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: 0.2

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

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 its 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.

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.

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

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[1] APL[2].

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") version 2.0.16.001 is a software library providing a C language API[3] 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 SSP establishment functions to secure data-at-rest/data-in-flight and to support industry-standard secure communications protocols.

The CorSSL FOM is built upon the OpenSSL FIPS Object Module 2.0.16 code base, providing engineering teams with a completely compatible cryptographic engine, allowing quick "drop-in" replacement into any existing OpenSSL FIPS Object Module solutions. The CorSSL FOM does not modify the OpenSSL interface, maintaining complete compatibility, and eliminating engineering development time to meet FIPS 140-3 requirements.

The CorSSL FOM is validated at the FIPS 140-3 section levels shown in Table 1.

Table 1 – Security Levels

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

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

ISO/IEC 24759 Section 6. [Number Below]	FIPS 140-3 Section Title	Security Level
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.

2. Cryptographic Module Specification

The CorSSL™ FIPS Object Module 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 environments listed in Table 2.

Table 2 – Tested Operational Environments

#	Operating System	Hardware Platform	Processor	PAA/Acceleration
1	Debian 9	Dell PowerEdge R440	Intel® Xeon Silver 4214R	With
2	Debian 9	Dell PowerEdge R440	Intel® Xeon Silver 4214R	Without

The module is designed to utilize the AES-NI[4] extended instruction set when available by the host platform's CPU for processor algorithm acceleration (PAA) of its AES implementation.

There are no vendor-affirmed operational environments claimed.

The cryptographic module maintains validation compliance when operating on any general-purpose computer (GPC) provided that the GPC uses any operating system/mode specified on the validation certificate, or another compatible 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.

2.2 Algorithm Implementations

The module implements the Approved algorithms listed in Table 3.

[4] AES-NI – Advanced Encryption Algorithm New Instructions

Table 3 – Approved Algorithms

CAVP Certificate[5]	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strengths	Use / Function
A3356	**AES** *FIPS PUB[6] 197* *NIST SP 800-38A*	CBC[7], CFB1[8], CFB8, CFB128, CTR[9], ECB[10], OFB[11]	128, 192, 256	Encryption/decryption
A3356	**AES** *NIST SP 800-38B*	CMAC[12]	128, 192, 256	MAC generation/verification
A3356	**AES** *NIST SP 800-38C*	CCM[13]	128, 192, 256	Encryption/decryption
A3356	**AES** *NIST SP 80- 38D*	GCM[14] (internal IV)	128, 192, 256	Encryption/decryption
A3356	**AES** *NIST SP 800-38E*	XTS[15,16,17]	128, 256	Encryption/decryption *This algorithm must only be used for storage applications.*
Vendor Affirmed	**CKG**[18] *NIST SP 800-133rev2*	-	-	Cryptographic key generation
A3356	**DRBG**[19] *NIST SP 800-90Arev1*	Counter-based	128, 192, 256-bit AES-CTR	Deterministic random bit generation
		Hash-based	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	Deterministic random bit generation
		HMAC-based	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	Deterministic random bit generation
A3356	**DSA**[20] *FIPS PUB 186-4*	-	2048/224, 2048/256, 3072/256 (SHA2-224, SHA2-256, SHA2-384, SHA2-512)	Domain parameter generation
		-	1024/160, 2048/224, 2048/256, 3072/256 (SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512)	Domain parameter verification
		-	2048/224, 2048/256, 3072/256	Key pair 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] CFB – Cipher Feedback
[9] CTR – Counter
[10] ECB – Electronic Code Book
[11] OFB – Output Feedback
[12] CMAC – Cipher-Based Message Authentication Code
[13] CCM – Counter with Cipher Block Chaining - Message Authentication Code
[14] GCM – Galois Counter Mode
[15] XOR – Exclusive OR
[16] XEX – XOR Encrypt XOR
[17] XTS – XEX-Based Tweaked-Codebook Mode with Ciphertext Stealing
[18] CKG – Cryptographic Key Generation
[19] DRBG – Deterministic Random Bit Generator
[20] DSA – Digital Signature Algorithm

CAVP Certificate[5]	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strengths	Use / Function
		-	2048/224, 2048/256, 3072/256 (SHA2-224, SHA2-256, SHA2-384, SHA2-512)	Digital signature generation
		-	1024/160, 2048/224, 2048/256, 3072/256 (SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512)	Digital signature verification
A3356	**ECDSA**[21] *FIPS PUB 186-4*	Testing candidates	B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521	Key pair generation
		-	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	Public key validation
		-	B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 (SHA2-224, SHA2-256, SHA2-384, SHA2-512)	Digital signature generation
		-	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 (SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512)	Digital signature verification
A3356	**HMAC** *FIPS PUB 198-1*	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	112 (minimum)	Message authentication
A3356	**KAS-ECC-SSC**[22] *NIST SP 800-56Arev3*	ephemeralUnified	B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521	Shared secret computation *Key agreement method complies with FIPS 140-3 Implementation Guidance D.F.*
A3356	**KTS**[23] *NIST SP 800-38C*	AES-CCM	128, 192, 256	Key wrap/unwrap (authenticated encryption)[24] *SSP establishment methodology provides between 128 and 256 bits of encryption strength*
A3356	**KTS** *NIST SP 800-38D*	AES-GCM	128, 192, 256	Key wrap/unwrap (authenticated encryption)[25] *SSP establishment methodology provides between 128 and 256 bits of encryption strength*

[21] ECDSA – Elliptic Curve Digital Signature Algorithm
[22] KAS-ECC-SSC – Key Agreement Scheme - Elliptic Curve Cryptography - Shared Secret Computation
[23] KTS – Key Transport Scheme
[24] Per *FIPS 140-3 Implementation Guidance* D.G, AES-CCM is an Approved key transport technique.
[25] Per *FIPS 140-3 Implementation Guidance* D.G, AES-GCM is an Approved key transport technique.

CAVP Certificate[5]	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strengths	Use / Function
A3356	**KTS** *FIPS PUB 197* *NIST SP 800-38B*	AES with CMAC	128, 192, 256	Key wrap/unwrap (encryption with message authentication)[26] *SSP establishment methodology provides between 128 and 256 bits of encryption strength*
A3356	**KTS** *FIPS PUB 197* *FIPS PUB 198-1*	AES with HMAC	128, 192, 256	Key wrap/unwrap (encryption with message authentication)[27] *SSP establishment methodology provides between 128 and 256 bits of encryption strength*
A3356	**KTS-IFC**[28] *NIST SP 800-56Brev2*	KTS-OAEP-basic	rsakpg1-basic 2048, 3072, 4096, 6144, 8192 (SHA2-224, SHA2,256, SHA2-384, SHA2-512)	Key encapsulation/un-encapsulation *SSP establishment methodology provides between 112 and 201 bits of encryption strength*
A3356	**RSA**[29] *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
			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[30]	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
A3356	**SHS**[31] *FIPS PUB 180-4*	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	-	Message digest

The vendor affirms the following cryptographic security methods:

[26] Per *FIPS 140-3 Implementation Guidance* D.G, AES with CMAC is an Approved key transport technique.
[27] Per *FIPS 140-3 Implementation Guidance* D.G, AES with HMAC is an Approved key transport technique.
[28] RSA – Rivest Shamir Adleman
[29] RSA – Rivest Shamir Adleman
[30] PSS – Probabilistic Signature Scheme
[31] SHS – Secure Hash Standard

- 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 4 below.

Table 4 – Non-Approved Algorithms Allowed in the Approved Mode of Operation

Algorithm	Caveat	Use / Function
AES (Cert. A3356)	SSP establishment methodology provides between 128 and 256 bits of encryption strength	Key unwrapping (using any Approved mode)[32]

The module does not implement any non-Approved algorithms allowed in the Approved mode of operation with no security 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. The physical perimeter of the cryptographic module is defined by each host platform on which the module is installed. Figure 1 below illustrates a block diagram of a typical GPC and the module's physical perimeter.

[32] Per *FIPS 140-3 Implementation Guidance* D.G, AES in any Approved mode is an Approved key transport technique (for unwrapping only).

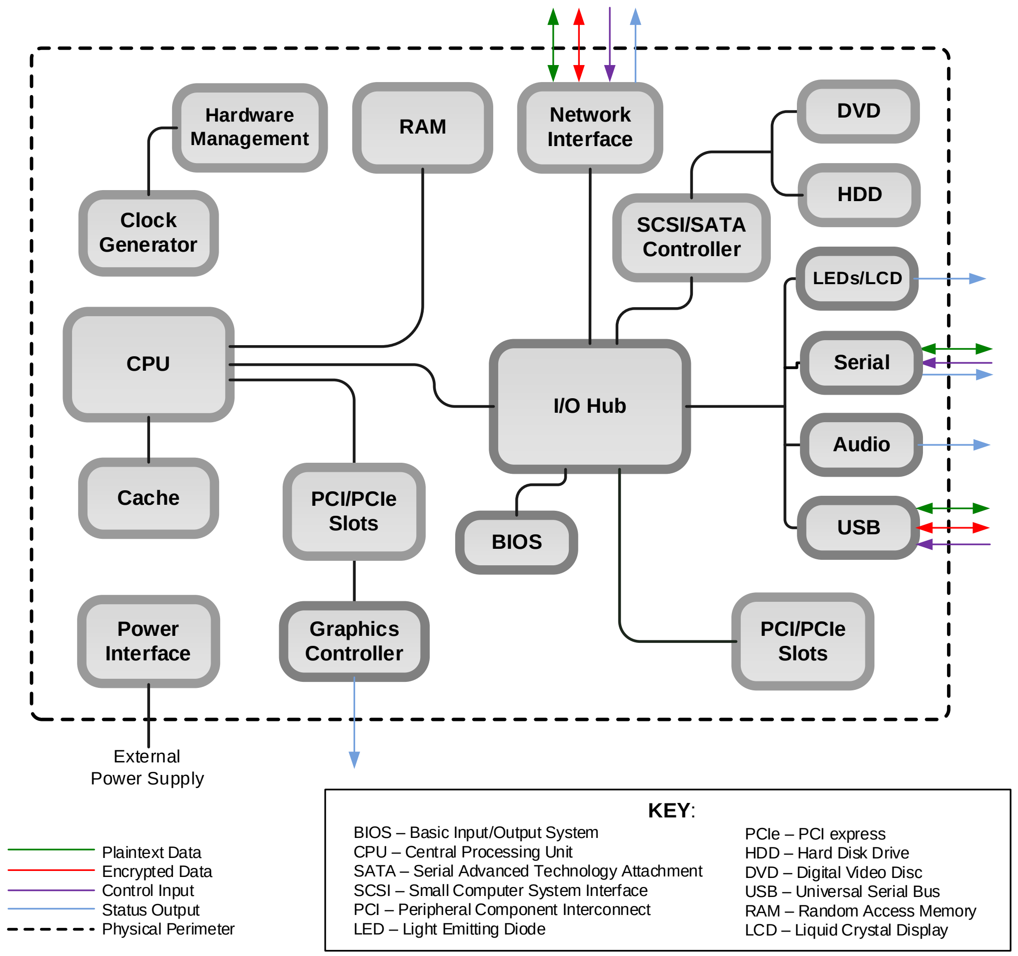

Figure 1 – GPC Block Diagram

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’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 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 image also includes an embedded HMAC SHA-1 MAC value for verifying the module’s integrity at runtime. The module is entirely contained within the physical perimeter.

Figure 2 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.

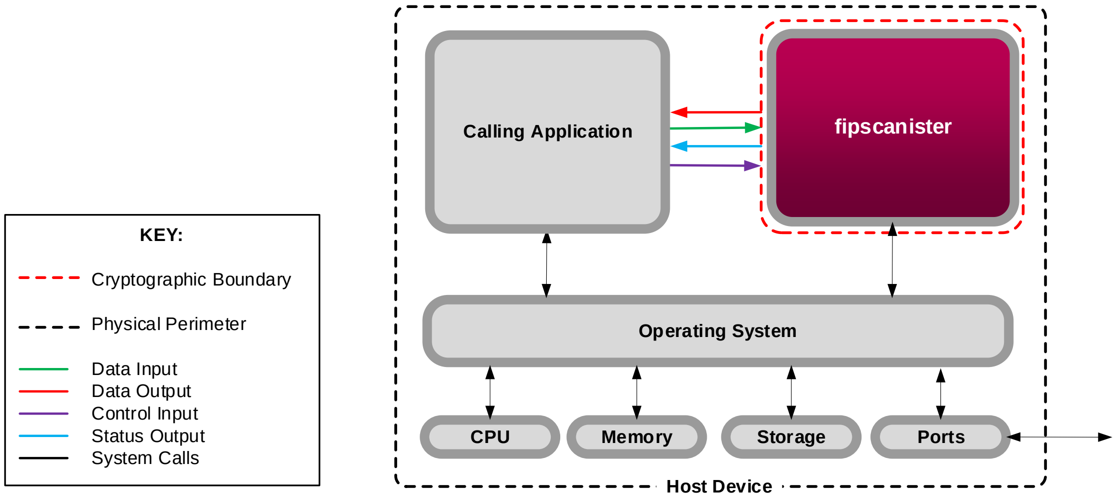

Figure 2 – Module Block Diagram (with Cryptographic Boundary)

2.4 Modes of Operation

Once all pre-operational self-tests have completed successfully, the module supports only an Approved mode of operation. Table 3, Table 4, and above list the algorithms available in the Approved mode; Table 7 provides descriptions of the Approved services.

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 logical interfaces can be found in Table 5. Note that the module does not output control information, and thus has no specified control output interface.

Table 5 – Ports and Interfaces

Physical Port	Logical Interface	Data That Passes Over Port/Interface
Physical data input port(s) of the host device	Data Input • API input arguments that provide input data for processing	• Data to be encrypted, decrypted, signed, verified, or hashed • Keys to be used in cryptographic services • Random seed material for the module's DRBG • Keying material to be used as input to SSP establishment services
Physical data output port(s) of the host device	Data Output • API output arguments that return generated or processed data back to the caller	• Data that has been encrypted, decrypted, or verified • Digital signatures • Hashes • Random values generated by the module's DRBG • Keys established using module's SSP establishment methods
Physical control input port(s) of the host device	Control Input • API input arguments that are used to initialize and control the operation of the module	• API commands invoking cryptographic services • Modes, key sizes, etc. used with cryptographic services
Physical status output port(s) of the host device	Status Output • API call return values	• Status information regarding the module • 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 (CO) 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 6 below lists the supported roles, along with the services (including input and output) available to each role.

Table 6 – Roles, Service Commands, Input and Output

Role	Service	Input	Output
CO	Show Status	API call parameters	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 message digest	API call parameters, message	Status, hash
User	Perform authenticated symmetric encryption	API call parameters, key, plaintext	Status, ciphertext
User	Perform authenticated symmetric decryption	API call parameters, key, ciphertext	Status, plaintext
User	Generate random number	API call parameters	Status, random bits
User	Generate keyed hash (HMAC)	API call parameters, key, message	Status, hash
User	Generate symmetric digest (CMAC)	API call parameters, key, message	Status, hash
User	Generate asymmetric key pair	API call parameters	Status, key pair

Role	Service	Input	Output
User	Calculate key agreement primitive	API call parameter	Status, key pairs
User	Perform key wrapping	API call parameter, encryption key, key	Status, encrypted key
User	Perform key unwrapping	API call parameters, decryption key, encrypted key	Status, decrypted key
User	Perform key encapsulation	API call parameter, encryption key, key	Status, encrypted key
User	Perform key un-encapsulation	API call parameters, decryption key, encrypted key	Status, decrypted key
User	Generate signature	API call parameters, key, message	Status, signature
User	Verify signature	API call parameters, key, signature, message	Status

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 services available are provided in Table 7 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.

Table 7 – Approved Services

Service	Description	Approved Security Function(s)	Keys and/or SSPs	Roles	Access Rights to Keys and/or SSPs	Indicator
Show Status	Return Approved mode indicator	None	None	CO	N/A	N/A
Perform self-tests on-demand	Perform pre-operational self-tests	HMAC (Cert. A3356) SHA-1 (Cert. A3356)	HMAC key	CO	HMAC key – E	API return value
Zeroize	Zeroizes and de-allocates memory containing sensitive data	None	All keys/SSPs	CO	All keys/SSPs – Z	N/A
Show versioning information	Return module's name and versioning information	None	None	CO	N/A	N/A

Service	Description	Approved Security Function(s)	Keys and/or SSPs	Roles	Access Rights to Keys and/or SSPs	Indicator
Perform symmetric encryption	Encrypt plaintext using supplied key and algorithm specification (AES)	AES (Cert. A3356) AES-XTS (Cert. A3356)	AES key AES-XTS key	User	AES key – WE AES-XTS key – WE	API return value
Perform symmetric decryption	Decrypt ciphertext using supplied key and algorithm specification (AES)	AES (Cert. A3356)	AES key	User	AES key – WE	API return value
Generate message digest	Compute and return a message digest	SHS (Cert. A3356)	None	User	None	API return value
Perform authenticated symmetric encryption	Encrypt plaintext using supplied AES GCM key and IV	AES-CCM (Cert. A3356) AES-GCM (Cert. A3356)	AES-CCM key AES-GCM key AES-GCM IV	User	AES-CCM key – WE AES-GCM key – WE AES-GCM IV – WE	API return value
Perform authenticated symmetric decryption	Decrypt ciphertext using supplied AES GCM key and IV	AES-CCM (Cert. A3356) AES-GCM (Cert. A3356)	AES-CCM key AES-GCM key AES-GCM IV	User	AES-CCM key – WE AES-GCM key – WE AES-GCM IV – WE	API return value
Generate random number	Returns the specified number of random bits to the calling application	Counter DRBG (Cert. A3356) Hash DRBG (Cert. A3356) HMAC DRBG (Cert. A3356)	DRBG entropy DRBG seed DRBG 'C' Value DRBG 'V' Value DRBG 'Key' Value	User	DRBG entropy – RE DRBG seed – GE DRBG 'C' Value – GE DRBG 'V' Value – GE DRBG 'Key' Value – GE	API return value
Generate keyed hash (HMAC)	Compute and return a message authentication code	HMAC (Cert. A3356)	HMAC key	User	HMAC key – WE	API return value
Generate symmetric digest (CMAC)	Compute and return a cipher message authentication code	AES-CMAC (Cert. A3356)	AES-CMAC key	User	AES-CMAC key – WE	API return value
Generate asymmetric key pair	Generate and return the specified type of asymmetric key pair (RSA, DSA, ECDSA)	Counter DRBG (Cert. A3356) Hash DRBG (Cert. A3356) HMAC DRBG (Cert. A3356) RSA (Cert. A3356) DSA (Cert. A3356) ECDSA (Cert. A3356)	DSA private key DSA public key ECDSA private key ECDSA public key RSA private key RSA public key	User	DSA private key – GR DSA public key – GR ECDSA private key – GR ECDSA public key – GR RSA private key – GR RSA public key – GR	API return value
Calculate key agreement primitive	Calculate ECDH key agreement primitive	ECDH (Cert. A3356)	ECDH public key ECDH private key	User	ECDH public key – WE ECDH private key – WE	API return value
Perform key wrapping	Perform key wrap with AES	AES (Cert. A3356)	AES key	User	AES key – WE	API return value
Perform key unwrapping	Perform key unwrap with AES	AES (Cert. A3356)	AES key	User	AES key – WE	API return value
Perform key encapsulation	Perform key encapsulation with RSA	RSA (Cert. A3356)	RSA public key	User	RSA public key – WE	API return value
Perform key un-encapsulation	Perform key un-encapsulation with RSA	RSA (Cert. A3356)	RSA private key	User	RSA private key – WE	API return value
Generate signature	Generate a signature for the supplied message using the specified key and algorithm (RSA, DSA, ECDSA)	RSA (Cert. A3356) DSA (Cert. A3356) ECDSA (Cert. A3356)	RSA private key DSA private key ECDSA private key	User	RSA private key – WE DSA private key – WE ECDSA private key – WE	API return value
Verify signature	Verify the signature on the supplied message using the specified key and algorithm (RSA, DSA, ECDSA)	RSA (Cert. A3356) DSA (Cert. A3356) ECDSA (Cert. A3356)	RSA public key DSA public key ECDSA public key	User	RSA public key – WE DSA public key – WE ECDSA public key – WE	API return value

The module does not provide 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).

5. Software/Firmware Security

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. This integrity test can also be performed on demand by the module operator by issuing the `FIPS_selftest()` API command.

The CorSSL™ FIPS Object Module is not a standalone application; it is a cryptographic toolkit intended for use in a with a vendor's solution. 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 itself requires no configuration steps to be performed by application developers or end-users, and no action is required from developers or end-users to initialize the module for operation. 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.

6. Operational Environment

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 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 implemented completely in software, such that the physical security is provided solely by the host device. Therefore, per *ISO/IEC 19790:2012* 7.7.1, this section is not applicable.

8. Non-Invasive Security

This section is not applicable. There are currently no approved non-invasive mitigation techniques references 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 8 below.

Table 8 – SSPs

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & Related Keys
Keys								
AES key (CSP)	Between 128 and 256 bits	AES (CBC, CCM, CFB, CTR, ECB, OFB, KW, KWP modes) (Cert. A3356) KTS (Cert. A3356)	-	IMPORT: Imported via API call parameter EXPORT: Exported in plaintext	-	Not persistently stored by the module	Unload module; API call; Remove power	Symmetric encryption/decryption; key wrap/unwrap
AES GCM key (CSP)	Between 128 and 256 bits	AES (GCM mode) (Cert. A3356)	-	IMPORT: Imported via API call parameter EXPORT: Exported in plaintext	-	Not persistently stored by the module	Unload module; API call; Remove power	Authenticated symmetric encryption/decryption; key wrap/unwrap
AES XTS key (CSP)	128 or 256 bits	AES (XTS mode) (Cert. A3356)	-	IMPORT: Imported via API call parameter EXPORT: Exported in plaintext	-	Not persistently stored by the module	Unload module; API call; Remove power	Symmetric encryption/decryption
AES CMAC key (CSP)	Between 128 and 256 bits	AES (CMAC mode) (Cert. A3356)	-	IMPORT: Imported via API call parameter EXPORT: Exported in plaintext	-	Not persistently stored by the module	Unload module; API call; Remove power	MAC generation/verification encryption/decryption; key wrap/unwrap
HMAC key (CSP)	112 bits (minimum)	HMAC (Cert. A3356)	-	IMPORT: Imported in plaintext via API parameter EXPORT: Exported in plaintext via API parameter	-	Not persistently stored by the module	Unload module; API call; Remove power	Message authentication code
RSA private key (CSP)	Between 112 and 150 bits	RSA (Cert. A3356)	Generated internally via Approved DRBG	IMPORT: Imported in plaintext via API parameter EXPORT: Exported in plaintext via API parameter	-	Not persistently stored by the module	Unload module; API call; Remove power	Digital signature generation; asymmetric decryption
RSA public key (PSP)	Between 80 and 150 bits	RSA (Cert. A3356)	Generated internally via Approved DRBG	IMPORT: Imported in plaintext via API parameter EXPORT: Exported in plaintext via API parameter	-	Not persistently stored by the module	Unload module; API call; Remove power	Digital signature verification; asymmetric encryption

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & Related Keys
DSA private key (CSP)	128 or 256 bits	DSA (Cert. A3356)	Generated internally via Approved DRBG	IMPORT: Imported in plaintext via API parameter EXPORT: Exported in plaintext via API parameter	-	Not persistently stored by the module	Unload module; API call; Remove power	Digital signature generation
DSA public key (PSP)	80 or 256 bits	DSA (Cert. A3356)	Generated internally via Approved DRBG	IMPORT: Imported in plaintext via API parameter EXPORT: Exported in plaintext via API parameter	-	Not persistently stored by the module	Unload module; API call; Remove power	Digital signature verification
ECDSA private key (CSP)	Between 80 and 256 bits	ECDSA (Cert. A3356)	Generated internally via Approved DRBG	IMPORT: Imported in plaintext via API parameter EXPORT: Exported in plaintext via API parameter	-	Not persistently stored by the module	Unload module; API call; Remove power	Digital signature generation
ECDSA public key (PSP)	Between 80 and 256 bits	ECDSA (Cert. A3356)	Generated internally via Approved DRBG	IMPORT: Imported in plaintext via API parameter EXPORT: Exported in plaintext via API parameter	-	Not persistently stored by the module	Unload module; API call; Remove power	Digital signature verification
ECDH private key (CSP)	Between 128 and 256 bits	KAS-SSC (Cert. A3356)	Generated internally via Approved DRBG	IMPORT: Imported in plaintext via API parameter EXPORT: Exported in plaintext via API parameter	-	Not persistently stored by the module	Unload module; API call; Remove power	Computation of ECDH shared secret
ECDH public key (PSP)	Between 128 and 256 bits	KAS-SSC (Cert. A3356)	Generated internally via Approved DRBG	IMPORT: Imported in plaintext via API parameter EXPORT: Exported in plaintext via API parameter	-	Not persistently stored by the module	Unload module; API call; Remove power	Computation of ECDH shared secret
Other SSPs								
AES GCM IV (PSP)	-	AES (GCM mode) (Cert. A3356)	-	-	Constructed at its entirety internally deterministically	Not persistently stored by the module	Unload module; API call; Remove power	Initialization vector for AES GCM
DRBG Entropy Input (CSP)	-	DRBG (Cert. A3356)	-	IMPORT: Imported in plaintext via API parameter EXPORT: Never exported	-	Plaintext in volatile memory	Unload module; API call; Remove power	Used in random bit generation (Counter, Hash, HMAC)
DRBG Seed (CSP)	-	DRBG (Cert. A3356)	Generated internally	-	-	Not persistently stored by the module	Unload module; API call; Remove power	Used in random bit generation
DRBG 'C' Value (CSP)	-	DRBG (Cert. A3356)	Generated internally	-	-	Plaintext in volatile memory	Unload module; API call; Remove power	Used in random bit generation (Hash)

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & Related Keys
DRBG 'V' Value (CSP)	-	DRBG (Cert. A3356)	Generated internally	-	-	Plaintext in volatile memory	Unload module; API call; Remove power	Used in random bit generation (Counter, Hash, HMAC)
DRBG 'Key' Value (CSP)	-	DRBG (Cert. A3356)	Generated internally	-	-	Plaintext in volatile memory	Unload module; API call; Remove power	Used in random bit generation (Counter, HMAC)

9.2 DRBGs

The module implements the following Approved DRBG(s):

- Counter-based DRBG
- Hash-based DRBG (non-compliant)
- HMAC-based DRBG (non-compliant)

These DRBGs are used to generate random values at the request of the calling application. Outputs from the DRBGs are also used in the generation of seeds for asymmetric key pair generation.

The module does not implement any non-Approved DRBGs.

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.

9.4 SSP Zeroization Methods

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 temporarily stored SSPs, zeroization of sensitive data is performed automatically by API function calls. The module will first overwrite existing values with "0"s and then free the memory. Additionally, module operators can unload the module from memory or reboot/power-cycle the host device.

9.5 RBG 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 (using an HMAC SHA-1 digest verification)

10.2 Conditional Self-Tests

Conditional self-tests are performed by the module during module operation when certain conditions exist.

The module performs the following conditional self-tests:

- Conditional cryptographic algorithm self-tests (CASTs)
    - AES ECB encrypt KAT (128-bit)
    - AES ECB decrypt KAT (128-bit)
    - AES CCM encrypt KAT (192-bit)
    - AES CCM decrypt KAT (192-bit)
    - AES GCM encrypt KAT (256-bit)
    - AES GCM decrypt KAT (256-bit)
    - AES XTS encrypt KAT (128/256-bit)
    - AES XTS decrypt KAT (128/256-bit)
    - AES CMAC generate KAT (CBC mode; 128/192/256-bit)
    - AES CMAC verify KAT (CBC mode; 128/192/256-bit)
    - CTR_DRBG generate/instantiate/reseed KAT (256-bit AES)
    - Hash_DRBG generate/instantiate/reseed KAT (SHA2-256)
    - HMAC_DRBG generate/instantiate/reseed KAT (SHA2-256)
    - HMAC KATs (SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512)
    - 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 sign KAT (2048-bit; SHA2-256; PKCS#1 v1.5 scheme)
    - RSA verify KAT (2048-bit; SHA2-256; PKCS#1 v1.5 scheme)
    - RSA encrypt KAT
    - RSA decrypt KAT
    - ECC CDH shared secret computation KAT (P-224 curve)

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
  - RSA encrypt/decrypt PCT
  - ECDH key generation PCT

- Conditional critical function tests
  - XTS AES duplicate key test

10.3 Self-Test Failure Handling

Upon failure of any of the module's pre-operational or conditional self-tests, the module will enter a critical error state. An internal global error flag `FIPS_selftest_fail` will be set and subsequently tested to prevent the execution of any cryptographic services while the error state persists. For any subsequent request for cryptographic services, the module will return an error result consistent with the API's function declaration for every invocation.

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.

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 third-party vendors as a package containing the binary and HMAC digest file that the Crypto Officer is to install onto a target platform specified in section 2.1 above or one where portability is maintained.

11.2 Initialization

This module is designed to support third-party 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 by the calling application, 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 Approved mode flag to 'TRUE', placing the module in its Approved mode. 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. 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.

11.5 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 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.

- The module performs assurances for its RSA-based key transport scheme as specified in the following sections of *NIST SP 800-56Brev2:*

  - 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 800-56Brev2*. 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.

- To meet the AES GCM (key/IV) pair uniqueness requirements from *NIST SP 800-38D*, the module complies with *FIPS 140-3 IG* C.H as follows:

    - Per scenario 3 of *FIPS 140-3 IG* C.H, the module employs deterministic IV construction as specified in section 8.2.1 of *NIST SP 800-38D*. The IV is constructed at its entirety internally deterministically, with a length of 96 bits.

  In the event that power to the module is lost and subsequently restored, the calling application must ensure that any AES-GCM keys used for encryption or decryption are re-distributed.

- 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 800-90Arev1,* 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 800-90Arev1*.

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

Table 9 – Acronyms and Abbreviations

Term	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
CBC	Cipher Block Chaining
CCCS	Canadian Centre for Cyber Security
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
ECB	Electronic Code Book
ECC CDH	Elliptic Curve Cryptography Cofactor Diffie-Hellman
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EMI/EMC	Electromagnetic Interference /Electromagnetic Compatibility
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
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
NIST	National Institute of Standards and Technology

Term	Definition
OS	Operating System
PCT	Pairwise Consistency Test
PKCS	Public Key Cryptography Standard
PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, and Adleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SP	Special Publication

Prepared by:
Corsec Security, Inc.

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

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