Riverbed Technology, LLC

Riverbed Cryptographic Module

Software Version: 2.0.1

FIPS 140-3 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 0.5

Prepared for:

Riverbed Technology, LLC
275 Shoreline Drive
Redwood City, CA 94065
United States of America

Phone: +1 415.247.8800
www.riverbed.com

Prepared by:

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

Phone: +1 703.267.6050
www.corsec.com

Table of Contents

List of Tables

List of Figures

1. General

1.1 Overview

1.1.1 Abstract

This is a non-proprietary Cryptographic Module Security Policy for the Riverbed Cryptographic Module (software version: 2.0.1) from Riverbed Technology, LLC (Riverbed). This Security Policy describes how the Riverbed Cryptographic Module meets the security requirements of Federal Information Processing Standards (FIPS) Publication 140-3, which details the U.S. and Canadian government requirements for cryptographic modules. More information about the FIPS 140-3 standard and validation program is available on the Cryptographic Module Validation Program (CMVP) website, which is maintained by the National Institute of Standards and Technology (NIST) and the Canadian Centre for Cyber Security (CCCS).

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

1.1.2 References

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

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

1.1.3 Document Organization

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

1.2 Security Levels

The Riverbed Cryptographic Module is validated at the FIPS 140-3 section levels shown in the table below.

Section	Title	Security Level
1	General	1
2	Cryptographic module specification	1

Section	Title	Security Level
3	Cryptographic module interfaces	1
4	Roles, services, and authentication	1
5	Software/Firmware security	1
6	Operational environment	1
7	Physical security	N/A
8	Non-invasive security	N/A
9	Sensitive security parameter management	1
10	Self-tests	1
11	Life-cycle assurance	1
12	Mitigation of other attacks	N/A
	Overall Level	1

Table 1: Security Levels

The module has an overall security level of 1.

2. Cryptographic Module Specification

2.1 Description

2.1.1 Purpose and Use

Since its inception in 2002, Riverbed Technology, LLC has helped the world's largest organizations maximize the performance of their networks and applications so they can reach the full potential of their IT investments. Riverbed's products consist of software and hardware focused on network performance monitoring, application performance management, and wide area networks (WANs).

The Riverbed Network and Application Performance Platform enables organizations to visualize, optimize, accelerate, and remediate the performance of any network for any application. Only Riverbed addresses performance and visibility holistically with best-in-class WAN optimization, network performance management, application acceleration, and enterprise-grade SD-WAN[1].

The Riverbed Cryptographic Module v2.0.1 is a software library providing a C language API[2] for use by Riverbed applications requiring cryptographic functionality. The Riverbed Cryptographic Module offers symmetric encryption/decryption, digital signature generation/verification, hashing, cryptographic key generation, random number generation, message authentication, and key establishment functions to secure data-at-rest/data-in-flight and to support secure communications protocols (including TLS[3] 1.2/1.3).

2.1.2 Module Type

The Riverbed Cryptographic Module 2.0.1 is a Software module.

2.1.3 Module Embodiment

The Riverbed Cryptographic Module has a MultiChipStand embodiment.

2.1.4 Cryptographic Boundary

The cryptographic boundary is the contiguous perimeter that surrounds all memory-mapped functionality provided by the module when loaded and stored in the host platform's memory. Figure 2 is a block diagram of the module executing in memory and its interactions with surrounding software components, as well as the module's cryptographic boundary and Tested Operational Environment's Physical Perimeter (TOEPP).

[1] SD-WAN – Software-Defined Wide Area Network
[2] API – Application Programming Interface
[3] TLS – Transport Layer Security

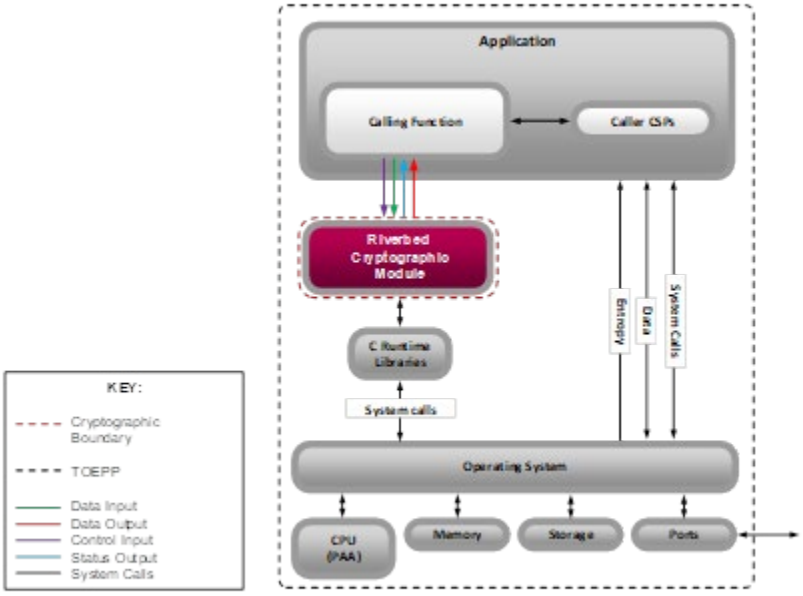

Figure 1. Module Block Diagram (with Cryptographic Boundary)

The module is entirely contained within the physical perimeter.

2.1.5 Tested Operational Environment's Physical Perimeter (TOEPP)

As a software cryptographic module, the TOEPP of the cryptographic module is defined by each host platform on which the module is installed. Figure 2 below illustrates a block diagram of a typical GPC (the black dotted line represents the module's physical perimeter).

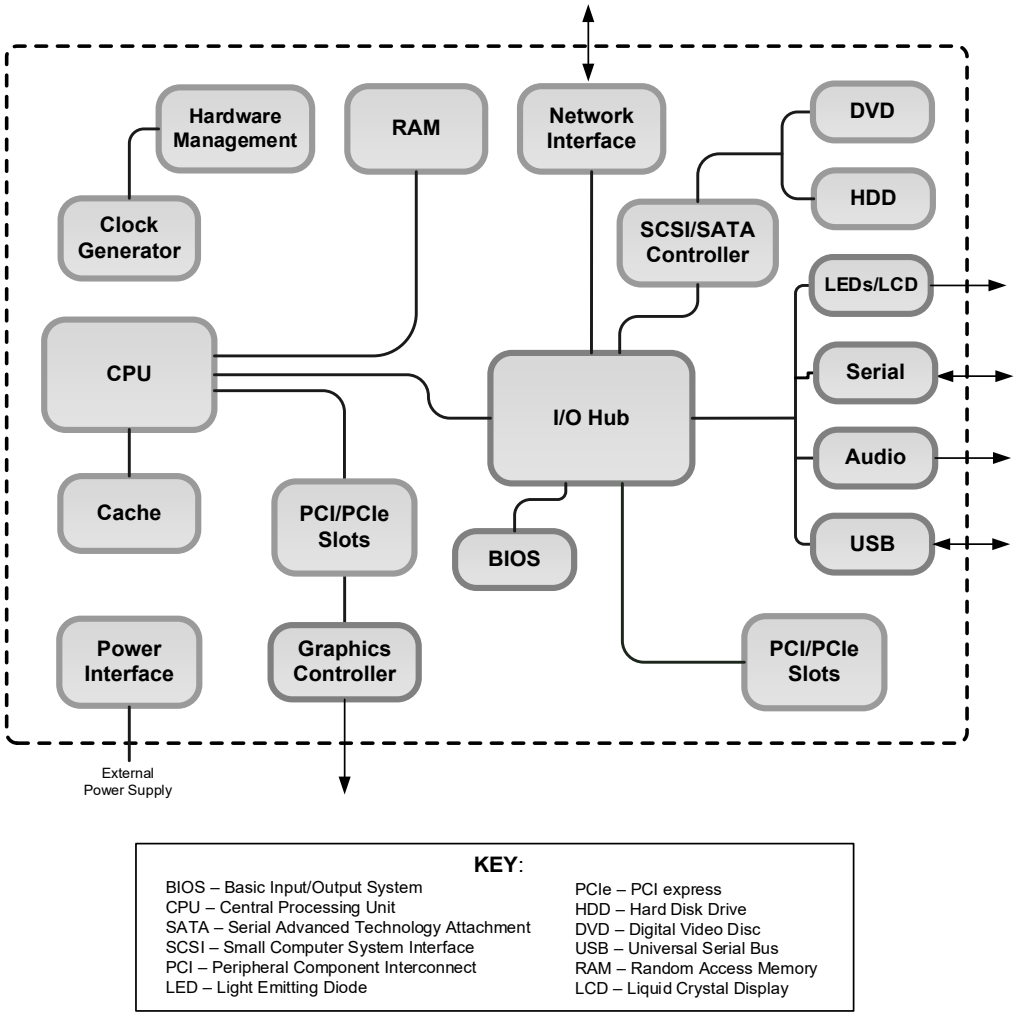

Figure 2. GPC Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

2.2.1 Tested Module Identification – Hardware

This section is only applicable for hardware modules.

N/A for this module.

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

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libcrypto.so	2.0.1	N/A	Yes
libssl.so	2.0.1	N/A	Yes

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

2.2.3 Tested Module Identification – Hybrid Disjoint Hardware

This section is only applicable to hybrid modules.

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)
AlmaLinux 8	Riverbed AppResponse 2180	Intel Xeon Silver 4110	Yes	N/A	2.0.1
AlmaLinux 8	Riverbed AppResponse 2180	Intel Xeon Silver 4110	No	N/A	2.0.1

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

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

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

There are no vendor-affirmed operational environments claimed.

N/A for this module.

2.3 Excluded Components

The module does not exclude any components from the requirements.

2.4 Modes of Operation

2.4.1 Modes List and Description

The module supports two modes of operation: Approved and non-Approved. These operational modes are described in the table below.

Mode Name	Description	Type	Status Indicator
Approved	The module switches between the Approved mode and Non-Approved mode depending on the service executed. The module is in this mode once all pre-operational self-tests have completed successfully, and only Approved services are invoked.	Approved	Indicator API return value = 1
Non-Approved	The module will switch to the non-Approved mode upon execution of a non-Approved service.	Non-Approved	Indicator API return value other than 1

Table 4: Modes List and Description

Section 4.3 of this Security Policy lists the services that constitute the Approved mode of operation. Section 4.4 below lists the services that constitute the non-Approved mode. When following the guidance in section 11.3 of this Security Policy, CSPs are not shared between Approved and non-Approved services and modes of operation.

The module does not support degraded operation.

2.5 Algorithms

2.5.1 Approved Algorithms

The module employs cryptographic algorithm implementations from the following sources:

- Riverbed Cryptographic Module (libcrypto) version 2.0.1 (Cert. A5835)
- Riverbed Cryptographic Module (libssl) version 2.0.1 (Cert. A5836)

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

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A5835	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A5835	Key Length - 128, 192, 256	SP 800-38C
AES-CFB1	A5835	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB128	A5835	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A5835	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A5835	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B
AES-CTR	A5835	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A5835	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
AES-GCM	A5835	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.2 Key Length - 128, 192, 256	SP 800-38D
AES-GMAC	A5835	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-KW	A5835	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KWP	A5835	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-OFB	A5835	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-XTS Testing Revision 2.0	A5835	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E
Counter DRBG	A5835	Prediction Resistance - No, Yes Mode - AES-128, AES-192, AES-256 Derivation Function Enabled - Yes	SP 800-90A Rev. 1
DSA KeyGen (FIPS186-4)	A5835	L - 2048, 3072 N - 224, 256	FIPS 186-4
DSA PQGGen (FIPS186-4)	A5835	L - 2048, 3072 N - 224, 256 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
DSA PQGVer (FIPS186-4)	A5835	L - 1024, 2048, 3072 N - 160, 224, 256 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
DSA SigGen (FIPS186-4)	A5835	L - 2048, 3072 N - 224, 256 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
DSA SigVer (FIPS186-4)	A5835	L - 2048, 3072 N - 224, 256 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
ECDSA KeyGen (FIPS186-4)	A5835	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Secret Generation Mode - Testing Candidates	FIPS 186-4
ECDSA KeyVer (FIPS186-4)	A5835	Curve - B-163, B-233, B-283, B-409, B-571, K-163, K-233, K-283, K-409, K-571, P-192, P-224, P-256, P-384, P-521	FIPS 186-4
ECDSA SigGen (FIPS186-4)	A5835	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A5835	Curve - B-163, B-233, B-283, B-409, B-571, K-163, K-233, K-283, K-409, K-571, P-192, P-224, P-256, P-384, P-521 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
HMAC-SHA-1	A5835	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A5835	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A5835	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A5835	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A5835	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-224	A5835	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-256	A5835	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-384	A5835	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-512	A5835	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1

Algorithm	CAVP Cert	Properties	Reference
KAS-ECC-SSC Sp800-56Ar3	A5835	Domain Parameter Generation Methods - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-FFC-SSC Sp800-56Ar3	A5835	Domain Parameter Generation Methods - FB, FC Scheme - dhEphem - KAS Role - initiator, responder	SP 800-56A Rev. 3
PBKDF	A5835	Iteration Count - Iteration Count: 10-10000 Increment 1 Password Length - Password Length: 8-128 Increment 1	SP 800-132
RSA KeyGen (FIPS186-4)	A5835	Key Generation Mode - B.3.3 Modulo - 2048, 3072, 4096 Primality Tests - Table C.2 Private Key Format - Standard	FIPS 186-4
RSA SigGen (FIPS186-4)	A5835	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-4)	A5835	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096	FIPS 186-4
SHA-1	A5835	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-224	A5835	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-256	A5835	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-384	A5835	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-512	A5835	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA3-224	A5835	Message Length - Message Length: 0-65528 Increment 8	FIPS 202
SHA3-256	A5835	Message Length - Message Length: 0-65528 Increment 8	FIPS 202
SHA3-384	A5835	Message Length - Message Length: 0-65528 Increment 8	FIPS 202
SHA3-512	A5835	Message Length - Message Length: 0-65528 Increment 8	FIPS 202
SHAKE-128	A5835	Output Length - Output Length: 16-1024 Increment 8	FIPS 202
SHAKE-256	A5835	Output Length - Output Length: 16-1024 Increment 8	FIPS 202
TDES-CBC	A5835	Direction - Decrypt	SP 800-67 Rev. 2
TDES-CFB1	A5835	Direction - Decrypt	SP 800-67 Rev. 2
TDES-CFB64	A5835	Direction - Decrypt	SP 800-67 Rev. 2
TDES-CFB8	A5835	Direction - Decrypt	SP 800-67 Rev. 2
TDES-CMAC	A5835	Direction - Verification	SP 800-67 Rev. 2
TDES-ECB	A5835	Direction - Decrypt	SP 800-67 Rev. 2
TDES-OFB	A5835	Direction - Decrypt	SP 800-67 Rev. 2
TLS v1.2 KDF RFC7627 (CVL)	A5835	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
TLS v1.3 KDF (CVL)	A5836	HMAC Algorithm - SHA2-256, SHA2-384 KDF Running Modes - DHE, PSK, PSK-DHE	SP 800-135 Rev. 1

Table 5: Approved Algorithms

2.5.2 Vendor-Affirmed Algorithms

The vendor affirms the following cryptographic security methods:

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

Name	Properties	Implementation	Reference
CKG1	Key Type:Asymmetric	Riverbed Cryptographic Module (libcrypto)	SP 800-133 Rev. 2 Section 4.

Table 6: Vendor-Affirmed Algorithms

2.5.3 Non-Approved, Allowed Algorithms

The table below lists the non-Approved algorithms implemented by the module that are allowed for use in the Approved mode of operation.

Name	Properties	Implementation	Reference
AES-CBC	Key unwrapping:128, 192, 256	Riverbed Cryptographic Module (libcrypto)	FIPS 197, SP 800-38A, FIPS 140-3 IG D.G
AES-CFB1	Key unwrapping:128, 192, 256	Riverbed Cryptographic Module (libcrypto)	FIPS 197, SP 800-38A, FIPS 140-3 IG D.G
AES-CFB128	Key unwrapping:128, 192, 256	Riverbed Cryptographic Module (libcrypto)	FIPS 197, SP 800-38A, FIPS 140-3 IG D.G
AES-CFB8	Key unwrapping:128, 192, 256	Riverbed Cryptographic Module (libcrypto)	FIPS 197, SP 800-38A, FIPS 140-3 IG D.G
AES-CTR	Key unwrapping:128, 192, 256	Riverbed Cryptographic Module (libcrypto)	FIPS 197, SP 800-38A, FIPS 140-3 IG D.G
AES-ECB	Key unwrapping:128, 192, 256	Riverbed Cryptographic Module (libcrypto)	FIPS 197, SP 800-38A, FIPS 140-3 IG D.G
AES-OFB	Key unwrapping:128, 192, 256	Riverbed Cryptographic Module (libcrypto)	FIPS 197, SP 800-38A, FIPS 140-3 IG D.G
TDES-CBC (2-key or 3-key)	Key unwrapping:	Riverbed Cryptographic Module (libcrypto)	SP 800-67 Rev. 2, SP 800-38A, FIPS 140-3 IG D.G
TDES-CFB1 (2-key or 3-key)	Key unwrapping:	Riverbed Cryptographic Module (libcrypto)	SP 800-67 Rev. 2, SP 800-38A, FIPS 140-3 IG D.G
TDES-CFB64 (2-key or 3-key)	Key unwrapping:	Riverbed Cryptographic Module (libcrypto)	SP 800-67 Rev. 2, SP 800-38A, FIPS 140-3 IG D.G
TDES-CFB8 (2-key or 3-key)	Key unwrapping:	Riverbed Cryptographic Module (libcrypto)	SP 800-67 Rev. 2, SP 800-38A, FIPS 140-3 IG D.G
TDES-ECB (2-key or 3-key)	Key unwrapping:	Riverbed Cryptographic Module (libcrypto)	SP 800-67 Rev. 2, SP 800-38A, FIPS 140-3 IG D.G
TDES-OFB (2-key or 3-key)	Key unwrapping:	Riverbed Cryptographic Module (libcrypto)	SP 800-67 Rev. 2, SP 800-38A, FIPS 140-3 IG D.G

Table 7: Non-Approved, Allowed Algorithms

2.5.4 Non-Approved, Allowed Algorithms with No Security Claimed

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

N/A for this module.

2.5.5 Non-Approved, Not Allowed Algorithms

The table below lists the non-Approved algorithms that are not allowed for use in the Approved mode of operation.

Name	Use and Function
AES-GCM (non-compliant)	Authenticated encryption/decryption using external IV
AES-OCB	Authenticated encryption/decryption
ANSI X9.31 RNG (non-compliant)	Random number generation using with 128-bit AES core
ARIA	Encryption/decryption
Blake2	Encryption/decryption
Blowfish	Encryption/decryption
Camellia	Encryption/decryption
CAST, CAST5	Encryption/decryption
ChaCha20	Encryption/decryption
DES	Encryption/decryption
DH (non-compliant)	Key agreement (non-compliant with key sizes below 2048)
DRBG (non-compliant)	Random bit generation (non-compliant when using Hash_DRBG and HMAC_DRBG)
DSA (non-compliant)	Key pair generation; digital signature generation; digital signature verification (non-compliant with key sizes below the minimums for Approved mode)
DSA, ECDSA, and RSA (non-compliant)	Digital signature generation (non-compliant when used with SHA-1 outside the TLS protocol)
ECDH (non-compliant)	Key agreement (non-compliant with curves P-192, K-163, B-163, and non-NIST curves)
ECDSA (non-compliant)	Key pair generation; digital signature generation; digital signature verification (non-compliant with curves P-192, K-163, B-163, and non-NIST curves)
EdDSA	Key pair generation; digital signature generation; digital signature verification
HKDF	HMAC-based key derivation
IDEA	Encryption/decryption
MD2, MD4, MD5	Message digest
Poly1305	Message authentication code
RC2, RC4, RC5	Encryption/decryption
RIPEMD	Message digest
RMD160	Message digest
RSA (non-compliant)	Key pair generation; digital signature generation; signature verification; key transport (non-compliant with non-approved/untested key sizes, and functions)
SEED	Encryption/decryption
SHA-1 (non-compliant)	Signature generation in TLS 1.0/1.1
SM2, SM3	Message digest
SM4	Encryption/decryption
TLS 1.2 KDF (non-compliant)	Key derivation function per (RFC 5246)
Triple-DES (non-compliant)	Encryption; MAC generation; key wrapping
Whirlpool	Message digest

Table 8: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

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

Name	Type	Description	Properties	Algorithms
AES for Symmetric Encryption/Decryption	BC-UnAuth	AES for the AES key, which is used for symmetric encryption and decryption.	Publication:SP 800-38A	AES-CBC AES-CFB1 AES-CFB8 AES-CFB128 AES-CTR AES-ECB AES-OFB
AES-CMAC for MAC Generation/Verification	MAC	AES-CMAC for the AES CMAC key, which is used for MAC generation and verification.	Publication:SP 800-38B	AES-CMAC
AES-GMAC for MAC Generation/Verification	MAC	AES for the AES GMAC key, which is used for MAC generation and verification.	Publication:SP 800-38D	AES-GMAC AES-CTR
AES-CCM for Authenticated Symmetric Encryption/Decryption	BC-Auth	AES-CCM for the AES CCM key, which is used for authenticated symmetric encryption and decryption.	Publication:SP 800-38C	AES-CCM AES-CBC
AES-GCM for Authenticated Symmetric Encryption/Decryption	BC-Auth	AES-GCM for the AES GCM key, which is used for authenticated symmetric encryption and decryption.	Publication:SP 800-38D	AES-GCM AES-CTR Counter DRBG
AES-XTS for Symmetric Encryption/Decryption	BC-UnAuth	AES-XTS for the AES XTS key, which is used for symmetric encryption and decryption.	Publication:SP 800-38E	AES-XTS Testing Revision 2.0 AES-ECB Counter DRBG
KTS-AES+MAC	KTS-Wrap	AES-CMAC for the AES CMAC key, which is used for key transport.	Publication:Publication: Per FIPS 140-3 Implementation Guidance D.G, any Approved mode of AES with CMAC is an Approved key transport technique. Key Strength:Key establishment methodology provides between 128 and 256 bits of encryption strength.	AES-CMAC AES-GMAC HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512 HMAC-SHA3-224 HMAC-SHA3-256 HMAC-SHA3-384 HMAC-SHA3-512 AES-CBC AES-CFB1 AES-CFB8 AES-CFB128 AES-CTR AES-ECB AES-OFB

Name	Type	Description	Properties	Algorithms
KTS-AES-CCM	KTS-Wrap	AES-CCM for the AES CCM key, which is used for key transport.	Publication:Per FIPS 140-3 Implementation Guidance D.G, AES-CCM is an Approved key transport technique. Key Strength:Key establishment methodology provides between 128 and 256 bits of encryption strength.	AES-CCM AES-CTR
KTS-AES-GCM	KTS-Wrap	AES-GCM for the AES GCM key, which is used for key transport.	Publication:Per FIPS 140-3 Implementation Guidance D.G, AES-GCM is an Approved key transport technique. Key Strength:Key establishment methodology provides between 128 and 256 bits of encryption strength.	AES-GCM AES-CTR
KTS-AES-KW	KTS-Wrap	AES-KW and AES-KWP for the AES key, which is used for key transport.	Publication:SP 800-38F Key Strength:Key establishment methodology provides between 128 and 256 bits of encryption strength.	AES-KW AES-KWP
DRBG	DRBG	Deterministic random bit generator	Publication:SP 800-90A	Counter DRBG
DSA KeyGen for DH	AsymKeyPair-KeyGen	Key generation of the DSA private component and the DSA public component.	Publication:FIPS 186-4	DSA KeyGen (FIPS186-4) Counter DRBG CKG1
DSA for Digital Signature Verification (legacy)	DigSig-SigVer	DSA digital signature verification for the DSA public key.	Publication:FIPS 186-4 Publication:FIPS 140-3 IG C.M	DSA SigVer (FIPS186-4) SHA-1 SHA2-224 SHA2-256 SHA2-384 SHA2-512
ECDSA for Key Generation	AsymKeyPair-KeyGen	Key generation of the ECDSA private key and ECDSA public key.	Publication:FIPS 186-4	ECDSA KeyGen (FIPS186-4) Counter DRBG CKG1
ECDSA KeyGen for ECDH	AsymKeyPair-KeyGen	Key generation of the ECDH private component and the ECDSA public component.	Publication:FIPS 186-4	ECDSA KeyGen (FIPS186-4) Counter DRBG CKG1
ECDSA for Key Verification	AsymKeyPair-KeyVer	Public key validation	Publication:FIPS 186-4	ECDSA KeyVer (FIPS186-4)

Name	Type	Description	Properties	Algorithms
ECDSA for Digital Signature Generation	DigSig-SigGen	ECDSA digital signature generation for the ECDSA private key.	Publication:FIPS 186-4	ECDSA SigGen (FIPS186-4) Counter DRBG SHA2-224 SHA2-256 SHA2-384 SHA2-512
ECDSA for Digital Signature Verification	DigSig-SigVer	ECDSA digital signature verification for the ECDSA public key.	Publication:FIPS 186-4	ECDSA SigVer (FIPS186-4) SHA2-224 SHA2-256 SHA2-384 SHA2-512
HMAC for Message Authentication	MAC	Message Authentication	Publication:FIPS 198-1	HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512 HMAC-SHA3-224 HMAC-SHA3-384 HMAC-SHA3-512 SHA-1 SHA2-224 SHA2-256 SHA2-384 SHA2-512 SHA3-224 SHA3-256 SHA3-384 SHA3-512
ECDH Shared Secret Computation	KAS-SSC	Shared secret computation for ECDH.	Publication:SP 800-56A Rev. 3 Publication:SP 800-90A Rev. 1 Publication:SP 800-133 Rev. 2 Publication:FIPS 140-3 IG D.F Scenario 2(1)	KAS-ECC-SSC Sp800-56Ar3 ECDSA KeyGen (FIPS186-4) ECDSA KeyVer (FIPS186-4) SHA2-224 SHA2-256 SHA2-384 SHA2-512 Counter DRBG HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512

Name	Type	Description	Properties	Algorithms
DH Shared Secret Computation	KAS-SSC	Shared secret computation for DH.	Publication:SP800 56A Rev.3 Publication:SP 800-90A Rev. 1 Publication:SP 800-133 Rev. 2 Publication:FIPS 140-3 IG D.F Scenario 2(1)	KAS-FFC-SSC Sp800-56Ar3 DSA PQGGen (FIPS186-4) DSA KeyGen (FIPS186-4) SHA2-224 SHA2-256 SHA2-384 SHA2-512 Counter DRBG HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512
PBKDF	PBKDF	Password-based key derivation	Publication: SP 800-132	PBKDF SHA-1 SHA2-224 SHA2-256 SHA2-384 SHA2-512 SHA3-224 SHA3-256 SHA3-384 SHA3-512
RSA for Key Generation	AsymKeyPair-KeyGen	RSA digital signature generation for the RSA private key.	Publication:FIPS 186-4	RSA KeyGen (FIPS186-4) Counter DRBG CKG1
RSA for Signature Generation	DigSig-SigGen	RSA digital signature generation for the RSA private key.	Publication:FIPS 186-4	RSA SigGen (FIPS186-4) SHA2-224 SHA2-256 SHA2-384 SHA2-512 Counter DRBG
RSA for Signature Verification	DigSig-SigVer	RSA signature verification for the RSA public key.	Publication:FIPS 186-4	RSA SigVer (FIPS186-4) SHA2-224 SHA2-256 SHA2-384 SHA2-512
SHA/SHAKE for Message Digest	SHA XOF	Message Digest	Publication:FIPS 180-4	SHA-1 SHA2-224 SHA2-256 SHA2-384 SHA2-512 SHA3-224 SHA3-256 SHA3-384 SHA3-512 SHAKE-128 SHAKE-256
TDES for Symmetric Decryption (legacy)	BC-UnAuth	TDES for the TDES key, which is used for symmetric decryption.	Publication:SP 800-67 Rev. 2	TDES-CBC TDES-CFB1 TDES-CFB64 TDES-CFB8 TDES-ECB TDES-OFB

Name	Type	Description	Properties	Algorithms
TDES for MAC Verification (legacy)	MAC	TDES-CMAC for the TDES CMAC key, which is used for MAC verification.	Publication:SP 800-67 Rev. 2	TDES-CMAC
TLS1.2-KDF (CVL)	KAS-135KDF	TLS 1.2 key derivation, used to derive the TLS Session Key (AES key or AES-GCM key) and TLS Authentication Key (HMAC key).	Publication :SP 800-135 Rev. 1 Caveat:No part of the TLS v1.2 protocol, other than the KDF, has been tested by the CAVP and CMVP.	TLS v1.2 KDF RFC7627 SHA2-256 SHA2-384 SHA2-512
TLS1.3-KDF (CVL)	KAS-135KDF	TLS 1.3 key derivation, used to derive the TLS Session Key (AES key or AES-GCM key) and TLS Authentication Key (HMAC key).	Publication:SP 800-135 Rev. 1 Caveat:No part of the TLS v1.3 protocol, other than the KDF, has been tested by the CAVP and CMVP.	TLS v1.3 KDF HMAC-SHA2-256 HMAC-SHA2-384 SHA2-256 SHA2-384
DSA for Domain Parameter Generation	AsymKeyPair-DomPar	DSA domain parameter generation	Publication:Publication: FIPS 186-4	DSA PQGVer (FIPS186-4)
DSA for Key Generation	AsymKeyPair-KeyGen	DSA key generation	Publication:FIPS 186-4	DSA KeyGen (FIPS186-4) Counter DRBG CKG1 Key Type: Asymmetric
DSA for Digital Signature Generation	DigSig-SigGen	DSA digital signature generation Publication: FIPS 186-4.	Publication:FIPS 186-4	DSA SigGen (FIPS186-4)
DSA for Domain Parameter Verification (legacy)	AsymKeyPair-DomPar	DSA domain parameter verification	Publication:FIPS 186-4 Publication:FIPS 140-3 IG C.M	DSA PQGVer (FIPS186-4)
AES for Unauthenticated Key Unwrap (allowed) (legacy)	KTS-Wrap	AES key unwrap using the AES key (with any Approved unauthenticated mode)	Publication:FIPS PUB 197 Publication:SP 800-38C Publication:FIPS 140-3 IG C.M Publication:FIPS 140-3 IG D.G Key Strength: Key establishment methodology provides between 128 and 256 bits of encryption strength.	AES-CBC AES-CFB1 AES-CFB8 AES-CFB128 AES-CTR AES-ECB AES-OFB

Name	Type	Description	Properties	Algorithms
TDES+MAC for Key Unwrap (legacy)	KTS-Wrap	TDES-CMAC key unwrap using the TDES-CMAC key TDES+HMAC key unwrap using the TDES key and HMAC key	Publication :SP 800-67 Rev. 2 Publication:SP 800-38A Publication:SP 800-38B Publication:FIPS PUB 198-1 Publication:FIPS 140-3 IG C.M Publication:FIPS 140-3 IG D.G Key Strength:Key establishment methodology provides 112 or 168 bits of encryption strength.	TDES-CBC TDES-CFB1 TDES-CFB8 TDES-CMAC TDES-ECB TDES-CFB64 TDES-OFB HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512 HMAC-SHA3-224 HMAC-SHA3-256 HMAC-SHA3-384 HMAC-SHA3-512
TDES for Unauthenticated Key Unwrap (allowed) (legacy)	KTS-Wrap	TDES key unwrap using the TDES key (with any Approved unauthenticated mode)	Publication:SP 800-67 Rev. 2 Publication:SP 800-38A Publication:FIPS 140-3 IG C.M Publication:FIPS 140-3 IG D.G Key Strength:Key establishment methodology provides 112 or 168 bits of encryption strength.	TDES-CBC TDES-CFB1 TDES-CFB64 TDES-CFB8 TDES-ECB TDES-OFB
AES+MAC for Key Wrap/Unwrap	KTS-Wrap	AES-CMAC key wrap and unwrap using the AES-CMAC key AES-GMAC key wrap and unwrap using the AES-GMAC key AES+HMAC key wrap and unwrap using the AES key and HMAC key	Publication:FIPS PUB 197 Publication:SP 800-38A Publication:SP 800-38B Publication:SP 800-38D Publication:FIPS PUB 198-1 Publication:FIPS PUB 180-4 Publication:FIPS PUB 202 Publication:FIPS 140-3 IG D.G Key Strength:Key establishment methodology provides between 128 and 256 bits of encryption strength.	AES-CBC AES-CFB1 AES-CFB128 AES-CFB8 AES-CMAC AES-CTR AES-ECB AES-GMAC AES-OFB HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512 HMAC-SHA2-512 HMAC-SHA3-224 HMAC-SHA3-256 HMAC-SHA3-384 HMAC-SHA3-512
ECDSA for Key Verification (legacy)	AsymKeyPair-KeyVer	ECDSA key verification using the ECDSA public key (with curves B-163, K-163, and P-192)	Publication:FIPS 186-4 Publication:FIPS 140-3 IG C.M	ECDSA KeyVer (FIPS186-4)
ECDSA for Digital Signature Verification (legacy)	DigSig-SigVer	ECDSA digital signature verification using the ECDSA public key (with curves B-163, K-163, and P-192)	Publication:FIPS 186-4 Publication:FIPS 180-4 Publication:FIPS 140-3 IG C.M	ECDSA SigVer (FIPS186-4) SHA2-224 SHA2-256 SHA2-384 SHA2-512

Name	Type	Description	Properties	Algorithms
RSA for Signature Verification (legacy)	DigSig-SigVer	RSA signature verification using the RSA public key (with SHA-1 and/or a 1024-bit modulo)	Publication:FIPS 186-4 Publication:FIPS 180-4 Publication:FIPS 140-3 IG C.M	RSA SigVer (FIPS186-4) SHA-1 SHA2-224 SHA2-256 SHA2-384 SHA2-512

Table 9: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 AES-GCM

The module supports internal IV generation using its Approved DRBG. The IV is at least 96 bits in length per section 8.2.2 of *NIST SP 800-38D*, and the Approved DRBG generates outputs such that the (key, IV) pair collision probability is less than 2^{-32} per section 8 of *NIST SP 800-38D*.

The module also supports AES GCM encryption used in the context of the TLS protocol versions 1.2 and 1.3. 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:

- For TLS v1.2, the module supports acceptable AES GCM cipher suites from section 3.3.1.1 of *NIST SP 800-52rev2*.

  The mechanism for IV generation falls into scenario 1 in *FIPS 140-3 IG* C.H and is compliant with *RFC 5288*. The 64-bit counter portion of the IV is strictly increasing. The module explicitly ensures that the counter does not exhaust the maximum number of possible values of 2^{64}-1 for a given session key. If this exhaustion condition is observed, the module will return an error indication to the calling application, which will then need to either abort the connection, or trigger a handshake to establish a new encryption key. It is the responsibility of the module operator (i.e., the first party, client, or server) to trigger this handshake when this condition is encountered.

- For TLS v1.3, the module supports acceptable AES GCM cipher suites from section 3.3.1.2 of *NIST SP 800-52rev2*. The protocol's implementation is contained within the boundary of the module, and the generated IV is only used in the context of the AES GCM encryption executing the provisions of the TLS 1.3 protocol.

  The mechanism for IV generation falls into scenario 5 in *FIPS 140-3 IG* C.H and is compliant with *RFC 8446*. Each session employs a "per-record nonce", a 64-bit sequence number (or IV) maintained separately for reading and writing records. Each sequence number is set to 0 at the beginning of a connection and whenever the key is changed (the first record transmitted under a particular traffic key uses sequence number 0), and the appropriate sequence number is incremented by one after reading or writing each record. Because the size of sequence numbers is 64 bits, they should not wrap. If a sequence number needs to wrap, it is the responsibility of the module operator to either re-key with a new key for AES-GCM or terminate the connection.

In case the module's power is lost and then restored, the calling application is responsible for ensuring that a new key for use with the AES-GCM encryption/decryption shall be established. This condition is not enforced by the

module but is met implicitly. The module does not retain any state across resets or power-cycles, and AES-GCM key/IVs are not stored in non-volatile persistent memory (i.e., disk). Hence, no reconnection can occur without a fresh key establishment operation and the associated SSPs.

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

2.7.2 AES-XTS

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

In compliance with *FIPS 140-3 IG* C.I, the module implements a check to ensure that the two AES keys used in the XTS-AES algorithm are not identical.

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

2.7.3 PBKDF2

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

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

As specified in *NIST SP 800-132*, keys derived from passwords/passphrases may only be used in storage applications.

2.8 RNG and Entropy

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

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

2.9 Key Generation

The cryptographic module uses its counter-based 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 utilized 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 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-KW wrap/unwrap
- AES-KWP wrap/unwrap
- AES unwrap (legacy)
- TDES unwrap (legacy)
- TDES + MAC unwrap (legacy)

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

2.11 Industry Protocols

The module supports the following industry protocols in the Approved mode of operation:

- TLS 1.2 (per *RFC 7627*)
- TLS 1.3

The KDFs associated with these protocols shall only be used within the context of their respective protocols. No parts of these protocols, other than the Approved cryptographic algorithms and the KDFs, have been tested by the CAVP and CMVP.

2.12 Additional Information

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

3. Cryptographic Module Interfaces

3.1 Ports and Interfaces

The module supports the following four 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, as it communicates only to the calling application via its well-defined API. A mapping of the FIPS-defined interface to the module's physical ports and logical interfaces can be found in the table below. Note that the module does not output control information, and this 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 10: Ports and Interfaces

Data output via the data output interface is inhibited when the module is performing pre-operational and conditional tests, zeroization, or when the module is in the error state.

4. Roles, Services, and Authentication

4.1 Authentication Methods

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

N/A for this module.

4.2 Roles

The table below lists the supported roles.

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

Table 11: Roles

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

4.3 Approved Services

This module is a software library that provides cryptographic functionality to calling applications. As such, the security functions provided by the module are considered the module's security services. Indicators for Approved services (in the case of this module, those security functions with algorithm validation certificates and all required self-tests) are provided via API return value.

When invoking a security function, the calling application provides inputs via an internal structure, or "context". Upon each service invocation, the module will determine if the invoked security function is an Approved service. To access the resulting value, the calling application must pass the finalized context to the indicator API associated with that security function (note the indicator check must be performed by the calling application before any context cleanup is performed). The indicator API will return "1" to indicate the usage of an Approved service. Indicators for services providing non-Approved security functions (as well as for services not requiring an indicator) will have a value other than "1", ensuring that the indicators for Approved services are unambiguous. Additional details on the APIs used for the Approved service indicators are provided in Appendix B 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.

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

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show Status	Return FIPS mode status	API call parameters	Current operational status	N/A	None	Crypto Officer
Perform self-tests on-demand	Perform pre-operational self-tests	Indicator API return value = 1	Indicator API return value = 1	Status	None	Crypto Officer

Zeroize	Zeroize and de-allocate memory containing sensitive data	N/A	Restart calling application; reboot or power-cycle host platform	None	None	Crypto Officer - AES key: Z - AES CCM key : Z - AES GCM key : Z - AES XTS key : Z - AES CMAC key : Z - AES GMAC key : Z - Triple-DES key : Z - Triple-DES CMAC key : Z - HMAC key : Z - DSA public key : Z - ECDSA private key : Z - ECDSA public key : Z - RSA private key : Z - RSA public key : Z - DH private component : Z - DH public component : Z - ECDH private component : Z - ECDH public component : Z - Passphrase: Z - AES GCM IV: Z - TLS pre-master secret: Z - TLS master secret: Z - DRBG entropy input: Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- DRBG seed: Z - DRBG 'Key' value: Z - DSA private key : Z
Perform symmetric encryption	Encrypt plaintext data	Indicator API return value = 1	API call parameters, key, plaintext	Status, ciphertext	AES for Symmetric Encryption/Decryption AES-XTS for Symmetric Encryption/Decryption	User - AES key: W,E - AES XTS key : W,E
Perform symmetric decryption	Decrypt ciphertext data	Indicator API return value = 1	API call parameters, key, ciphertext	Status, plaintext	AES for Symmetric Encryption/Decryption AES-XTS for Symmetric Encryption/Decryption TDES for Symmetric Decryption (legacy)	User - AES key: W,E - AES XTS key : W,E - Triple-DES key : W,E
Generate symmetric digest	Generate symmetric digest	Indicator API return value = 1	API call parameters, key, plaintext	Status, digest	AES-CMAC for MAC Generation/Verification TDES for MAC Verification (legacy)	User - AES CMAC key : W,E - AES GMAC key : W,E
Verify symmetric digest	Verify symmetric digest	Indicator API return value = 1	API call parameters, digest	API call parameters, digest	AES-CMAC for MAC Generation/Verification AES-GMAC for MAC Generation/Verification	User - AES GCM key : W,E - AES GCM IV: W,E - Triple-DES CMAC key : W,E
Perform authenticated encryption	Encrypt plaintext using supplied AES GCM key and IV	Encrypt plaintext using supplied AES GCM key and IV	API call parameters, key, plaintext	Status, ciphertext, tab	AES-CCM for Authenticated Symmetric Encryption/Decryption AES-GCM for Authenticated Symmetric Encryption/Decryption	User - AES GCM key : W,E - AES GCM IV: W,E - AES CCM key : W,E
Perform authenticated decryption	Decrypt ciphertext using supplied AES GCM key and IV	Indicator API return value = 1	Indicator API return value = 1	Status, plaintext	AES-CCM for Authenticated Symmetric Encryption/Decryption AES-GCM for Authenticated Symmetric Encryption/Decryption	User - AES CCM key : W,E - AES GCM key : W,E - AES GCM IV: W,E
Generate random number	Return random bits to the calling application	Indicator API return value = 1	API call parameters, entropy DRBG state values	Status, random number	DRBG	User - DRBG entropy input: G,E - DRBG seed: G,E - DRBG 'V' value: G,E - DRBG 'Key' value: G,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Perform keyed hash operations	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
Generate message digest	Generate a message digest	Indicator API return value = 1	Indicator API return value = 1	Status, digest	SHA/SHAKE for Message Digest	User
Generate asymmetric key pair	Generate a public/private key pair	Indicator API return value = 1	Indicator API return value = 1	Status, key pair	ECDSA for Key Generation RSA for Key Generation DSA for Domain Parameter Generation DSA for Key Generation	User - ECDSA private key : G - ECDSA public key : G - RSA private key : G - RSA public key : G
Verify ECDSA public key	Verify an ECDSA public key	Indicator API return value = 1	API call parameters, key	Status	ECDSA for Key Verification ECDSA for Key Verification (legacy)	User - ECDSA public key : W
Generate digital signature	Generate a digital signature	Indicator API return value = 1	API call parameters, key, message	Status, signature	ECDSA for Digital Signature Generation RSA for Signature Generation DSA for Digital Signature Generation	User - ECDSA private key : W,E - RSA private key : W,E
Verify digital signature	Verify a digital signature	Indicator API return value = 1	API call parameters, key, signature, message	Status	DSA for Digital Signature Verification (legacy) ECDSA for Digital Signature Verification RSA for Signature Verification ECDSA for Digital Signature Verification (legacy) RSA for Signature Verification (legacy)	User - DSA public key : W,E - ECDSA public key : W,E - RSA public key : W,E
Perform key wrap	Perform key wrap	Indicator API return value = 1	API call parameters, encryption key, key	Status, encrypted key	KTS-AES+MAC KTS-AES-CCM KTS-AES-GCM KTS-AES-KW AES+MAC for Key Wrap/Unwrap	User - AES key: W,E - AES CCM key : W,E - AES CMAC key : W,E - AES GMAC key : W,E - AES GCM key : W,E - AES GCM IV: W,E - HMAC key : W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Perform key unwrap	Perform key unwrap	Indicator API return value = 1	API call parameters, decryption key, key	Status, decrypted key	KTS-AES+MAC KTS-AES-CCM KTS-AES-GCM KTS-AES-KW AES for Unauthenticated Key Unwrap (allowed) (legacy) TDES+MAC for Key Unwrap (legacy) TDES for Unauthenticated Key Unwrap (allowed) (legacy) AES+MAC for Key Wrap/Unwrap	User - AES key: W,E - AES CCM key : W,E - AES CMAC key : W,E - AES GMAC key : W,E - AES GCM key : W,E - AES GCM IV: W,E - HMAC key : W,E - Triple-DES key : W,E
Compute shared secret	Perform key unwrap Compute DH/ECDH shared secret suitable for use as input to a TLS KDF	Perform key unwrap Compute DH/ECDH shared secret suitable for use as input to a TLS KDF	API call parameters	API call parameters	DSA KeyGen for DH ECDSA KeyGen for ECDH ECDH Shared Secret Computation DH Shared Secret Computation	User - DH public component : W,E - DH private component : W,E - ECDH private component : W,E - ECDH public component : W,E - TLS pre-master secret: G,E
Derive TLS keys	Derive TLS session and integrity keys	Indicator API return value = 1	API call parameters, TLS pre-master secret	Status, TLS keys	TLS1.2-KDF (CVL) TLS1.3-KDF (CVL)	User - AES key: G,R - AES GCM key : G,R - AES GCM IV: G,R - HMAC key : G,R - TLS pre-master secret: W,E - TLS master secret: G,E
Derive key via PBKDF2	Derive key via PBKDF2	Indicator API return value = 1	API call parameters, password	Status, key	PBKDF	User - Passphrase: W,E - AES key: G,R - Triple-DES key : G,R

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show versioning information	Return module versioning information	N/A	API call parameters	Module name, version	None	Crypto Officer
Generate DSA domain parameters	Generate DSA domain parameters	Indicator API return value = 1	API call parameters	Status, domain parameters	DSA for Domain Parameter Generation	User
Verify DSA domain parameters	Verify DSA domain parameters	Indicator API return value = 1	API call parameters	Status	DSA for Domain Parameter Verification (legacy)	User

Table 12: Approved Services

4.4 Non-Approved Services

The table below lists the non-Approved services available to module operators.

Name	Description	Algorithms	Role
Perform data encryption (non-compliant)	Perform symmetric data encryption	ARIA Blake2 Blowfish Camellia CAST, CAST5 ChaCha20 DES IDEA RC2, RC4, RC5 SEED SM4 Triple-DES (non-compliant)	User
Perform data decryption (non-compliant)	Perform symmetric data decryption	ARIA Blake2 Blowfish Camellia CAST, CAST5 ChaCha20 DES IDEA RC2, RC4, RC5 SEED SM4	User
Perform MAC operations (non-compliant)	Perform message authentication operations	Poly1305 Triple-DES (non-compliant)	User
Perform hash operation (non-compliant)	Perform hash operation	MD2, MD4, MD5 RIPEMD RMD160 SHA-1 (non-compliant) SM2, SM3 Whirlpool	User
Perform digital signature functions (non-compliant)	Perform digital signature functions	DSA (non-compliant) ECDSA (non-compliant) EdDSA RSA (non-compliant)	User
Perform key agreement functions (non-compliant)	Perform key agreement functions	DH (non-compliant) ECDH (non-compliant)	User
Perform key wrap (non-compliant)	Perform key wrap functions	Triple-DES (non-compliant)	User

Name	Description	Algorithms	Role
Perform key encapsulation function (non-compliant)	Perform key encapsulation function	RSA (non-compliant)	User
Perform key un-encapsulation function (non-compliant)	Perform key un-encapsulation function	RSA (non-compliant)	User
Perform key derivation functions (non-compliant)	Perform key derivation functions	HKDF TLS 1.2 KDF (non-compliant)	User
Perform authenticated encryption/decryption	Perform authenticated encryption/decryption	AES-GCM (non-compliant) AES-OCB	User
Perform random number generation	Perform random number generation	ANSI X9.31 RNG (non-compliant) DRBG (non-compliant)	User
Perform key pair generation	Perform key pair generation	DSA (non-compliant) DSA, ECDSA, and RSA (non-compliant) ECDSA (non-compliant) EdDSA RSA (non-compliant)	User

Table 13: Non-Approved Services

4.5 External Software/Firmware Loaded

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

5. Software/Firmware Security

5.1 Integrity Techniques

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

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

5.2 Initiate on Demand

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

6. Operational Environment

6.1 Operational Environment Type and Requirements

The module is a software cryptographic library that executes in a Non-Modifiable operational environment.

The cryptographic module has control over its own SSPs. The process and memory management functionality of the host platform's OS prevents unauthorized access to plaintext private and secret keys, intermediate key generation values and other SSPs by external processes during module execution. The module only allows access to SSPs through its well-defined API. The operational 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

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

8. Non-Invasive Security

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

9. Sensitive Security Parameters Management

9.1 Storage Areas

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

Storage Area Name	Description	Persistence Type
RAM	SSPs are stored in the RAM	Dynamic

Table 14: Storage Areas

The module stores DRBG state values for the lifetime of the DRBG instance. The module uses SSPs passed in on the stack by the calling application and does not store these SSPs beyond the lifetime of the API call.

9.2 SSP Input-Output Methods

The table below lists input and output methods for the module's SSPs. Section 9.4 below selects from the input and output methods listed and specifies the appropriate method(s) in the "Inputs - Outputs" column applicable to each SSP.

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
[Input] External to RAM via Plaintext	External	RAM	Plaintext	Automated	Electronic	
[Output] RAM to External via Plaintext	RAM	External	Plaintext	Automated	Electronic	

Table 15: SSP Input-Output Methods

9.3 SSP Zeroization Methods

The table below lists SSP zeroization methods available to module operators. Section 9.4 below selects from the zeroization methods listed and specifies the appropriate method(s) in the "Zeroization" column applicable to each SSP.

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

Zeroization Method	Description	Rationale	Operator Initiation
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 16: SSP Zeroization Methods

At the end of each applicable function call, temporary SSPs in memory are automatically overwritten with zeroes and the space is deallocated. Maintenance of any keys and CSPs that exist outside the module's cryptographic boundary, including protection and zeroization, are the responsibility of the module operator.

9.4 SSPs

The module supports the keys and other SSPs listed in the table below. All SSP imports and exports are electronic and performed within the TOEPP.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES key	Symmetric encryption, decryption	Between 128 and 256 bits - Between 128 and 256 bits	Symmetric Key - CSP	PBKDF TLS1.2-KDF (CVL) TLS1.3-KDF (CVL)		AES for Symmetric Encryption/Decryption KTS-AES+MAC
AES CCM key	Authenticated symmetric encryption, decryption	Between 128 and 256 bits - Between 128 and 256 bits	Symmetric Key - CSP	TLS1.2-KDF (CVL) TLS1.3-KDF (CVL)		AES-CCM for Authenticated Symmetric Encryption/Decryption KTS-AES-CCM
AES GCM key	Authenticated symmetric encryption, decryption	Between 128 and 256 bits - Between 128 and 256 bits	Symmetric Key - CSP	TLS1.2-KDF (CVL) TLS1.3-KDF (CVL)		AES-GCM for Authenticated Symmetric Encryption/Decryption
AES XTS key	Symmetric encryption, decryption	128 or 256 bits - 128 or 256 bits	Symmetric Key - CSP			AES-XTS for Symmetric Encryption/Decryption
AES CMAC key	MAC generation, verification	Between 128 and 256 bits - Between 128 and 256 bits	MAC - CSP			AES-CMAC for MAC Generation/Verification KTS-AES+MAC
AES GMAC key	MAC generation, verification	Between 128 and 256 bits - Between 128 and 256 bits	MAC - CSP			AES-GMAC for MAC Generation/Verification KTS-AES+MAC
Triple-DES key	Triple-DES key	N/A - N/A	Symmetric Key - CSP	PBKDF		TDES for Symmetric Decryption (legacy)
Triple-DES CMAC key	MAC Verification	N/A - N/A	MAC - CSP			TDES for MAC Verification (legacy)
HMAC key	Keyed Hash	112 bits (minimum) - 112 bits (minimum)	MAC - CSP	TLS1.2-KDF (CVL) TLS1.3-KDF (CVL)		KTS-AES+MAC HMAC for Message Authentication
DSA private key	Digital signature generation	2048 or 3072 bits - 112 or 128 bits	Private - CSP	DSA for Key Generation		DSA for Digital Signature Generation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DSA public key	Digital signature verification	Between 2048 and 3072 bits - 112 or 128 bits	Public - PSP			DSA for Digital Signature Verification (legacy)
ECDSA private key	Digital signature generation	Between 224 and 521 bits - Between 112 and 256 bits	Private - CSP	ECDSA for Key Generation		ECDSA for Digital Signature Generation
ECDSA public key	Digital signature verification	Between 224 and 521 bits - Between 112 and 256 bits	Public - PSP	ECDSA for Key Verification		ECDSA for Digital Signature Verification
RSA private key	Digital signature generation	Between 2048 and 4096 bits - Between 112 and 150 bits	Private - CSP	RSA for Key Generation		RSA for Signature Generation
RSA public key	Signature verification	Between 2048 and 4096 bits - Between 80 and 150 bits	Public - PSP	RSA for Signature Generation		RSA for Signature Verification RSA for Signature Verification (legacy)
DH private component	DH shared secret computation	2048 bits - 112 bits	Private - CSP	DSA KeyGen for DH		KAS-FFC-SSC Sp800-56Ar3 (A5835)
DH public component	DH shared secret computation	2048 bits - 112 bits	Public - PSP	DSA KeyGen for DH		KAS-FFC-SSC Sp800-56Ar3 (A5835)
ECDH private component	ECDH private component	Between 224 and 521 bits - Between 112 and 256 bits	Private - CSP	ECDSA KeyGen for ECDH		KAS-ECC-SSC Sp800-56Ar3 (A5835)
ECDH public component	ECDH shared secret computation	Between 224 and 521 bits - Between 112 and 256 bits	Public - PSP	ECDSA KeyGen for ECDH		KAS-ECC-SSC Sp800-56Ar3 (A5835)
Passphrase	Input to PBKDF for key derivation	N/a - N/A	Passphrase - CSP			PBKDF
AES GCM IV	Derivation of the TLS master secret	96 bits - N/A	Initialization Vector - CSP	DRBG		AES-GCM for Authenticated Symmetric Encryption/Decryption
TLS pre-master secret	Derivation of the TLS master secret	384 bits - N/A	Pre-master Secret - CSP			TLS1.2-KDF (CVL) TLS1.3-KDF (CVL)
TLS master secret	Derivation of the AES/AES-GCM key and HMAC key used for securing TLS connections	384 bits - N/A	Master Secret - CSP			TLS1.2-KDF (CVL) TLS1.3-KDF (CVL)
DRBG entropy input	Entropy material for DRBG	Between 128 and 512 bits - N/A	Entropy input - CSP			DRBG
DRBG seed	Seeding material for DRBG	Between 256 and 384 bits - N/A	Seed - CSP	DRBG		DRBG
DRBG 'V' value	State value for DRBG	128 bits - N/A	State Value - CSP	DRBG		DRBG
DRBG 'Key' value	State value for DRBG	Between 128 and 256 bits - N/A	State Value - CSP	DRBG		DRBG

Table 17: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES key	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	TLS master secret:Derived From Passphrase:Derived From
AES CCM key	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed	Remove Power Reboot Power-cycle	TLS master secret:Derived From
AES GCM key	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	TLS master secret:Derived From AES GCM IV:Paired With
AES XTS key	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	
AES CMAC key	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	
AES GMAC key	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	
Triple-DES key		RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	
Triple-DES CMAC key	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	
HMAC key	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	TLS master secret:Derived From
DSA private key	[Output] RAM to External via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	DSA public key :Paired With
DSA public key	[Output] RAM to External via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	
ECDSA private key	[Output] RAM to External via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	ECDSA public key :Paired With
ECDSA public key	[Output] RAM to External via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	ECDSA private key :Paired With
RSA private key	[Output] RAM to External via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	RSA public key :Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
RSA public key	[Output] RAM to External via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	RSA private key :Paired With
DH private component	[Output] RAM to External via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	DH public component :Paired With
DH public component	[Output] RAM to External via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	DH private component :Paired With
ECDH private component	[Output] RAM to External via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	ECDH public component :Paired With
ECDH public component	[Output] RAM to External via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	ECDH private component :Paired With
Passphrase	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	AES key:Derived From
AES GCM IV	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	AES GCM key :Used With
TLS pre-master secret	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	DH private component :Derived From DH public component :Derived From ECDH private component :Derived From ECDH public component :Derived From
TLS master secret	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	TLS pre-master secret:Derived From
DRBG entropy input	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	DRBG seed:Derived From DRBG ‘V’ value:Derived From DRBG ‘Key’ value:Derived From
DRBG seed	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	DRBG entropy input:Derived From
DRBG ‘V’ value	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	DRBG seed:Derived From DRBG ‘Key’ value:Used With
DRBG ‘Key’ value	[Input] External to RAM via Plaintext	RAM:Plaintext	Until the module is unloaded or the power is removed.	Remove Power Reboot Power-cycle	DRBG seed:Derived From DRBG ‘V’ value:Used With

Table 18: SSP Table 2

10. Self-Tests

10.1 Pre-Operational Self-Tests

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

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A5835)	SHA2-256	Software Integrity Test	SW/FW Integrity	Returned success or error code	Test for libcrypto. Performed automatically without operator action
HMAC-SHA2-256 (A5835)	SHA2-256	Software Integrity Test	SW/FW Integrity	Returned success or error code	Test for libssl. Performed automatically without operator action

Table 19: 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 (A5835)	128 bit	KAT	CAST	returned success or error code	encrypt	After successful software integrity test
AES-ECB (A5835)	128 bit	KAT	CAST	returned success or error code	decrypt	After successful software integrity test
AES-CCM (A5835)	192 bit	KAT	CAST	returned success or error code	encrypt	After successful software integrity test
AES-CCM (A5835)	192 bit	KAT	CAST	returned success or error code	decrypt	After successful software integrity test
AES-GCM (A5835)	128 bit	KAT	CAST	returned success or error code	encrypt	After successful software integrity test
AES-GCM (A5835)	128 bit	KAT	CAST	returned success or error code	decrypt	After successful software integrity test
AES-XTS Testing Revision 2.0 (A5835)	128 bit	KAT	CAST	returned success or error code	encrypt	After successful software integrity test
AES-XTS Testing Revision 2.0 (A5835)	128 bit	KAT	CAST	returned success or error code	decrypt	After successful software integrity test
AES-CMAC (A5835)	CBC mode, 128-bit; 192-bit; 256-bit	KAT	CAST	returned success or error code	Generate	After successful software integrity test
AES-CMAC (A5835)	CBC mode, 128-bit; 192-bit; 256-bit	KAT	CAST	returned success or error code	Verify	After successful software integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
TDES-ECB (A5835)	3Key	KAT	CAST	returned success or error code	Encrypt	After successful software integrity test
TDES-ECB (A5835)	3Key	KAT	CAST	returned success or error code	Decrypt	After successful software integrity test
TDES-CMAC (A5835)	CBC mode, 3Key	KAT	CAST	returned success or error code	Generate	After successful software integrity test
TDES-CMAC (A5835)	CBC mode, 3Key	KAT	CAST	returned success or error code	Verify	After successful software integrity test
Counter DRBG (A5835)	AES, 256-bit	KAT	CAST	returned success or error code	Generate/Instantiate/Reseed	After successful software integrity test
DSA SigGen (FIPS186-4) (A5835)	2048-bit; SHA2-256	KAT	CAST	returned success or error code	Sign	After successful software integrity test
DSA SigVer (FIPS186-4) (A5835)	2048-bit; SHA2-256	KAT	CAST	returned success or error code	Verify	After successful software integrity test
ECDSA SigVer (FIPS186-4) (A5835)	P-224; K-233; SHA-256	KAT	CAST	returned success or error code	Verify	After successful software integrity test
RSA SigGen (FIPS186-4) (A5835)	2048-bit; SHA2-256; PKCS#1.5 scheme	KAT	CAST	returned success or error code	Sign	After successful software integrity test
RSA SigVer (FIPS186-4) (A5835)	2048-bit; SHA2-256; PKCS#1.5 scheme	KAT	CAST	returned success or error code	Verify	After successful software integrity test
HMAC-SHA-1 (A5835)	SHA-1	KAT	CAST	returned success or error code	Hashed Message	After successful software integrity test
HMAC-SHA2-224 (A5835)	SHA2-224	KAT	CAST	returned success or error code	Hashed Message	After successful software integrity test
HMAC-SHA2-256 (A5835)	SHA2-256	KAT	CAST	returned success or error code	Hashed Message	Prior to the software integrity test
HMAC-SHA2-384 (A5835)	SHA2-384	KAT	CAST	returned success or error code	Hashed Message	After successful software integrity test
HMAC-SHA2-512 (A5835)	SHA2-512	KAT	CAST	returned success or error code	Hashed Message	After successful software integrity test
HMAC-SHA3-224 (A5835)	SHA3-224	KAT	CAST	returned success or error code	Hashed Message	After successful software integrity test
HMAC-SHA3-256 (A5835)	SHA3-256	KAT	CAST	returned success or error code	Hashed Message	Prior to the software integrity test
HMAC-SHA3-384 (A5835)	SHA3-384	KAT	CAST	returned success or error code	Hashed Message	After successful software integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA3-512 (A5835)	SHA3-512	KAT	CAST	returned success or error code	Hashed Message	After successful software integrity test
SHA-1 (A5835)	-	KAT	CAST	returned success or error code	Hash	After successful software integrity test
SHA2-224 (A5835)	-	KAT	CAST	returned success or error code	Hash	After successful software integrity test
SHA2-256 (A5835)	-	KAT	CAST	returned success or error code	Hash	After successful software integrity test
SHA2-384 (A5835)	-	KAT	CAST	returned success or error code	Hash	After successful software integrity test
SHA2-512 (A5835)	-	KAT	CAST	returned success or error code	Hash	After successful software integrity test
SHA3-224 (A5835)	-	KAT	CAST	returned success or error code	Hash	After successful software integrity test
SHA3-256 (A5835)	-	KAT	CAST	returned success or error code	Hash	After successful software integrity test
SHA3-384 (A5835)	-	KAT	CAST	returned success or error code	Hash	After successful software integrity test
SHA3-512 (A5835)	-	KAT	CAST	returned success or error code	Hash	After successful software integrity test
KAS-FFC-SSC Sp800-56Ar3 (A5835)	2048-bit	KAT	CAST	returned success or error code	Shared Secret “Z” Computation	After successful software integrity test
KAS-ECC-SSC Sp800-56Ar3 (A5835)	P-224	KAT	CAST	returned success or error code	Shared Secret “Z” Computation	After successful software integrity test
PBKDF (A5835)	SHA2-224	KAT	CAST	returned success or error code	KDF	After successful software integrity test
TLS v1.2 KDF RFC7627 (A5835)	-	KAT	CAST	returned success or error code	KDF	After successful software integrity test
TLS v1.3 KDF (A5836)	-	KAT	CAST	returned success or error code	KDF	After successful software integrity test
DSA KeyGen (FIPS186-4) (A5835)	-	PCT	PCT	returned success or error code	Sign/Verify	When an ECDSA key pair is generated for use with sign/verify functions
ECDSA KeyGen (FIPS186-4) (A5835)	-	PCT	PCT	returned success or error code	Sign/Verify	When an ECDSA key pair is generated for use with sign/verify functions
RSA KeyGen (FIPS186-4) (A5835)	-	PCT	PCT	returned success or error code	Sign/Verify	When an RSA key pair is generated for use with sign/verify functions

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
DH	-	PCT	PCT	returned success or error code	Key Agreement	When a DSA key pair is generated for use with DH key transport functions
ECDH	-	PCT	PCT	returned success or error code	Key Agreement	When an ECDSA key pair is generated for use with ECDH key transport functions
ECDSA SigGen (FIPS186-4) (A5835)	P-224; K-233; SHA-256	KAT	CAST	returned success or error code	Sign	After successful software integrity test

Table 20: Conditional Self-Tests

10.3 Periodic Self-Test Information

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

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A5835)	Software Integrity Test	SW/FW Integrity	On Demand	Manually
HMAC-SHA2-256 (A5835)	Software Integrity Test	SW/FW Integrity	On Demand	Manually

Table 21: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC (A5835)	KAT	CAST	On Demand	Manually
AES-ECB (A5835)	KAT	CAST	On Demand	Manually
AES-CCM (A5835)	KAT	CAST	On Demand	Manually
AES-CCM (A5835)	KAT	CAST	On Demand	Manually
AES-GCM (A5835)	KAT	CAST	On Demand	Manually
AES-GCM (A5835)	KAT	CAST	On Demand	Manually
AES-XTS Testing Revision 2.0 (A5835)	KAT	CAST	On Demand	Manually
AES-XTS Testing Revision 2.0 (A5835)	KAT	CAST	On Demand	Manually
AES-CMAC (A5835)	KAT	CAST	On Demand	Manually
AES-CMAC (A5835)	KAT	CAST	On Demand	Manually
TDES-ECB (A5835)	KAT	CAST	On Demand	Manually
TDES-ECB (A5835)	KAT	CAST	On Demand	Manually
TDES-CMAC (A5835)	KAT	CAST	On Demand	Manually
TDES-CMAC (A5835)	KAT	CAST	On Demand	Manually
Counter DRBG (A5835)	KAT	CAST	On Demand	Manually
DSA SigGen (FIPS186-4) (A5835)	KAT	CAST	On Demand	Manually
DSA SigVer (FIPS186-4) (A5835)	KAT	CAST	On Demand	Manually
ECDSA SigVer (FIPS186-4) (A5835)	KAT	CAST	On Demand	Manually
RSA SigGen (FIPS186-4) (A5835)	KAT	CAST	On Demand	Manually
RSA SigVer (FIPS186-4) (A5835)	KAT	CAST	On Demand	Manually
HMAC-SHA-1 (A5835)	KAT	CAST	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-224 (A5835)	KAT	CAST	On Demand	Manually
HMAC-SHA2-256 (A5835)	KAT	CAST	On Demand	Manually
HMAC-SHA2-384 (A5835)	KAT	CAST	On Demand	Manually
HMAC-SHA2-512 (A5835)	KAT	CAST	On Demand	Manually
HMAC-SHA3-224 (A5835)	KAT	CAST	On Demand	Manually
HMAC-SHA3-256 (A5835)	KAT	CAST	On Demand	Manually
HMAC-SHA3-384 (A5835)	KAT	CAST	On Demand	Manually
HMAC-SHA3-512 (A5835)	KAT	CAST	On Demand	Manually
SHA-1 (A5835)	KAT	CAST	On Demand	Manually
SHA2-224 (A5835)	KAT	CAST	On Demand	Manually
SHA2-256 (A5835)	KAT	CAST	On Demand	Manually
SHA2-384 (A5835)	KAT	CAST	On Demand	Manually
SHA2-512 (A5835)	KAT	CAST	On Demand	Manually
SHA3-224 (A5835)	KAT	CAST	On Demand	Manually
SHA3-256 (A5835)	KAT	CAST	On Demand	Manually
SHA3-384 (A5835)	KAT	CAST	On Demand	Manually
SHA3-512 (A5835)	KAT	CAST	On Demand	Manually
KAS-FFC-SSC Sp800-56Ar3 (A5835)	KAT	CAST	On Demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A5835)	KAT	CAST	On Demand	Manually
PBKDF (A5835)	KAT	CAST	On Demand	Manually
TLS v1.2 KDF RFC7627 (A5835)	KAT	CAST	On Demand	Manually
TLS v1.3 KDF (A5836)	KAT	CAST	On Demand	Manually
DSA KeyGen (FIPS186-4) (A5835)	PCT	PCT	On Demand	Manually
ECDSA KeyGen (FIPS186-4) (A5835)	PCT	PCT	On Demand	Manually
RSA KeyGen (FIPS186-4) (A5835)	PCT	PCT	On Demand	Manually
DH	PCT	PCT	On Demand	Manually
ECDH	PCT	PCT	On Demand	Manually
ECDSA SigGen (FIPS186-4) (A5835)	KAT	CAST	On Demand	Manually

Table 22: Conditional Periodic Information

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

10.4 Error States

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

Name	Description	Conditions	Recovery Method	Indicator
Critical Error	Module immediately terminates the calling application and sets an internal flag signaling the error condition. The module disables access to all cryptographic functions, SSPs, and data output services while the error condition persists.	Upon failure of any pre-operational or conditional self-test,	The module must be re-instantiated by the calling application. The CO should contact Riverbed Technology LLC if errors persist after re-instantiation.	Returns error code upon self-test failure; returns failure indicator for subsequent requests for cryptographic services.

Table 23: Error States

11. Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

The Riverbed Cryptographic Module is not delivered to end-users as a standalone offering. Rather, it is a pre-built integrated component of Riverbed's application software, and these applications are the sole consumers of the cryptographic services provided by the module.

The module and its calling application are delivered pre-installed on one of the Riverbed platforms specified in section 6 above or one where portability is maintained. Riverbed does not provide end-users with any mechanisms to directly access the module, its source code, its APIs, or any information sent to/from the module.

The module's integrity check is performed automatically at module instantiation (i.e., when the module is loaded into memory for execution) without action from the module operator, and end-users have no ability to bypass the automatic integrity check.

No setup steps are 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 Riverbed Customer Support should be contacted.

The following list provides additional guidance for the CO:

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

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

11.3 Non-Administrator Guidance

The following list provides additional policies for the User 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 186-5* (including those required of the intended signatory and the signature verifier) are outside the scope of the module and are the responsibility of the calling application.

- The calling application is responsible for ensuring that CSPs are not shared between Approved and non-Approved services and modes of operation.

- The calling application is responsible for using entropy sources that meet the minimum security strength of 112 bits required for the CTR_DRBG as shown in *NIST SP 800-90Arev1*, Table 3.

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

Table 24. Acronyms and Abbreviations

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

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

Appendix B. Approved Service Indicators

This appendix specifies the APIs that are externally accessible and return the Approved service indicators.

Synopsis

```
#include <openssl/service_indicator.h>
#include <openssl/ssl.h>

int EVP_cipher_get_service_indicator(EVP_CIPHER_CTX *ctx);
int DSA_get_service_indicator(DSA * ptr_dsa, DSA_MODES_t mode);
int RSA_key_get_service_indicator(RSA * ptr_rsa);
int PBKDF_get_service_indicator();
int EVP_Digest_get_service_indicator(EVP_MD_CTX *ctx);
int EC_key_get_service_indicator(EC_KEY *ec_key);
int CMAC_get_service_indicator(CMAC_CTX *cmac_ctx, CMAC_MODE_t mode);
int HMAC_get_service_indicator(HMAC_CTX *ctx);
int TLSKDF_get_service_indicator(EVP_PKEY_CTX *tls_ctx);
int TLS1_3_kdf_get_service_indicator(EVP_MD *md);
int TLS1_3_get_service_indicator(SSL *s);
int DRBG_get_service_indicator(RAND_DRBG *drbg);
```

Description

These APIs are high-level interfaces that return the Approved service indicator value based on the parameter(s) passed to them.

- EVP_cipher_get_service_indicator() is used to return the appropriate Approved service indicator status for block ciphers like AES and Triple DES.
- DSA_get_service_indicator() is used to return the appropriate Approved service indicator status for the DSA algorithm and its modes. You must include the mode you want the indicator for, which are specified in the DSA_MODES_t enum.
- RSA_key_get_service_indicator() is used to return the appropriate Approved service indicator status for RSA algorithm and its modes.
- PBKDF_get_service_indicator() is used to return the appropriate Approved service indicator status for PBKDF usage.
- EVP_Digest_get_service_indicator() is used to return the appropriate Approved service indicator status for SHS algorithms like SHA-1 and SHAKE.
- EC_key_get_service_indicator() is used to return the appropriate Approved service indicator status for elliptic curve algorithms like ECDSA and its modes.

- CMAC_get_service_indicator() is used to return the appropriate Approved service indicator status for CMAC requests that use AES or Triple DES. You must include the mode you want the indicator for, which are specified in the CMAC_MODE_t enum.

- HMAC_get_service_indicator() is used to return the appropriate Approved service indicator status for HMAC requests and the associated SHS algorithm.

- TLSKDF_get_service_indicator() is used to return the appropriate Approved service indicator status for TLS KDF usage excluding TLS 1.3.

- TLS1_3_kdf_get_service_indicator() is used to return the appropriate Approved service indicator status for TLS 1.3 KDF usage. This function requires the ssl.h file and is used to call the TLS1_3_get_service_indicator() function because of the SSL struct requirement. You cannot call TLS1_3_get_service_indicator() directly unless you have the SSL struct that was used.

- DRBG_get_service_indicator() is used to return the appropriate Approved service indicator status for DRBG usage.

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