Wind River Systems, Inc.

Wind River FIPS Object Module

Version: 2.0.16.004

FIPS 140-3 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 0.3

Prepared for:

WNDRVR

Wind River Systems, Inc.
500 Wind River Way
Alameda, CA 94501
United States of America

Phone: +1 510 748 4100
www.windriver.com

Prepared by:

Corsec

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

List of Tables

List of Figures

1. General

1.1 Overview

This is a non-proprietary Cryptographic Module Security Policy for the Wind River FIPS Object Module (version 2.0.16.004) from Wind River Systems, Inc. (Wind River). This Security Policy describes how the Wind River 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 FIPS 140-3 validation of the module. The Wind River FIPS Object Module is referred to in this document as “Wind River FOM” or “module”.

1.2 Security Levels

The Wind River FIPS Object Module is validated at the FIPS 140-3 security 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.

1.3 Additional Information

More information is available from the following sources:

- The Wind River website www.windriver.com contains information on the full line of services and solutions from Wind River.

- 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.

2. Cryptographic Module Specification

2.1 Description

The Wind River FIPS Object Module is a general-purpose software cryptographic library offering symmetric encryption/decryption, digital signature generation/verification, hashing, cryptographic key generation, random number generation, message authentication, and key establishment functions. It is designed for ease of use with the popular OpenSSL cryptographic library and toolkit and is available for use as part of Wind River's platforms.

2.1.1 Purpose and Use

The module is intended for use by U.S. Federal agencies or other markets that require FIPS 140-3 validated cryptography. As a multi-chip standalone library, the module is intended to be used in embedded systems.

2.1.2 Module Type

The Wind River FIPS Object Module is a Software module.

2.1.3 Module Embodiment

The Wind River FIPS Object Module has a Multi-Chip Standalone embodiment.

2.1.4 Module Characteristics

The module does not have any additional characteristics. .

2.1.5 Cryptographic Boundary

The cryptographic boundary is the contiguous perimeter that surrounds all memory-mapped functionality provided by module image when loaded in the host platform's memory for execution. The module image also includes an embedded HMAC SHA-1 MAC value for verifying the module's integrity at runtime.

2.1.6 Tested Operational Environment's Physical Perimeter (TOEPP)

As a software cryptographic module, the module has no physical components. The physical perimeter of the tested operational environment is defined by the enclosure of each host platform on which the module is installed.

Figure 1 below shows the logical block diagram of the module executing in memory, its interactions with surrounding software components, and its physical perimeter. The module is entirely contained within the TOEPP.

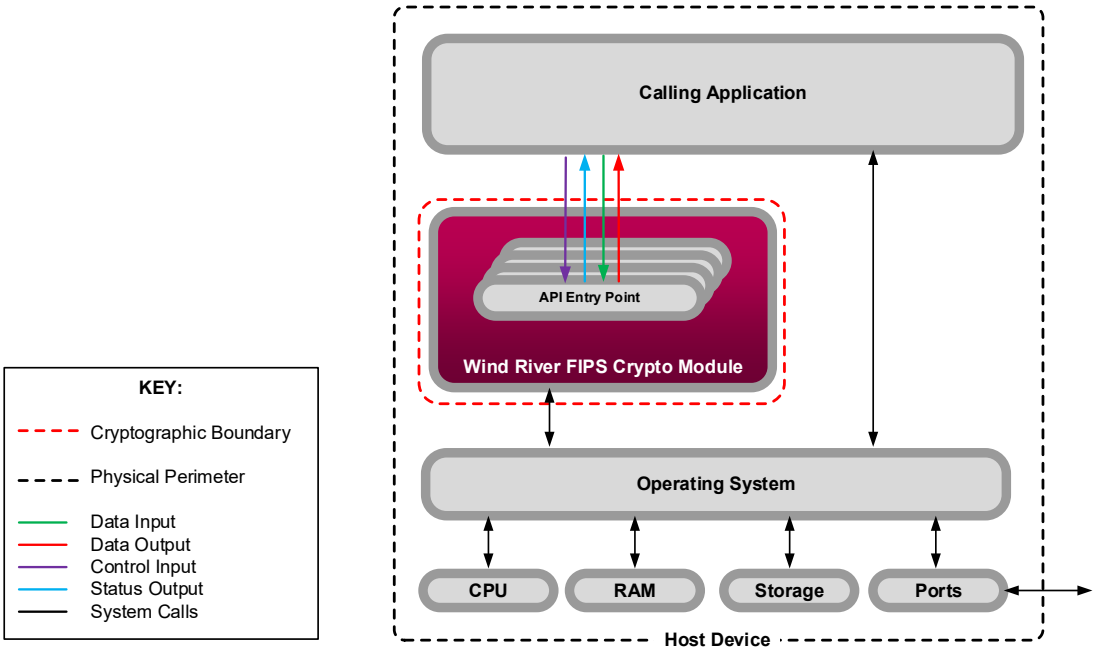

Figure 1: Module Block Diagram (with Cryptographic Boundary)

2.2 Tested and Vendor Affirmed Module Version and Identification

2.2.1 Tested Module Identification – Hardware

The module is not a hardware cryptographic module.

N/A for this module.

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

The Wind River FOM is a software module with the executable code sets shown in the table below.

Package or File Name	Software/ Firmware Version	Features	Integrity Test
fipscanister.o	2.0.16.004		Yes

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

2.2.3 Tested Module Identification – Hybrid Disjoint Hardware

The module is not a hybrid cryptographic module.

N/A for this module.

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)
Wind River VxWorks 7.0 SR640	BAE Systems RAD5545 SpaceVPX single-board computer	BAE Systems RAD5500	No		2.0.16.004
Wind River VxWorks 6.9.4.12	Gaisler GR-CPCI-GR740 single-board computer	Gaisler LEON4FT SPARC V8	No		2.0.16.004

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

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

There are no vendor-affirmed operational environments claimed.

Operating System	Hardware Platform
N/A	N/A

Table 4: Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid

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 table below lists the modes of operation for the module.

Mode Name	Description	Type	Status Indicator
ApprovedMode	Upon successful completion of all pre-operational self-tests, the module supports the use of Approved/allowed algorithms only.	Approved	A non-zero return from a call to the FIPS_module_mode() API command

Table 5: Modes List and Description

2.5 Algorithms

2.5.1 Approved Algorithms

The module employs cryptographic algorithm implementations from the following source:

- Wind River FIPS Object Module (libcrypto) (Cert. A6780)

The module implements the Approved algorithms listed in the table below.

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A6780	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A6780	Key Length - 128, 192, 256	SP 800-38C
AES-CFB1	A6780	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB128	A6780	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A6780	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A6780	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B
AES-CTR	A6780	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A6780	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A6780	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-OFB	A6780	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-XTS Testing Revision 2.0	A6780	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E
Counter DRBG	A6780	Prediction Resistance - No, Yes Mode - AES-128, AES-192, AES-256 Derivation Function Enabled - Yes	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-5)	A6780	Curve - P-224, P-256, P-384, P-521 Secret Generation Mode - testing candidates	FIPS 186-5
ECDSA KeyVer (FIPS186-5)	A6780	Curve - P-224, P-256, P-384, P-521	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A6780	Curve - P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512 Component - No	FIPS 186-5
ECDSA SigVer (FIPS186-5)	A6780	Curve - P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-5
Hash DRBG	A6780	Prediction Resistance - No, Yes Mode - SHA-1, SHA2-256, SHA2-512	SP 800-90A Rev. 1
HMAC DRBG	A6780	Prediction Resistance - No, Yes Mode - SHA-1, SHA2-256, SHA2-512	SP 800-90A Rev. 1
HMAC-SHA-1	A6780	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A6780	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A6780	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A6780	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A6780	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A6780	Domain Parameter Generation Methods - P-224, P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-FFC-SSC Sp800-56Ar3	A6780	Domain Parameter Generation Methods - MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Scheme - dhEphem - KAS Role - initiator, responder	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
KDF IKEv2 (CVL)	A6780	Diffie-Hellman Shared Secret Length - Diffie-Hellman Shared Secret Length: 224-8192 Increment 8 Derived Keying Material Length - Derived Keying Material Length: 160-3072 Increment 8 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
KTS-IFC	A6780	Modulo - 2048, 3072, 4096, 6144, 8192 Key Generation Methods - rsakpg1-basic Scheme - KTS-OAEP-basic - KAS Role - initiator, responder Key Transport Method - Key Length - 768	SP 800-56B Rev. 2
RSA KeyGen (FIPS186-5)	A6780	Key Generation Mode - probable Modulo - 2048, 3072, 4096 Primality Tests - 2pow100 Private Key Format - standard	FIPS 186-5
RSA SigGen (FIPS186-5)	A6780	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
RSA SigVer (FIPS186-4)	A6780	Signature Type - PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-5)	A6780	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
Safe Primes Key Generation	A6780	Safe Prime Groups - MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3
Safe Primes Key Verification	A6780	Safe Prime Groups - MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3
SHA-1	A6780	Message Length - Message Length: 160, 0-65528 Increment 8	FIPS 180-4
SHA2-224	A6780	Message Length - Message Length: 224, 0-65528 Increment 8	FIPS 180-4
SHA2-256	A6780	Message Length - Message Length: 256, 0-65528 Increment 8	FIPS 180-4
SHA2-384	A6780	Message Length - Message Length: 384, 0-65528 Increment 8	FIPS 180-4
SHA2-512	A6780	Message Length - Message Length: 512, 0-65528 Increment 8	FIPS 180-4

Table 6: Approved Algorithms

2.5.2 Vendor Affirmed Algorithms

The module implements the vendor affirmed algorithms listed in the table below.

Name	Properties	Implementation	Reference
CKG	Key Type:Asymmetric	N/A	SP 800-133 Rev2 section 4 example 1 and IG D.H

Table 7: Vendor-Affirmed Algorithms

2.5.3 Non-Approved, Allowed Algorithms

The module does not implement non-Approved algorithms that are allowed in the Approved mode.

N/A for this module.

2.5.4 Non-Approved, Allowed Algorithms with No Security Claimed

The module implements the non-Approved algorithms (allowed in the Approved mode with no security claimed) listed in the table below.

N/A for this module.

2.5.5 Non-Approved, Not Allowed Algorithms

The module does not offer non-Approved algorithms that are not allowed in the Approved mode.

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	BC-UnAuth	AES symmetric encryption using the AES key	Publication:FIPS 197, SP 800-38A	AES-CBC: (A6780) AES-CFB1: (A6780) AES-CFB8: (A6780) AES-CFB128: (A6780) AES-CTR: (A6780) AES-ECB: (A6780) AES-OFB: (A6780)
AES for Symmetric Decryption	BC-UnAuth	AES symmetric Decryption using the AES key	Publication :FIPS 197, SP 800-38A	AES-CBC: (A6780) AES-CFB1: (A6780) AES-CFB8: (A6780) AES-CFB128: (A6780) AES-CTR: (A6780) AES-ECB: (A6780) AES-OFB: (A6780)
AES-XTS for Symmetric Encryption	BC-UnAuth	AES-XTS symmetric encryption	Publication:SP 800-38E	AES-XTS Testing Revision 2.0: (A6780)
AES-XTS for Symmetric Decryption	BC-UnAuth	AES-XTS symmetric decryption	Publication:FIPS 197, SP 800-38D	AES-XTS Testing Revision 2.0: (A6780)
AES-GCM for Authenticated Symmetric Encryption	BC-Auth	AES-GCM authenticated symmetric encryption	Publication:FIPS 197, SP 800-38D	AES-GCM: (A6780) Counter DRBG: (A6780)
AES-GCM for Authenticated Symmetric Decryption	BC-Auth	AES-GCM authenticated symmetric decryption	Publication:FIPS 197, SP 800-38D	AES-GCM: (A6780) Counter DRBG: (A6780)
AES-CCM for Authenticated Symmetric Encryption	BC-Auth	AES-CCM authenticated symmetric encryption	Publication:FIPS 197, SP 800-38C	AES-CCM: (A6780) AES-CBC: (A6780)
AES-CCM for Authenticated Symmetric Decryption	BC-Auth	AES-CCM authenticated symmetric decryption	Publication:FIPS 197, SP 800-38C	AES-CCM: (A6780) AES-CBC: (A6780)
CTR_DRBG	DRBG	Used for generating random bit strings using CTR_DRBG	Publication:SP 800-90A Rev1	Counter DRBG: (A6780)
Hash_DRBG	DRBG	Used for generating random bit strings using Hash_DRBG	Publication:SP 800-90A Rev1	Hash DRBG: (A6780)
HMAC_DRBG	DRBG	Used for generating random bit strings using HMAC_DRBG	Publication:SP 800-90A Rev1	HMAC DRBG: (A6780)

Name	Type	Description	Properties	Algorithms
HMAC for Message Authentication	MAC	HMAC message authentication using the HMAC key	Publication:FIPS 198-1	HMAC-SHA-1: (A6780) HMAC-SHA2-224: (A6780) HMAC-SHA2-256: (A6780) HMAC-SHA2-384: (A6780) HMAC-SHA2-512: (A6780)
HMAC for Module Integrity Test	MAC	HMAC message authentication for the module's pre-operational integrity test using the HMAC key	Publications :FIPS 198-1	HMAC-SHA-1: (A6780)
AES-CMAC for MAC Generation	MAC	AES-CMAC MAC generation using the AES-CMAC key	Publication:FIPS 197	AES-CMAC: (A6780)
AES-CMAC for MAC Verification	MAC	AES-CMAC MAC verification using the AES-CMAC key	Publication::FIPS 197, SP 800-38B	AES-CMAC: (A6780)
RSA for Key Generation	AsymKeyPair-KeyGen	RSA key generation of the RSA private key and RSA public key	Publication:FIPS 186-5, SP 800-90A Rev1	RSA KeyGen (FIPS186-5): (A6780) Counter DRBG: (A6780) CKG: ()
ECDSA for Key Generation	AsymKeyPair-KeyGen	ECDSA key generation of the ECDSA private key and ECDSA public key	Publication:FIPS 186-5, SP 800-90A Rev1	ECDSA KeyGen (FIPS186-5): (A6780) Counter DRBG: (A6780) CKG: ()
ECDSA for Signature Generation	DigSig-SigGen	Used for generating ECDSA signatures	Publication:FIPS 186-5	ECDSA SigGen (FIPS186-5): (A6780) SHA2-224: (A6780) SHA2-256: (A6780) SHA2-384: (A6780) SHA2-512: (A6780)
ECDSA for Signature Verification	DigSig-SigVer	Used for verifying ECDSA signatures	Publication :FIPS 186-5	ECDSA SigVer (FIPS186-5): (A6780) SHA2-224: (A6780) SHA2-256: (A6780) SHA2-384: (A6780) HMAC-SHA2-512: (A6780)
KDF for IKEv2	KAS-135KDF	Key derivation function for IKEv2	Publication :SP 800-135 Rev1	KDF IKEv2: (A6780) HMAC-SHA-1: (A6780) HMAC-SHA2-224: (A6780) HMAC-SHA2-256: (A6780) HMAC-SHA2-384: (A6780) HMAC-SHA2-512: (A6780) SHA-1: (A6780) SHA2-224: (A6780) SHA2-256: (A6780) SHA2-384: (A6780) SHA2-512: (A6780)

Name	Type	Description	Properties	Algorithms
DH Key Generation	KAS-KeyGen	Used for generating ECDH public/private key pairs	Publication:SP 800-56A Rev. 3, SP 800-90A Rev. 1, SP 800-90A Rev. 2	ECDSA KeyGen (FIPS186-5): (A6780) Counter DRBG: (A6780) CKG: () Key Type: Asymmetric
ECDH Key Generation	AsymKeyPair-KeyGen	Used for generating ECDH public/private key pairs	Publication :SP 800-56A Rev3, SP 800-90A Rev1	ECDSA KeyGen (FIPS186-5): (A6780) Counter DRBG: (A6780) CKG: ()
DH Shared Secret Computation	KAS-SSC	Shared secret computation for DH using the DH private key and DH public key	Publication:SP 800-56A Rev3 IG:D.F Scenario 2(1)	KAS-FFC-SSC Sp800-56Ar3: (A6780) Safe Primes Key Generation: (A6780) Safe Primes Key Verification: (A6780) SHA-1: (A6780) SHA2-224: (A6780) SHA2-256: (A6780) SHA2-384: (A6780) SHA2-512: (A6780) Counter DRBG: (A6780)
ECDH Shared Secret Computation	KAS-SSC	Shared secret computation for ECDH using the ECDH private key and ECDH public key	Publication:SP 800-56A Rev3 IG:D.F Scenario 2(1)	KAS-ECC-SSC Sp800-56Ar3: (A6780) ECDSA KeyGen (FIPS186-5): (A6780) Counter DRBG: (A6780) SHA-1: (A6780) SHA2-224: (A6780) SHA2-256: (A6780) SHA2-384: (A6780) SHA2-512: (A6780)
AES-CCM for Wrap/Unwrap	KTS-Wrap	AES-CCM key wrap and unwrap using the AES-CCM key	Publication:FIPS 197, SP 800-38C IG:D.G Key strength :Key establishment methodology provides between 128 and 256 bits of encryption strength	AES-CCM: (A6780) AES-CBC: (A6780) Counter DRBG: (A6780)
AES-GCM for Wrap/Unwrap	KTS-Wrap	AES-GCM key wrap and unwrap using the AES-GCM key	Publication:FIPS 197, SP 800-38D IG:D.G Key Strength:Key establishment methodology provides between 128 and 256 bits of encryption strength.	AES-GCM: (A6780) AES-CBC: (A6780) Counter DRBG: (A6780)

Name	Type	Description	Properties	Algorithms
AES+MAC for Key Wrap/Unwrap	KTS-Wrap	AES-CMAC key wrap and unwrap using the AES-CMAC key AES+HMAC key wrap and unwrap using the AES key and HMAC key	Publication :FIPS 197, SP 800-38A, SP 800-38B, FIPS 198-1 IG :D.G Key Strength :Key establishment methodology provides between 128 and 256 bits of encryption strength	AES-CMAC: (A6780) AES-CBC: (A6780) AES-CFB1: (A6780) AES-CFB8: (A6780) AES-CFB128: (A6780) AES-CTR: (A6780) AES-ECB: (A6780) AES-OFB: (A6780) HMAC-SHA-1: (A6780) HMAC-SHA2-224: (A6780) HMAC-SHA2-256: (A6780) HMAC-SHA2-384: (A6780) HMAC-SHA2-512: (A6780)
AES for Unauthenticated Key Unwrap (allowed) (legacy)	KTS-Wrap	AES key unwrap using the AES key (with any Approved unauthenticated mode)	Publication:FIPS 197, SP 800-38A IG:C.M, D.G Key Strength:Key establishment methodology provides between 128 and 256 bits of encryption strength.	AES-CBC: (A6780) Direction: Decrypt AES-CFB1: (A6780) AES-CFB8: (A6780) AES-CTR: (A6780) AES-CFB128: (A6780) AES-ECB: (A6780) AES-OFB: (A6780)
RSA for Key Transport	KTS-Encap	RSA key transport using the RSA public key (for key encapsulation) and RSA private key (for key un-encapsulation)	Publication :FIPS 186-5, SP 800-56B rev 2 Key Strength :Key establishment methodology provides between 112 and 201 bits of encryption strength.	KTS-IFC: (A6780) RSA KeyGen (FIPS186-5): (A6780)
RSA (FIPS 186-5) for Signature Generation	DigSig-SigGen	RSA digital signature generation using the RSA private key	Publication:FIPS 186-5, FIPS 180-4, SP 800-90A Rev1	RSA SigGen (FIPS186-5): (A6780) SHA2-224: (A6780) SHA2-256: (A6780) SHA2-384: (A6780) SHA2-512: (A6780)
RSA (FIPS 186-4) for Signature Verification (legacy)	DigSig-SigVer	RSA digital signature verification using the RSA public key (per FIPS 186-4 with SHA-1 and/or a 1024-bit modulo)	Publication:FIPS 186-4, FIPS 180-4 IG. :C.M	RSA SigVer (FIPS186-4): (A6780) SHA-1: (A6780) SHA2-224: (A6780) SHA2-256: (A6780) SHA2-384: (A6780) SHA2-512: (A6780)
RSA (FIPS 186-5) for Signature Verification	DigSig-SigVer	RSA digital signature verification using the RSA public key (per FIPS 186-5)	Publication:FIPS 186-5, FIPS 180-4	RSA SigVer (FIPS186-5): (A6780) SHA2-224: (A6780) SHA2-256: (A6780) SHA2-384: (A6780) SHA2-512: (A6780)

Name	Type	Description	Properties	Algorithms
SHA for Message Digest Generation	SHA	Used for generating hashes	Publication :FIPS 180-4	SHA2-224: (A6780) SHA2-256: (A6780) SHA2-384: (A6780) SHA2-512: (A6780) SHA-1: (A6780)
ECDSA (FIPS 186-5) for Key Verification	AsymKeyPair-KeyVer	ECDSA key verification of the ECDSA public key	Publication:FIPS 186-5	ECDSA KeyVer (FIPS186-5): (A6780)

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

The following algorithm-specific information is applicable to the module:

2.7.1 AES

The module supports the use of the following methods for unauthenticated key unwrap:

- AES-CBC
- AES-CFB1
- AES-CFG128
- AES-CTR
- AES-ECB
- AES-OFB

These methods are allowed for legacy use only and can only be used on data that was generated prior to 2018 (as specified in *FIPS 140-3 IG* C.M).

2.7.2 AES-GCM

To meet the AES GCM (key/IV) pair uniqueness requirements from *NIST SP 800-38D*, the module's AES-GCM implementation constructs the IV using the following methods:

- In compliance with scenario 1 of *FIPS 140-3 IG* C.H, the module generates the AES-GCM IV as specified in *RFC 5282*. In this case, the IV is only for use within the IPsec protocol.

  The module uses *RFC 7296* compliant IKEv2 to establish the shared secret SKEYSEED from which the AES-GCM encryption and authentication keys are derived. The counter portion of the IV is set by the module within its cryptographic boundary. When the IV exhausts the maximum number of possible values for a given session key, the first party, client or server, to encounter this condition will trigger a handshake to establish a new encryption key.

- In compliance with 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.

  The module uses 32 bits of the IV field as a name and 64 bits as a deterministic non-repetitive counter for a combined IV length of 96 bits. The name field includes an encoding of the module name, and the

name construction allows for at least 2^32 different names. The implementation of the deterministic non-repetitive counter management logic inside the module ensures that when the counter part of the IV exhausts the maximum number of possible values for a given session key using a 64-bit counter starting from 0 and increasing, the encryptor shall abort the session when the counter reaches the maximum value of 2^64 -1.

In the event that power to the module is lost and subsequently restored, the calling application must ensure that new AES-GCM keys are distributed.

2.7.3 AES-XTS

The 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.

The module implements a check to ensure that the two AES keys used in the XTS-AES algorithm are not identical.

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 800-133rev2* for an Approved use of AES-XTS.

2.7.4 RSA Signature Verification (FIPS 186-4)

As specified in *FIPS PUB 186-4*, the module supports verification of RSA signatures in the following ways:

- Using a modulus providing less that 112 bits of security strength
- Using SHA-1 as the underlying hash function

These methods are Approved for legacy use only and may be used only to process information that was protected prior to 2011 (as specified in *FIPS 140-3 IG* C.M).
The module also supports the verification of X9.31 RSA signatures as specified in *FIPS PUB 186-4*. This method is Approved for legacy use only and may be used only to process information that was protected prior to February 5, 2024 (as specified in *FIPS 140-3 IG* C.M).

2.7.5 SHA

The module supports the use of SHA-1 as an underlying hash for digital signature verification as specified in *FIPS PUB 180-4* and the *FIPS PUB 186* series. This use is Approved for legacy use only and may be used only to process information that was protected prior to 2011 (as specified in *FIPS 140-3 IG* C.M).

2.8 RBG and Entropy

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. This is the logical equivalent of a LOAD command.

The calling application and its entropy sources are located outside the module's cryptographic boundary but within its TOEPP. Thus, there is no assurance of the minimum strength of the generated SSPs. 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. This entropy is supplied by means of callback functions. Those functions must return an error if the minimum entropy strength cannot be met.

As allowed by Additional Comment #12 of *FIPS 140-3 IG* 9.3.A, this complies with an earlier version of Resolution 2(b), which can be found at the following URL:

https://csrc.nist.gov/CSRC/media/Projects/cryptographic-module-validation-program/documents/IG%209.3.A%20Resolution%202b%5BMarch%2026%202024%5D.pdf

N/A for this module.

N/A for this module.

2.9 Key Generation

The module supports the generation of cryptographic keys as follows:

- ECDSA and RSA key pairs (per section 5.1 of *NIST SP 800-133rev2)*
- DH and ECDH key pairs (per section 5.2 of *NIST SP 800-133rev2)*

As specified in section 4 of *NIST SP 800-133rev2*, the cryptographic module uses its Approved DRBG 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 provides the cryptographic primitives necessary to support key agreement schemes and key transport methods for use by the calling application to establish keys.

2.10.1 Key Agreement Schemes

The module implements the following Approved key agreement schemes (as specified in *FIPS 140-3 IG* D.F Scenario 2, path 1) which have been CAVP tested and validated:

- KAS-ECC-SSC
- KAS-FFC-SSC

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.

Key confirmation is not supported by the module.

These methods are not used to establish keys into the module.

2.10.2 Key Transport Methods

The module implements the Approved RSA-based key transport scheme (as specified in *FIPS 140-3 IG* D.G), which has been CAVP tested and validated. The module performs assurances as specified in the following section 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.

The module also implements the following Approved/allowed key transport methods (as specified in *FIPS 140-3 IG* D.G) which have been CAVP tested and validated:

- AES + MAC wrap/unwrap
- AES-CCM wrap/unwrap
- AES-GCM wrap/unwrap
- AES unwrap[1] (legacy)

These methods are not used to establish keys into the module.

2.11 Industry Protocols

The module implements Approved KDFs for the following industry protocols, which have been CAVP tested and validated:

- IKEv2

While the module provides cryptographic functionality that a calling application can leverage in support of these protocols, the module does not implement these protocols. No parts of the protocols other than the Approved cryptographic algorithms and the KDFs have been tested by the CAVP and CMVP.

[1] 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. Note that the module does not output control information and has no specified control output interface.

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	API input parameters - 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	API output parameters and return values - Includes data that has been encrypted/decrypted/verified, digital signatures, hashes, random values generated by the module's DRBG, and keys established using module's key establishment methods
N/A	Control Input	API method calls - Includes API commands invoking cryptographic services, modes/key sizes/etc. used with cryptographic services
N/A	Status Output	API output parameters and return/error codes - Includes status information regarding the module and status information regarding the invoked service/operation

Table 9: Ports and Interfaces

4. Roles, Services, and Authentication

4.1 Authentication Methods

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

N/A for this module.

4.2 Roles

The module provides two distinct operator roles: Cryptographic Officer (CO) and User.The table below lists the supported roles.

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None
User	Role	User	None

Table 10: Roles

Only one role may be active at a time and the module does not allow concurrent operators. As the module is a software library, 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
Compute shared secret	Compute DH/ECDH shared secret	Indicator API return value = 1	API call parameters	API call parameters	DH Shared Secret Computation ECDH Shared Secret Computation	User - DH public key: W,E,Z - DH private key: W,E,Z - ECDH public key: W,E,Z - ECDH private key: W,E,Z - Shared secret: G,R,Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Generate keyed hash	Compute a message authentication code	Indicator API return value = 1	API call parameters, key, message	Status, MAC	HMAC for Message Authentication	User - HMAC key: W,E,Z
Generate hash	Generate a message digest	Indicator API return value = 1	API call parameters, message	Status, hash	SHA for Message Digest Generation	User
Generate random number	Return random bits to the calling application	Indicator API return value = 1	API call parameters, DRBG type, entropy	Status, random number	CTR_DRBG Hash_DRBG HMAC_DRBG	User - DRBG entropy input: W,E,Z - DRBG 'C' value: G,E,Z - DRBG 'V' value: G,E,Z - DRBG 'Key" Value: G,E,Z - DRBG seed: W,E,Z
Generate symmetric digest	Generate symmetric digest	Indicator API return value = 1	API call parameters, key, plaintext	Status, digest	AES-CMAC for MAC Generation	User - AES-CMAC key: W,E,Z
Perform self-tests on-demand	Perform pre-operational self-tests	Indicator API return value = 1	Re-instantiate module; API call parameters	Status	HMAC for Module Integrity Test	
Perform symmetric encryption	Encrypt plaintext data	Indicator API return value = 1	API call parameters, key, plaintext	Status, ciphertext	AES for Symmetric Encryption AES for Symmetric Decryption AES-XTS for Symmetric Encryption AES-XTS for Symmetric Decryption	User - AES key: W,E,Z - AES-XTS key: W,E,Z
Perform symmetric decryption	Decrypt ciphertext data	Indicator API return value = 1	API call parameters, key, ciphertext	Status, plaintext	AES for Symmetric Encryption AES for Symmetric Decryption AES-XTS for Symmetric Encryption AES-XTS for Symmetric Decryption	User - AES key: W,E,Z - AES-XTS key: W,E,Z
Show versioning information	Return module versioning information	N/A	API call parameters	Module name, version	None	Crypto Officer
Show status	Return FIPS mode status	N/A	API call parameters	Current operational status	None	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Zeroize	Zeroize and de-allocate memory containing sensitive data	N/A	Remove power; reboot/power-cycle host device; API call parameters	None	None	Crypto Officer - AES key: Z - AES-CCM key: Z - AES-GCM key: Z - AES-XTS key: Z - AES-CMAC key: Z - HMAC key: Z - RSA private key: Z - RSA public key: Z - ECDSA private key: Z - ECDSA public key: Z - DH private key: Z - DH public key: Z - ECDH private key: Z - ECDH public key: Z - IKEv2 encryption key: Z - Shared secret: Z - DRBG entropy input: Z - DRBG seed: Z - DRBG 'C' value: Z - DRBG 'V' value: Z - DRBG 'Key" Value: Z - AES-GCM IV: Z - IKEv2 authentication key: Z
Verify symmetric digest	Verify symmetric digest	Indicator API return value = 1	API call parameters, key, ciphertext, digest	Status	AES-CMAC for MAC Verification	User - AES-CMAC key: W,E,Z
Perform authenticated symmetric encryption	Encrypt plaintext with authentication tag	Indicator API return value = 1	API call parameters, key, plaintext	Status, ciphertext, tag	AES-GCM for Authenticated Symmetric Encryption AES-CCM for Authenticated Symmetric Encryption	User - AES-CCM key: G,E,Z - AES-GCM key: G,E,Z - AES-GCM IV: G,E,Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Perform authenticated symmetric decryption	Decrypt ciphertext with authentication tag	Indicator API return value = 1	API call parameters, key, ciphertext, tag	Status, plaintext	AES-GCM for Authenticated Symmetric Decryption AES-CCM for Authenticated Symmetric Decryption	User - AES-CCM key: G,E,Z - AES-GCM key: G,E,Z - AES-GCM IV: G,E,Z
Generate asymmetric key pair	Generate a public/private key pair	Indicator API return value = 1	API call parameters	Status, key pair	RSA for Key Generation ECDSA for Key Generation DH Key Generation ECDH Key Generation	User - ECDSA public key: G,R,Z - ECDSA private key: G,R,Z - RSA public key: G,R,Z - RSA private key: G,R,Z - DH private key: G,R,Z - DH public key: G,R,Z - ECDH private key: G,R,Z - ECDH public key: G,R,Z
Verify ECDSA public key	Verify an ECDSA public key	Indicator API return value = 1	API call parameters, key	Status	ECDSA (FIPS 186-5) for Key Verification	User - ECDSA public key: W,E,Z
Generate digital signature	Generate a digital signature	Indicator API return value = 1	API call parameters, key, message	Status, signature	ECDSA for Signature Generation RSA (FIPS 186-5) for Signature Generation	User - ECDSA private key: W,E,Z - RSA private key: W,E,Z
Verify digital signature	Verify a digital signature	Indicator API return value = 1	API call parameters, key, signature, message	Status	ECDSA for Signature Verification RSA (FIPS 186-4) for Signature Verification (legacy) RSA (FIPS 186-5) for Signature Verification	User - ECDSA public key: W,E,Z - RSA public key: W,E,Z
Perform key wrap	Perform key wrap	Indicator API return value = 1	API call parameters, encryption key, key	Status, encrypted key	AES-CCM for Wrap/Unwrap AES-GCM for Wrap/Unwrap AES+MAC for Key Wrap/Unwrap	User - AES key: W,E,Z - AES-CCM key: W,E,Z - AES-CMAC key: W,E,Z - AES-GCM key: W,E,Z - HMAC key: W,E,Z - AES-GCM IV: E,Z
Perform key encapsulation	Perform key encapsulation	Indicator API return value = 1	API call parameter, encapsulating key, key	Status, encapsulated key	RSA for Key Transport	User - RSA private key: W,E,Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Perform key decapsulation	Perform key decapsulation	Indicator API return value = 1	API call parameter, decapsulating key, encapsulated key	Status, decapsulated key	RSA for Key Transport	User - RSA public key: W,E,Z
Perform IKEv2 key derivation	Derive keys for IKEv2 sessions	Indicator API return value = 1	API call parameters, shared secret	Status, derived key	KDF for IKEv2	User - Shared secret: W,E,Z - IKEv2 encryption key: G,R - IKEv2 authentication key: G,R
Perform key unwrap	Perform key unwrap	Indicator API return value = 1	API call parameters, decryption key, key	Status, decrypted key	AES-CCM for Wrap/Unwrap AES-GCM for Wrap/Unwrap AES+MAC for Key Wrap/Unwrap AES for Unauthenticated Key Unwrap (allowed) (legacy)	User - AES key: W,E,Z - AES-CCM key: W,E,Z - AES-GCM key: W,E,Z - AES-CMAC key: W,E,Z - HMAC key: W,E,Z - AES-GCM IV: E,Z

Table 11: Approved Services

4.4 Non-Approved Services

The module does not offer any non-Approved services. Thus, per *FIPS 140-3 IG* C.H, the module provides indicators for the use of Approved services through a combination of an explicit indication (via a global FIPS mode indicator) and an implicit indication (via the API return indicating the successful completion of the service.)

N/A for this module.

4.5 External Software/Firmware Loaded

The module does not load any software or firmware components 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 module comprises a single software component: the fipscanister.o object file. Its integrity is verified using a single embedded HMAC SHA-1 digest value. The module's default entry point 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.

5.2 Initiate on Demand

The CO can initiate the pre-operational self-test and conditional CASTs on demand of the module by re-instantiating the module, rebooting/power-cycling the host device, or issuing the `FIPS_selftest()` API command.

6. Operational Environment

6.1 Operational Environment Type and Requirements

The Wind River 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 deuring 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.

7. Physical Security

The cryptographic module is a multi-chip standalone software module and does not include physical security mechanisms. Therefore, per section *FIPS 140-3 IG* G.3, 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 Annex F of *ISO/IEC 19790:2012*.

9. Sensitive Security Parameters Management

9.1 Storage Areas

The table below lists sensitive security parameters (SSPs) storage areas for this module. Section 9.4 selects from the storage areas listed and specifies the appropriate parameter in the “Storage” column if applicable to a specific SSP.

Storage Area Name	Description	Persistence Type
RAM	SSP Stored in the RAM	Dynamic

Table 12: Storage Areas

The module is a software module and provides no direct access to persistent storage within the module. All SSPs reside in volatile memory only.

9.2 SSP Input-Output Methods

The table below lists SSP input and output methods for this module. Section 9.4 selects from the input and output methods listed and specifies the appropriate parameter in the “Inputs/Outputs” column if applicable to a specific SSP.

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
Input via API	External (calling application)	RAM (module)	Plaintext	Manual	Electronic	
Output via API	RAM (module)	External (calling application)	Plaintext	Manual	Electronic	

Table 13: SSP Input-Output Methods

All SSP input and output methods are electronic and are performed within the TOEPP.

9.3 SSP Zeroization Methods

The table below lists SSP zeroization methods for this module. Section 9.4 selects from the zeroization methods listed and specifies the appropriate parameter in the “Zeroization” column if applicable to a specific SSP.

Zeroization Method	Description	Rationale	Operator Initiation
API call	API call zeroizes SSPs.	The operator executing the API call overwrites the SSPs with zeroes, yielding the SSPs irretrievable.	Invoke the following API commands to zeroize the specified SSPs: * FIPS_rsa_free * FIPS_ecdsa_sig_free * FIPS_ec_key_free * FIPS_dh_free * OPENSSL_cleanse (HMAC, AES, CMAC)

Zeroization Method	Description	Rationale	Operator Initiation
Remove power	Upon removing power from the host device, the SSPs in memory are zeroized.	Removing power from the host device yields SSPs in memory irretrievable and unusable, effectively zeroizing them.	Operator removes power from the host device
Reboot	Upon rebooting the host device, the SSPs in memory are zeroized.	Rebooting the host device yields SSPs in memory irretrievable and unusable, effectively zeroizing them.	Operator reboots the host device
Power-cycle	Upon power-cycling the host device, the SSPs in memory are zeroized.	Power-cycling the host device yields SSPs in memory irretrievable and unusable, effectively zeroizing them.	Operator power-cycles the host device

Table 14: SSP Zeroization Methods

The module provides no means of persistent storage of SSPs and key components. Persistent storage and zeroization of unprotected SSPs passed into and out of the module are the responsibility of the calling application.

9.4 SSPs

The module supports the keys and other SSPs listed in the tables below.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES key	Used for symmetric encryption and decryption Used for key wrap and	Between 128 and 256 bits - Between 128 and 256 bits	Symmetric - CSP			AES for Symmetric Encryption AES for Symmetric Decryption AES+MAC for Key Wrap/Unwrap AES for Unauthenticated Key Unwrap (allowed) (legacy)
AES-CCM key	Used for symmetric encryption and decryption Used for key wrap	Between 128 and 256 bits - Between 128 and 256 bits	Symmetric - CSP			AES-CCM for Authenticated Symmetric Encryption AES-CCM for Authenticated Symmetric Decryption AES-CCM for Wrap/Unwrap

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES-GCM key	Used for authenticated symmetric encryption and decryption Used for key wrap and unwrap	Between 128 and 256 bits - Between 128 and 256 bits	Symmetric - CSP			AES-GCM for Authenticated Symmetric Encryption AES-GCM for Authenticated Symmetric Decryption AES-GCM for Wrap/Unwrap
AES-XTS key	Used for symmetric encryption and decryption	128 or 256 bits - 128 or 256 bits	Symmetric - CSP			AES-XTS for Symmetric Encryption AES-XTS for Symmetric Decryption
AES-CMAC key	Used for MAC generation and verification Used for symmetric encryption and decryption Used for key wrap/unwrap	Between 128 and 256 bits - Between 128 and 256 bits	Symmetric - CSP			AES-CMAC for MAC Generation AES-CMAC for MAC Verification AES+MAC for Key Wrap/Unwrap
HMAC key	Used for computing a MAC	224 bits (minimum) - 112 bits (minimum)	Authentication - CSP			HMAC for Message Authentication HMAC for Module Integrity Test AES+MAC for Key Wrap/Unwrap
RSA private key	Used for digital signature generation and asymmetric decryption	Between 2048 and 4096 bits - Between 112 and 150 bits	Private - CSP	RSA for Key Generation		RSA for Key Transport RSA (FIPS 186-5) for Signature Generation
RSA public key	Used for digital signature verification and asymmetric encryption	Between 2048 and 4096 bits - Between 112 and 150 bits	Public - PSP	RSA for Key Generation		RSA for Key Transport RSA (FIPS 186-4) for Signature Verification (legacy) RSA (FIPS 186-5) for Signature Verification
ECDSA private key	Used for digital signature generation	Between 224 and 512 bits - Between 112 and 256 bits	Private - CSP	ECDSA for Key Generation		ECDSA for Signature Generation
ECDSA public key	Used for digital signature verification	Between 224 and 512 bits - Between 112 and 256 bits	Public - PSP	ECDSA for Key Generation		ECDSA for Signature Verification

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DH private key	Used for computing a DH shared secret	Between 2048 and 8192 bits - Between 112 and 200 bits	Private - CSP	DH Key Generation		DH Shared Secret Computation
DH public key	Used for computing a DH shared secret	Between 2048 and 8192 bits - Between 112 and 200 bits	Public - PSP	DH Key Generation		DH Shared Secret Computation
ECDH private key	Used for computing an ECDH shared secret	Between 224 and 521 bits - Between 112 and 256 bits	Private - CSP	ECDH Key Generation		ECDH Shared Secret Computation
ECDH public key	Used for computing an ECDH shared secret	Between 224 and 521 bits - Between 112 and 256 bits	Public - PSP	ECDH Key Generation		ECDH Shared Secret Computation
IKEv2 encryption key	Output from the IKEv2 KDF	Between 128 and 256 bits - Between 128 and 256 bits	Symmetric - CSP	KDF for IKEv2		
Shared secret	Used as input to internal (IKEv2) and external protocol KDFs	N/A - N/A	N/A - CSP		DH Shared Secret Computation ECDH Shared Secret Computation	KDF for IKEv2
DRBG entropy input	Used in random bit generation (Counter, Hash, HMAC)	min 112 bits - min 112 bits	N/A - CSP	Entropy		CTR_DRBG Hash_DRBG HMAC_DRBG
DRBG seed	Used in random bit generation (Counter, Hash, HMAC DRBGs)	N/A - N/A	N/A - CSP	Entropy		CTR_DRBG Hash_DRBG HMAC_DRBG
DRBG 'C' value	Used in random bit generation (Hash DRBG)	N/A - N/A	N/A - CSP	Hash_DRBG		Hash_DRBG
DRBG 'V' value	Used in random bit generation (Counter, Hash, HMAC DRBGs)	N/A - N/A	N/A - CSP	CTR_DRBG Hash_DRBG HMAC_DRBG		CTR_DRBG Hash_DRBG HMAC_DRBG
DRBG 'Key" Value	Used in random bit generation (Counter, HMAC DRBGs)	N/A - N/A	N/A - CSP	CTR_DRBG HMAC_DRBG		CTR_DRBG HMAC_DRBG

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES-GCM IV	Used as initialization vector for AES-GCM	96 bits - N/A	N/A - CSP	Constructed per industry protocol specification or at its entirety internally deterministically		AES-GCM for Authenticated Symmetric Encryption AES-GCM for Authenticated Symmetric Decryption
IKEv2 authentication key	Output from the IKEv2 KDF	224 bits (minimum) - 112 bits (minimum)	Authentication - CSP	KDF for IKEv2		

Table 15: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES key	Input via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	API call Remove power Reboot Power-cycle	
AES-CCM key	Input via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	API call Remove power Reboot Power-cycle	
AES-GCM key	Input via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	API call Remove power Reboot Power-cycle	AES-GCM IV:Paired With
AES-XTS key	Input via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	API call Remove power Reboot Power-cycle	
AES-CMAC key	Input via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	API call Remove power Reboot Power-cycle	
HMAC key	Input via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	API call Remove power Reboot Power-cycle	
RSA private key	Input via API Output via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	API call Remove power Reboot Power-cycle	RSA public key:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
RSA public key	Input via API Output via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	API call Remove power Reboot Power-cycle	RSA private key:Paired With
ECDSA private key	Input via API Output via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	API call Remove power Reboot Power-cycle	ECDSA public key:Paired With
ECDSA public key	Input via API Output via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	API call Remove power Reboot Power-cycle	ECDSA private key:Paired With
DH private key	Input via API Output via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	API call Remove power Reboot Power-cycle	DH public key:Paired With Shared secret:Derives
DH public key	Input via API Output via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	API call Remove power Reboot Power-cycle	DH private key:Paired With Shared secret:Derives
ECDH private key	Input via API Output via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	API call Remove power Reboot Power-cycle	ECDH public key:Paired With Shared secret:Derives
ECDH public key	Input via API Output via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	API call Remove power Reboot Power-cycle	ECDSA private key:Paired With Shared secret:Derives
IKEv2 encryption key	Output via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	Remove power Reboot Power-cycle	Shared secret:Derived From
Shared secret	Input via API Output via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	Remove power Reboot Power-cycle	DH private key:Derived From DH public key:Derived From ECDH private key:Derived From ECDH public key:Derived From IKEv2 encryption key:Derives IKEv2 authentication key:Derives
DRBG entropy input	Input via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	Remove power Reboot Power-cycle	DRBG seed:Derives

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
DRBG seed	Input via API	RAM:Plaintext	Stored in RAM until a zeroization method is applied	Remove power Reboot Power-cycle	DRBG entropy input:Derived From DRBG 'V' value:Derives DRBG 'Key" Value:Derives DRBG 'C' value:Derives
DRBG 'C' value		RAM:Plaintext	Stored in RAM until a zeroization method is applied	Remove power Reboot Power-cycle	DRBG 'V' value:Paired With DRBG seed:Derived From
DRBG 'V' value		RAM:Plaintext	Stored in RAM until a zeroization method is applied	Remove power Reboot Power-cycle	DRBG 'C' value:Paired With DRBG 'Key" Value:Paired With DRBG seed:Derived From
DRBG 'Key" Value		RAM:Plaintext	Stored in RAM until a zeroization method is applied	Remove power Reboot Power-cycle	DRBG 'V' value:Paired With DRBG seed:Derived From
AES-GCM IV		RAM:Plaintext	Stored in RAM until a zeroization method is applied	Remove power Reboot Power-cycle	AES-GCM key:Paired With
IKEv2 authentication key	Output via API		RAM: Plaintext	Remove power Reboot Power-cycle	Shared secret:Derived From

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:

- The module includes an implementation of SHA-1 for MAC generation and digital signature verification. SHA-1 will be non-Approved for all uses starting January 1, 2031.

- In compliance with *NIST SP 800-131Arev2*, the module supports algorithms and key lengths that provide a minimum of 112 bits of security strength for applying cryptographic protection. Starting January 1, 2031, the minimum security strength for applying cryptographic protection will be 128 bits, and security strengths between 112 bits and 128 bits will be allowed for legacy use only to process information that is already protected.

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-test(s) listed in the following table.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
Integrity Test (Wind River FIPS Cryptographic Module (libcrypto))	HMAC SHA-1	KAT	SW/FW Integrity	Returns "1" for success and "0" for failure	MAC verification

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-CBC (A6780) - Encrypt	128-bit	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Encrypt	On initial power-up sequence and prior to first use of the algorithm
AES-CBC (A6780) - Decrypt	128-bit	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Decrypt	On initial power-up sequence and prior to first use of the algorithm.
AES-CCM (A6780) - Encrypt	192-bit	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Encrypt	On initial power-up sequence and prior to first use of the algorithm
AES-CCM (A6780) - Decrypt	192-bit	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Decrypt	On initial power-up sequence and prior to first use of the algorithm
AES-GCM (A6780) - Encrypt	256-bit	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Encrypt	On initial power-up sequence and prior to first use of the algorithm
AES-GCM (A6780) - Decrypt	256-bit	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Decrypt	On initial power-up sequence and prior to first use of the algorithm

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-XTS Testing Revision 2.0 (A6780) - Encrypt	128/256-bit	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Encrypt	On initial power-up sequence and prior to first use of the algorithm
AES-XTS Testing Revision 2.0 (A6780) - Decrypt	128/256-bit	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Decrypt	On initial power-up sequence and prior to first use of the algorithm
AES-CMAC (A6780) - Generate	128/256-bit	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Generate	On initial power-up sequence and prior to first use of the algorithm
AES-CMAC (A6780) - Verify	128/256-bit	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Verify	On initial power-up sequence and prior to first use of the algorithm
Counter DRBG (A6780)	N/A	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Generate/Instantiate/Reseed	On initial power-up sequence and prior to first use of the algorithm
Hash DRBG (A6780)	N/A	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Generate/Instantiate/Reseed	On initial power-up sequence and prior to first use of the algorithm
HMAC DRBG (A6780)	N/A	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Generate/Instantiate/Reseed	On initial power-up sequence and prior to first use of the algorithm
HMAC-SHA-1 (A6780)	N/A	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	MAC	On initial power-up sequence and prior to first use of the algorithm
HMAC-SHA2-256 (A6780)	N/A	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	MAC	On initial power-up sequence and prior to first use of the algorithm
HMAC-SHA2-512 (A6780)	N/A	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	MAC	On initial power-up sequence and prior to first use of the algorithm
ECDSA SigGen (FIPS186-5) (A6780)	P-384	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Sign	On initial power-up sequence and prior to first use of the algorithm
ECDSA SigVer (FIPS186-5) (A6780)	P-384	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Sign	On initial power-up sequence and prior to first use of the algorithm
RSA SigGen (FIPS186-5) (A6780)	2048	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Sign	On initial power-up sequence and prior to first use of the algorithm
RSA SigVer (FIPS186-5) (A6780)	2048	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Verify	On initial power-up sequence and prior to first use of the algorithm

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
RSA Encrypt	2048-bit	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Encrypt	On initial power-up sequence and prior to first use of the algorithm
RSA Decrypt	2048-bit	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Decrypt	On initial power-up sequence and prior to first use of the algorithm
KAS-ECC-SSC Sp800-56Ar3 (A6780)	P-224	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Shared secret computation	On initial power-up sequence and prior to first use of the algorithm
KDF IKEv2 (A6780)	SHA2-256	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Key derivation	On initial power-up sequence and prior to first use of the algorithm
ECDSA KeyGen (FIPS186-5) (A6780)	N/A	PCT	PCT	Sets return value to "0" if test is failed (default value is "1")	Sign/Verify	Prior to using an EDSA keypair as part of an ECDSA
RSA KeyGen (FIPS186-5) (A6780)	N/A	PCT	PCT	Sets return value to "0" if test is failed (default value is "1")	Sign/Verify	Prior to using an RSA keypair as part of an RSA sign/verify function
RSA KeyGen KTS (A6780)	N/A	PCT	PCT	Sets return value to "0" if test is failed (default value is "1")	Encrypt/Decrypt	Prior to using an RSA keypair as part of an RSA key transport function
DH KeyGen	N/A	PCT	PCT	Sets return value to "0" if test is failed (default value is "1")	Key Generation	Prior to using a DH keypair as part of an DH shared secret computation
ECDH KeyGen	N/A	PCT	PCT	Sets return value to "0" if test is failed (default value is "1")	Key generation	Prior to using an ECDH keypair as part of an ECDH key agreement function
AES-ECB (A6780) - Encrypt	128-bit	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Encrypt	On initial power-up sequence and prior to first use of the algorithm
AES-ECB (A6780) - Decrypt	128-bit	KAT	CAST	Sets return value to "0" if test is failed (default value is "1")	Decrypt	On initial power-up sequence and prior to first use of the algorithm
AES-XTS Testing Revision 2.0 (A6780)	N/A	Duplicate Key Test	Critical Function	Sets return value to "0" if test is failed (default value is "1")	Key comparison	Prior to using AES-XTS keys for encryption

Table 18: Conditional Self-Tests

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 HMAC and SHA 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.

10.3 Periodic Self-Test Information

The CO can initiate the pre-operational self-test 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 tables below specify the period and the policy for these conditions.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
Integrity Test (Wind River FIPS Cryptographic Module (libcrypto))	KAT	SW/FW Integrity	On Demand	Manually

Table 19: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC (A6780) - Encrypt	KAT	CAST	on demand	Manually
AES-CBC (A6780) - Decrypt	KAT	CAST	On Demand	Manually
AES-CCM (A6780) - Encrypt	KAT	CAST	On Demand	Manually
AES-CCM (A6780) - Decrypt	KAT	CAST	On Demand	Manually
AES-GCM (A6780) - Encrypt	KAT	CAST	On Demand	Manually
AES-GCM (A6780) - Decrypt	KAT	CAST	On Demand	Manually
AES-XTS Testing Revision 2.0 (A6780) - Encrypt	KAT	CAST	On Demand	Manually
AES-XTS Testing Revision 2.0 (A6780) - Decrypt	KAT	CAST	On Demand	Manually
AES-CMAC (A6780) - Generate	KAT	CAST	On Demand	Manually
AES-CMAC (A6780) - Verify	KAT	CAST	On Demand	Manually
Counter DRBG (A6780)	KAT	CAST	On Demand	Manually
Hash DRBG (A6780)	KAT	CAST	On Demand	Manually
HMAC DRBG (A6780)	KAT	CAST	On demand	Manually
HMAC-SHA-1 (A6780)	KAT	CAST	On Demand	Manually
HMAC-SHA2-256 (A6780)	KAT	CAST	On Demand	Manually
HMAC-SHA2-512 (A6780)	KAT	CAST	On Demand	Manually
ECDSA SigGen (FIPS186-5) (A6780)	KAT	CAST	On Demand	Manually
ECDSA SigVer (FIPS186-5) (A6780)	KAT	CAST	On Demand	Manually
RSA SigGen (FIPS186-5) (A6780)	KAT	CAST	On Demand	Manually
RSA SigVer (FIPS186-5) (A6780)	KAT	CAST	On Demand	Manually
RSA Encrypt	KAT	CAST	On Demand	Manually
RSA Decrypt	KAT	CAST	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
KAS-ECC-SSC Sp800-56Ar3 (A6780)	KAT	CAST	On Demand	Manually
KDF IKEv2 (A6780)	KAT	CAST	On Demand	Manually
ECDSA KeyGen (FIPS186-5) (A6780)	PCT	PCT	On Demand	Manually
RSA KeyGen (FIPS186-5) (A6780)	PCT	PCT	On Demand	Manually
RSA KeyGen KTS (A6780)	PCT	PCT	On Demand	Manually
DH KeyGen	PCT	PCT	On Demand	Manually
ECDH KeyGen	PCT	PCT	On Demand	Manually
AES-ECB (A6780) - Encrypt	KAT	CAST	On Demand	Manually
AES-ECB (A6780) - Decrypt	KAT	CAST	On Demand	Manually
AES-XTS Testing Revision 2.0 (A6780)	Duplicate Key Test	Critical Function	On Demand	Manually

Table 20: Conditional Periodic Information

10.4 Error States

The table below describes the error states and status indicators of the module.

Name	Description	Conditions	Recovery Method	Indicator
Critical Error	Will prevent the execution of any cryptographic services while the error state persists. Subsequent requests will return an error result.	Upon failure of any of the module's pre-operational or conditional self-tests.	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 or conditional CASTs, the module should not resume normal operations, and Wind River Systems, Inc. should be contacted for assistance.	FIPS_self_test_fail error flag will be set.

Table 21: Error States

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

Module operators shall follow all guidance specified in this document. 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

11.1.1 Secure Installation

The module is distributed to 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 2.2.4 above or one where portability is maintained.

11.1.2 Initialization

The Wind River FIPS Object Module is not a standalone application; it is a cryptographic toolkit designed to be linked to vendor applications, and these applications are the sole consumers of the cryptographic services provided by the module. 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 pre-operational integrity test and conditional CASTs 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_module_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.

Failure of any of the initialization actions will result in a failure of the module to load for execution.

11.1.3 Startup

There are no module startup steps required to be performed by end-users.

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 Wind River Customer Support should be contacted.

The following list provides additional guidance for the CO:

- 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.

- In the event that the module encounters a DRBG self-test failure, the calling application must uninstantiate and re-instantiate the DRBG per the requirements found in *NIST SP 800-90Arev1*.

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.
- There are no restrictions on which keys or SSPs are zeroized by the zeroization service.
- The module does not support a maintenance interface or role.
- The module does not support manual SSP establishment method.
- The module does not have any proprietary external input/output devices used for entry/output of data.
- The module does not store any plaintext CSPs
- 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 above. This will ensure that any SSPs in volatile memory are zeroized.

12. Mitigation of Other Attacks

The module does not claim to mitigate any attacks beyond the FIPS 140-3 Level 1 requirements for this validation. Therefore, per *ISO/IEC 19790:2012* section 7.12, requirements for this section are not applicable.

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
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
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
OS	Operating System
PCT	Pairwise Consistency Test
PKCS	Public Key Cryptography Standard

Term	Definition
PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, and Adleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SP	Special Publication
XEX	XOR Encrypt XOR
XTS	XEX-Based Tweaked-Codebook Mode with Ciphertext Stealing

Prepared by:
Corsec Security, Inc.

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

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