Key - CSP			AES for Symmetric Encryption/Decryption
AES CCM key	Used for authenticated symmetric encryption/decryption.	Between 128 and 256 bits - Between 128 and 256 bits	Symmetric Key - CSP			AES-CCM for Authenticated Symmetric Encryption/Decryption
AES GCM key	Used for authenticated symmetric encryption/decryption.	Between 128 and 256 bits - Between 128 and 256 bits	Symmetric Key - CSP			AES-GCM for Authenticated Symmetric Encryption/Decryption
AES XTS key	Used for symmetric encryption/decryption.	256 or 512 bits - 128 or 256 bits	Symmetric Key - CSP			AES-XTS for Symmetric Encryption/Decryption
AES CMAC key	Used for generating and verifying message authentication.	Between 128 and 256 bits - Between 128 and 256 bits	MAC - CSP			AES-CMAC for Symmetric Digest
HMAC key	Used for keyed hash messages.	Between 32 and 512 - 112 bits (minimum)	MAC - CSP			HMAC for Keyed Hash
DSA private key	Used for digital signature generation.	Between 224 and 256 bits - Between 112 and 128 bits	Private - CSP	DSA for Key Generation		DSA for Signature Generation
DSA public key	Used for digital signature verification.	Between 1024 and 3072 bits - Between 80 and 128 bits	Public - PSP	DSA for Key Generation		DSA for Signature Verification
ECDSA private key	Used for digital signature generation and asymmetric decryption.	[P-curves] Between 224 and 521 bits [B/K-curves] Between 223 and 571 bits - [P-curves] Between 112 and 256 bits [B/K-curves] Between 112 and 256 bits	Private - CSP	ECDSA for Key Generation		ECDSA for Signature Generation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
ECDSA public key	Used for digital signature verification and asymmetric encryption.	[P-curves] Between 192 and 521 bits [B/K-curves] Between 163 and 571 bits - [P-curves] Between 80 and 256 bits [B/K-curves] Between 80 and 256 bits	Public - PSP	ECDSA for Key Generation		ECDSA for Signature Verification
RSA private key	Used for digital signature generation and asymmetric decryption.	[signature generation] Between 2048 and 4096 bits [RSA decryption] Between 2048 and 8192 - [signature generation] Between 112 and 150 bits [RSA decryption] Between 112 and 201 bits	Private - CSP	RSA for Key Generation		RSA for Signature Generation RSA for Asymmetric Encryption/Decryption
RSA public key	Used for digital signature verification and asymmetric encryption.	[signature verification] Between 1024 and 4096 bits [RSA encryption] Between 2048 and 8192 - [signature verification] Between 80 and 150 bits [RSA encryption] Between 112 and 201 bits	Public - PSP	RSA for Key Generation		RSA for Signature Verification RSA for Asymmetric Encryption/Decryption
ECDH private key	Used for the computation of the ECDH shared secret.	[P-curves] Between 224 and 521 bits [B/K-curves] Between 223 and 571 bits - [P-curves] Between 112 and 256 bits [B/K-curves] Between 112 and 256 bits	Private - CSP	ECDSA for Key Generation		ECDH Shared Secret Computation
ECDH public key	Used for the computation of the ECDH shared secret.	[P-curves] Between 224 and 521 bits [B/K-curves] Between 223 and 571 bits - [P-curves] Between 112 and 256 bits [B/K-curves] Between 112 and 256 bits	Public - PSP	ECDSA for Key Generation		ECDH Shared Secret Computation
ECDH shared secret	Computed from the ECDH private key and ECDH public key.	[P-curves] Between 224 and 521 bits [B/K-curves] Between 223 and 571 bits - [P-curves] Between 112 and 256 bits [B/K-curves] Between 112 and 256 bits	Shared Secret - CSP		ECDH Shared Secret Computation	

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DRBG entropy input	Used in random bit generation (Counter, Hash, HMAC).	[Counter DRBG] Between 128 and 512 bits [Hash DRBG] Between 128 and 320 bits [HMAC DRBG] Between 160 and 1024 bits - [Counter DRBG] Between 128 and 512 bits [Hash DRBG] Between 128 and 320 bits [HMAC DRBG] Between 160 and 1024 bits	Entropy Input - CSP			DRBG
DRBG seed	Used in random bit generation.	[Counter DRBG] Between 256 and 384 bits [Hash DRBG] 440 or 888 bits [HMAC DRBG] 440 or 888 bits - [Counter DRBG] Between 256 and 384 bits [Hash DRBG] 440 or 888 bits [HMAC DRBG] 440 or 888 bits	Seed - CSP	DRBG		DRBG
DRBG 'C' value	Used in random bit generation (Hash).	440 or 888 bits - 440 or 888 bits	State Value - CSP	DRBG		DRBG
DRBG 'V' value	Used in random bit generation (Counter, Hash, HMAC).	[Counter DRBG] 128 bits [Hash DRBG] 440 or 888 bits [HMAC DRBG] 160, 256, or 512 bits - [Counter DRBG] 128 bits [Hash DRBG] 440 or 888 bits [HMAC DRBG] 160, 256, or 512 bits	State Value - CSP	DRBG		DRBG
DRBG 'Key' value	Used in random bit generation (Counter, HMAC).	[Counter DRBG] Between 128 and 256 bits [HMAC DRBG] 160, 256, or 512 bits - [Counter DRBG] Between 128 and 256 bits [HMAC DRBG] 160, 256, or 512 bits	State Value - CSP	DRBG		DRBG

Table 15: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES key	API call parameter input	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES CCM key	API call parameter input	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	
AES GCM key	API call parameter input	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	
AES XTS key	API call parameter input	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	
AES CMAC key	API call parameter input	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	
HMAC key	API call parameter input	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	
DSA private key	API call parameter input API call parameter output	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	DSA public key:Paired With
DSA public key	API call parameter input API call parameter output	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	DSA private key:Paired With
ECDSA private key	API call parameter input API call parameter output	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	ECDSA public key:Paired With
ECDSA public key	API call parameter input API call parameter output	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	ECDSA private key:Paired With
RSA private key	API call parameter input API call parameter output	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	RSA public key:Paired With
RSA public key	API call parameter input API call parameter output	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	RSA private key:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
ECDH private key	API call parameter input API call parameter output	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	ECDH public key:Paired With
ECDH public key	API call parameter input API call parameter output	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	ECDH private key:Paired With
ECDH shared secret	ECDH key agreement input ECDH key agreement output	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	ECDH private key:Derived From ECDH public key:Derived From
DRBG entropy input	API call parameter input	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	
DRBG seed	API call parameter input	RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	DRBG entropy input:Derived From
DRBG 'C' value		RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	
DRBG 'V' value		RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	
DRBG 'Key' value		RAM:Plaintext	Stored in RAM until zeroization API call is invoked, power is removed, or the module is unloaded.	API call Remove power Unload module	

Table 16: SSP Table 2

9.5 Transitions

The following list specifies applicable transition periods or timeframes where an algorithm or key length transitions from Approved to non-Approved:

- SHA-1: Per *NIST SP 800-131 Rev. 3*, SHA-1 usage will transition as follows:
    - The use of SHA-1 for applying cryptographic protection for non-digital signature applications is deprecated through December 31, 2030, and disallowed thereafter.

- The use of SHA-1 for processing already-protected information is acceptable through December 31, 2030, and allowed for legacy thereafter.

- SHA-1 will be disallowed for all uses starting January 1, 2031.

- ECDSA: There is currently no scheduled transition away from elliptic curves over binary fields (i.e., K-233, B233, K-283, B-283, K-409, B-409, K-571, B-571). However, these curves are now deprecated, and it is strongly recommended to use the prime curves defined *NIST SP 800 186-5* (i.e., P-224, P-256, P-384, P521) for the generation of ECDSA signatures. Despite their deprecation status, these curves are still considered Approved.

9.6 Additional Information

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.

10. Self-Tests

The module performs pre-operational self-tests and conditional self-tests. 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 pre-operational self-tests listed in the following table.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA-1 (A3356)	SHA-1	Software Integrity	SW/FW Integrity	Returns 1 upon success. Returns 0 upon error.	HMAC-SHA-1 software integrity test on fipscanister.o.

Table 17: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

The module performs the conditional self-tests listed in the following table.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-ECB (A3356)	128-bit	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Encrypt/Decrypt	After successful completion of the software integrity test.
AES-CCM (A3356)	192-bit	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Encrypt/Decrypt	After successful completion of the software integrity test.
AES-GCM (A3356)	256-bit	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Encrypt/Decrypt	After successful completion of the software integrity test.
AES-XTS Testing Revision 2.0 (A3356)	128/256-bit	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Encrypt/Decrypt	After successful completion of the software integrity test.
AES-CMAC (A3356)	128/192/256-bit	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Generate/Verify	After successful completion of the software integrity test.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
Counter DRBG (A3356)	AES-128/192/256-CTR; with/without derivation function	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Generate/Instantiate/Reseed	After successful completion of the software integrity test.
Hash DRBG (A3356)	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Generate/Instantiate/Reseed	After successful completion of the software integrity test.
HMAC DRBG (A3356)	HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Generate/Instantiate/Reseed	After successful completion of the software integrity test.
SHA-1 (A3356)	SHA-1	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Hash	Before the pre-operational software integrity test.
HMAC-SHA-1 (A3356)	SHA-1	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Hashed Message	Before the pre-operational software integrity test.
HMAC-SHA2-224 (A3356)	SHA2-224	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Hashed Message	Before the pre-operational software integrity test.
HMAC-SHA2-256 (A3356)	SHA2-256	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Hashed Message	Before the pre-operational software integrity test.
HMAC-SHA2-384 (A3356)	SHA2-384	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Hashed Message	Before the pre-operational software integrity test.
HMAC-SHA2-512 (A3356)	SHA2-512	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Hashed Message	Before the pre-operational software integrity test.
DSA SigGen (FIPS186-4) (A3356)	2048-bit; SHA2-256	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Sign	After successful completion of the software integrity test.
DSA SigVer (FIPS186-4) (A3356)	2048-bit; SHA2-256	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Verify	After successful completion of the software integrity test.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
ECDSA SigGen (FIPS186-4) (A3356)	P-224; K-223; SHA2-256	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Sign	After successful completion of the software integrity test.
ECDSA SigVer (FIPS186-4) (A3356)	P-224; K-223; SHA2-256	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Verify	After successful completion of the software integrity test.
RSA SigGen (FIPS186-4) (A3356)	2048-bit	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Sign	After successful completion of the software integrity test.
RSA SigVer (FIPS186-4) (A3356)	2048-bit	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Verify	After successful completion of the software integrity test.
KTS-IFC (A3356)	2048-bit	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Encrypt/Decrypt	After successful completion of the software integrity test.
KAS-ECC-SSC Sp800-56Ar3 (A3356)	P-224	KAT	CAST	Returns 1 upon success. Returns 0 upon error.	Shared secret computation	After successful completion of the software integrity test.
DSA KeyGen (FIPS186-4) (A3356)	-	PCT	PCT	Returns 1 upon success. Returns 0 upon error.	Sign/Verify	When the requested service requires the generation of an DSA key pair.
ECDSA KeyGen (FIPS186-4) (A3356)	-	PCT	PCT	Returns 1 upon success. Returns 0 upon error.	Sign/Verify	When the requested service requires the generation of an ECDSA key pair.
RSA KeyGen (FIPS186-4) (A3356)	-	PCT	PCT	Returns 1 upon success. Returns 0 upon error.	Sign/Verify/Encrypt/Decrypt	When the requested service requires the generation of an RSA key pair.
Other (ECDH)	-	PCT	PCT	Returns 1 upon success. Returns 0 upon error.	Key generation	When the requested service requires the generation of an ECDH key pair.
AES-XTS Duplicate Key Test	-	Duplicate Key Test	Critical Function	Returns 1 upon success. Returns 0 upon error.	Duplicate Key Test	When the requested service requires the generation of an AES-XTS key.

Table 18: Conditional Self-Tests

10.3 Periodic Self-Test Information

The table below specifies the module's periodic self-test information.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA-1 (A3356)	Software Integrity	SW/FW Integrity	On Demand	Manually

Table 19: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-ECB (A3356)	KAT	CAST	On Demand	Manually
AES-CCM (A3356)	KAT	CAST	On Demand	Manually
AES-GCM (A3356)	KAT	CAST	On Demand	Manually
AES-XTS Testing Revision 2.0 (A3356)	KAT	CAST	On Demand	Manually
AES-CMAC (A3356)	KAT	CAST	On Demand	Manually
Counter DRBG (A3356)	KAT	CAST	On Demand	Manually
Hash DRBG (A3356)	KAT	CAST	On Demand	Manually
HMAC DRBG (A3356)	KAT	CAST	On Demand	Manually
SHA-1 (A3356)	KAT	CAST	On Demand	Manually
HMAC-SHA-1 (A3356)	KAT	CAST	On Demand	Manually
HMAC-SHA2-224 (A3356)	KAT	CAST	On Demand	Manually
HMAC-SHA2-256 (A3356)	KAT	CAST	On Demand	Manually
HMAC-SHA2-384 (A3356)	KAT	CAST	On Demand	Manually
HMAC-SHA2-512 (A3356)	KAT	CAST	On Demand	Manually
DSA SigGen (FIPS186-4) (A3356)	KAT	CAST	On Demand	Manually
DSA SigVer (FIPS186-4) (A3356)	KAT	CAST	On Demand	Manually
ECDSA SigGen (FIPS186-4) (A3356)	KAT	CAST	On Demand	Manually
ECDSA SigVer (FIPS186-4) (A3356)	KAT	CAST	On Demand	Manually
RSA SigGen (FIPS186-4) (A3356)	KAT	CAST	On Demand	Manually
RSA SigVer (FIPS186-4) (A3356)	KAT	CAST	On Demand	Manually
KTS-IFC (A3356)	KAT	CAST	On Demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A3356)	KAT	CAST	On Demand	Manually
DSA KeyGen (FIPS186-4) (A3356)	PCT	PCT	N/A	N/A
ECDSA KeyGen (FIPS186-4) (A3356)	PCT	PCT	N/A	N/A
RSA KeyGen (FIPS186-4) (A3356)	PCT	PCT	N/A	N/A
Other (ECDH)	PCT	PCT	N/A	N/A
AES-XTS Duplicate Key Test	Duplicate Key Test	Critical Function	N/A	N/A

Table 20: Conditional Periodic Information

10.4 Error States

The table below specifies the module's error state information.

Name	Description	Conditions	Recovery Method	Indicator
Critical Error	When the module is in the Critical Error state, subsequent requests for cryptographic services will return an error result consistent with the API's function declaration for every invocation.	Upon failing any of the module's pre-operational or conditional self-tests.	To recover, the module must be re-instantiated, or the host device must be rebooted/power cycled. If these recovery methods do not result in the successful completion of all pre-operational self-tests and conditional CASTs, then the module will not be able to resume normal operations, and the CO should contact Corsec Security, Inc. for assistance.	Subsequent requests return FIPS_selftest_fail

Table 21: Error States

10.5 Operator Initiation of Self-Tests

The CO can initiate the module's pre-operational self-tests and conditional CASTs on-demand for periodic testing of the module manually by power-cycling or rebooting the module.

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
- 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 Installation, Initialization, and Startup Procedures

The CorSSL™ FIPS Object Module is not a standalone application; it is a cryptographic toolkit intended for use with third-party vendor solutions. action is required from developers or end-users to initialize the module for operation.

The module is not a standalone application. Rather, it is a cryptographic toolkit designed to support third-party vendor applications, and these applications are the sole consumers of the cryptographic services provided by the module. The module will be linked to a host application, and the host application will be pre-installed onto a target platform by the vendor or installed onto target platforms by the end-user.

The module is designed with a default entry point (DEP) that ensures that the pre-operational tests and conditional CASTS are initiated automatically when the module is loaded, without any specific action from the calling application or the end-user. The DEP invokes self-test code by calling the `FIPS_mode_set()` API command with a non-zero parameter. If successful, this action sets an internal FIPS mode flag to 'TRUE', placing the module in its Approved mode. End-users have no means to short-circuit or bypass these actions.

The module itself requires no configuration steps to be performed by application developers or end-users, and no end-user action is required to initialize the module for operation; the calling application performs any actions required to initialize the module. Failure of any of the initialization actions will result in a failure of the module to load for execution.

11.2 Administrator Guidance

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

The following list provides additional guidance for the CO:

- 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, rebooting/power-cycling the host device, or issuing the `FIPS_selftest()` API command.

- The `FIPS_mode()` API command can be used to determine the module's current mode of operation. This command will return a non-zero value when the module has properly initialized in its Approved mode of operation.

- The `FIPS_module_version_text()` API command 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.

11.3 Non-Administrator Guidance

The following list provides additional policies for module operators acting in a non-administrative role:

- 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 calling application is responsible for using entropy sources that meet the minimum security strength of 112 bits required for the Approved DRBGs as shown in *NIST SP 800-90A Rev. 1,* Table 2 and Table 3.

- In the event that the module encounters a DRBG self-test failure, the calling application must uninstantiate and reinstantiate the DRBG per the requirements found in *NIST SP 800-90A Rev. 1*.

11.4 Design and Rules

By design, the module follows or enforces the following rules of operation:

- The module provides two distinct operator roles: User and Cryptographic Officer.
- An operator does not have access to any cryptographic services prior to assuming an authorized role.
- The module performs all self-tests without any operator action required.
- The module inhibits data output during key generation, self-tests, zeroization, and error states.
- Status information output by the module does not contain CSPs or sensitive data that if misused could lead to a compromise of the module.
- The module does not support a maintenance interface or role.
- The module does not support manual SSP establishment methods.
- The module does not have any proprietary external input/output devices used for entry/output of data.
- The module does not output intermediate key values.
- The module does not provide bypass services or ports/interfaces.

11.5 End of Life

As the module does not persistently store SSPs, secure sanitization of the module consists of performing the zeroization methods described in section 9.3. This will ensure that any SSPs in volatile memory are zeroized.

12. Mitigation of Other Attacks

This section is not applicable. The module does not claim to mitigate any attacks beyond the level 1 requirements for this validation.

Appendix A. Acronyms and Abbreviations

Table 22 provides definitions for the acronyms and abbreviations used in this document.

Table 22. Acronyms and Abbreviations

Term	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCCS	Canadian Centre for Cyber Security
CCM	Counter with CBC MAC
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
DoDIN APL	Department of Defense Information Network Approved Product List
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard
FOM	FIPS Object Module
GCM	Galois/Counter Mode
GMAC	Galois Message Authentication Code
GPC	General-Purpose Computer
HMAC	(keyed-) Hash Message Authentication Code
IEC	International Electrotechnical Commission
IG	Implementation Guidance
ISO	International Organization for Standardization

Term	Definition
KAS	Key Agreement Scheme
KAT	Known Answer Test
KTS	Key Transport Scheme
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
OS	Operating System
PAA	Processor Algorithm Acceleration
PCT	Pairwise Consistency Test
PKCS	Public Key Cryptography Standard
PSP	Public Security Parameter
PSS	Probabilistic Signature Scheme
RAM	Random Access Memory
RNG	Random Number Generator
RSA	Rivest, Shamir, and Adleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SP	Special Publication
SSC	Shared Secret Computation
SSP	Sensitive Security Parameter
TOEPP	Tested Operational Environment's Physical Perimeter
XTS	XOR-Encrypt-XOR-based Tweakable Block Cipher 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
Web: www.corsec.com