Dell Inc., BSAFE Product Team

Dell BSAFE Crypto Module for Java

FIPS 140-3 Non-Proprietary Security Policy

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This is Dell Australia Pty Limited non-proprietary security policy for the Dell BSAFE Crypto Module for Java (hereinafter referred to as the as BSAFE Java Crypto Module, the module or JCM) with software version 7.0. The following details how this Module meets the security requirements of FIPS 140-3, SP 800-140, and ISO/IEC 19790 for a Security Level 1 software cryptographic module.

The security requirements cover areas related to the design and implementation of a cryptographic module. These areas include cryptographic module specification; cryptographic module interfaces; roles, services, and authentication; software/firmware security; operational environment; physical security; non-invasive security; sensitive security parameter management; self-tests; life-cycle assurance; and mitigation of other attacks. Table 1 below indicates the actual security levels for each area of the cryptographic module.

1.2 Security Levels

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	1
	Overall Level	1

Table 1: Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

BSAFE Java Crypto Module is a software module intended to be used as part of a software system, providing cryptographic services to that system.

The module is operated in a modifiable operational environment. It is provided as a Java Archive (jar) file and is intended to be distributed with, and used by, a Java application

The module consists of a jar file, jcmFIPS-7.0.jar. The name and version of the module can be accessed from the API `ModuleConfig.getVersionInfo()`.

Module Type: Software

Module Embodiment: Multi-Chip Standalone

Module Characteristics:

Cryptographic Boundary:

BSAFE Java Crypto Module is classified as a multi-chip standalone software cryptographic module for the purposes of FIPS 140-3. As such, it is tested on specific operating systems and computer platforms. The cryptographic boundary includes the module running on selected platforms running selected operating systems. The module is packaged as a jar file containing the Module's entire executable code. The module relies on the physical security provided by the host computer in which it runs. The module accepts Control Input through the API calls.

Tested Operational Environment's Physical Perimeter (TOEPP):

The following diagram depicts the cryptographic boundary and the physical perimeter defined as the tested platform's hard case enclosure around which everything runs.

The cryptographic boundary includes all of the software components of the cryptographic libraries.

The physical perimeter is the Tested Operational Environment's Physical Perimeter (TOEPP) on which the module runs. The module performs no communication other than with the calling application.

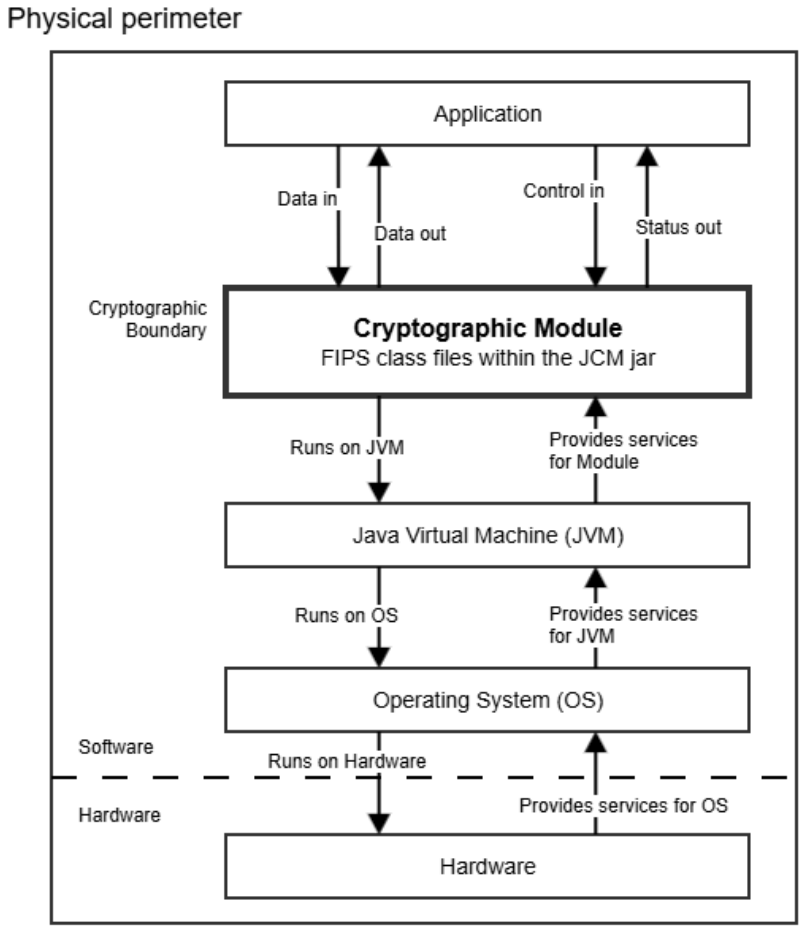

Figure 1: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Operational Environments - Software, Firmware, Hybrid:

For FIPS 140-3 validation, the module is tested by an accredited FIPS 140-3 testing laboratory on the following operational environments:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
SUSE Linux Enterprise Server 15 SP3 (64-bit) with OpenJDK 11	Dell PowerEdge R6525	AMD EPYC 7513	No		7.0
SUSE Linux Enterprise Server 15 SP3 (64-bit) with OpenJDK 8	Dell PowerEdge R6525	AMD EPYC 7513	No		7.0

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Windows Server 2019 (64-bit) with Oracle JRE 8	Dell PowerEdge R6525	AMD EPYC 7513	No		7.0
Windows Server 2016 (64-bit) with Oracle JRE 8	Dell PowerEdge T130	Intel Xeon CPU E3-1230	No		7.0

Table 2: Tested Operational Environments - Software, Firmware, Hybrid

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

Dell BSAFE affirms compliance for the following operational environments:

Operating System	Hardware Platform
Apple MacOS 10.15 (x86_64) with Oracle JDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Apple MacOS 10.15 (x86_64) with Oracle JDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Canonical Ubuntu 16.04 (x86) with OpenJDK JDK 8 (32-bit)	Generic Hardware Platform with Intel x86 (32-bit)
Canonical Ubuntu 16.04 (x86) with OpenJDK JDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
CentOS 7.9 (x86_64) with OpenJDK JDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Dell PowerProtect Data Domain OS (x86_64) with Oracle JDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Dell PowerStoreOS 4.0 (x86_64) with OpenJDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
FreeBSD Foundation 12 (x86_64) with OpenJDK JDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
HPE HP-UX 11.31 with HP JDK 8 (64-bit)	Generic Hardware Platform with Itanium 2
IBM AIX 7.2 with IBM JDK 8 (64-bit)	Generic Hardware Platform with PowerPC (64-bit)
Microsoft Windows 10 Enterprise (x86_64) with Oracle JDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Microsoft Windows 10 Enterprise (x86_64) with Oracle JDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Microsoft Windows 10 Enterprise (x86_64) with Oracle JDK 7 (32-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Microsoft Windows Server 2019 (x86_64) with Oracle JDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Microsoft Windows Server 2016 (x86_64) with Oracle JDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Oracle Solaris 11.4 with Oracle JDK 11 (64-bit)	Generic Hardware Platform with SPARC v9
Oracle Solaris 11.4 with Oracle JDK 8 (64-bit)	Generic Hardware Platform with SPARC v9

Operating System	Hardware Platform
Oracle Linux 7 64-bit on Oracle X Series Servers with Oracle JDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Oracle Linux 7 64-bit on Oracle X Series Servers with Oracle JDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Oracle Linux 7 64-bit on Oracle E Series Servers with Oracle JDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Oracle Linux 7 64-bit on Oracle E Series Servers with Oracle JDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Oracle Linux 7 64-bit on Oracle A Series Servers with Oracle JDK 8 (64-bit)	Generic Hardware Platform with ARMv8 (64-bit)
Oracle Linux 7 64-bit on Oracle A Series Servers with Oracle JDK 11 (64-bit)	Generic Hardware Platform with ARMv8 (64-bit)
Oracle Linux 8 64-bit on Oracle X Series Servers with Oracle JDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Oracle Linux 8 64-bit on Oracle X Series Servers with Oracle JDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Oracle Linux 8 64-bit on Oracle E Series Servers with Oracle JDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Oracle Linux 8 64-bit on Oracle E Series Servers with Oracle JDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Oracle Linux 8 64-bit on Oracle A Series Servers with Oracle JDK 8 (64-bit)	Generic Hardware Platform with ARMv8 (64-bit)
Oracle Linux 8 64-bit on Oracle A Series Servers with Oracle JDK 11 (64-bit)	Generic Hardware Platform with ARMv8 (64-bit)
Red Hat Enterprise Linux 8.6 (x86_64) with Oracle JDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Red Hat Enterprise Linux 8.6 (x86_64) with Oracle JDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Red Hat Enterprise Linux 7.9 (x86_64) with Oracle JDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Red Hat Enterprise Linux 7.9 (x86_64) with Oracle JDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
SUSE Linux Enterprise Server 15 SP4 (x86_64) with OpenJDK 17 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
SUSE Linux Enterprise Server 15 SP4 (x86_64) with OpenJDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
SUSE Linux Enterprise Server 15 SP4 (x86_64) with OpenJDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
SUSE Linux Enterprise Server 15 SP2 (x86_64) with OpenJDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
SUSE Linux Enterprise Server 15 SP2 (x86_64) with OpenJDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
SUSE Linux Enterprise Server 12 SP5 (x86_64) with IBM JDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
SUSE Linux Enterprise Server 12 SP5 (x86_64) with IBM JDK 7 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
SUSE Linux Enterprise Server 12 SP5 (x86_64) with OpenJDK 7 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)

Operating System	Hardware Platform
SUSE Linux Enterprise Server 12 SP5 (x86_64) with Oracle JDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
SUSE Linux Enterprise Server 12 SP5 (x86_64) with Oracle JDK 8 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
SUSE Linux Enterprise Server 12 SP5 (x86_64) with Oracle JDK 7 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Dell PowerStoreOS 4.1 (x86_64) with OpenJDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)
Dell PowerStoreOS 4.2 (x86_64) with OpenJDK 11 (64-bit)	Generic Hardware Platform with Intel x86_64 (64-bit)

Table 3: Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid

CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

2.3 Excluded Components

There are no components within the cryptographic boundary that are excluded from the module.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved Mode	Approved mode of operation is entered when the module utilizes the services that use the security functions listed in the Approved Algorithms Table and the Vendor Affirmed Algorithms Table.	Approved	API CryptoModule.isFips140Mode() returns true
Non-Approved mode	Non-Approved mode of operation is entered when the module utilizes non-approved security functions in the Table Non-Approved Algorithms Not Allowed in the Approved Mode of Operation.	Non-Approved	API CryptoModule.isFips140Mode() returns false

Table 4: Modes List and Description

The module supports both approved and non-approved modes of operation. It is operated in an approved mode after initial operations are performed, and all pre-operational self-tests have been completed successfully. The non-approved mode is entered when a non-approved algorithm or service is invoked. The Approved mode of operation can only be transitioned into the Non-Approved mode by calling one of the Non-Approved services. The module does not

claim implementation of a degraded mode of operation. Table 5 lists all the approved or vendor-affirmed security functions of the module, including specific key size(s) – in bits unless otherwise noted – employed for approved services and implemented modes of operation.

The mode of operation is identified by the following fields in the FIPS140Context interface

- FIPS140Context.MODE_FIPS140:
  Only Approved Algorithms can be used in this mode.

- FIPS140Context.MODE_NON_FIPS140:
  Both approved and non-approved algorithms can be used in this mode

The `ModuleLoader.load()` API loads the module for use in approved mode. This API returns the one and only instance of a ModuleConfig object.

The `ModuleConfig.newCryptoModule()` API creates CryptoModule objects. This API supports a FIPS140Context parameter which specifies the FIPS 140-3 mode of operation of the CryptoModule.

Refer to the API Javadoc for more information about these APIs.

An application using JCM must include the jar file, *jcmFIPS-7.0.jar*, in its Java classpath and call the `ModuleLoader.load()` API to load the module. This API runs the pre-operational self-tests and cryptographic algorithm self-tests automatically and if the self-tests complete successfully, the cryptographic services of the module can be used.

2.5 Algorithms

Approved Algorithms:

The following table lists the BSAFE Java Crypto Module Approved algorithms, with the appropriate standards and CAVP validation certificate numbers:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A2314	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC-CS1	A2314	Direction - decrypt, encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 128-65536 Increment 8	SP 800-38A
AES-CBC-CS2	A2314	Direction - decrypt, encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 128-65536 Increment 8	SP 800-38A
AES-CBC-CS3	A2314	Direction - decrypt, encrypt Key Length - 128, 192, 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
		Payload Length - Payload Length: 128-65536 Increment 8	
AES-CCM	A2314	Key Length - 128, 192, 256 Tag Length - 112, 128, 32, 48, 64, 80, 96 IV Length - IV Length: 56-104 Increment 8 Payload Length - Payload Length: 0-256 Increment 8 AAD Length - AAD Length: 0-524288 Increment 8	SP 800-38C
AES-CFB128	A2314	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A2314	Direction - Generation, Verification Key Length - 128, 192, 256 MAC Length - MAC Length: 32-128 Increment 8 Message Length - Message Length: 0-65536 Increment 8	SP 800-38B
AES-CTR	A2314	Direction - Decrypt, Encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 8-128 Increment 8 Supports Counter larger than maximum value - Yes Incremental Counter - Yes Counter Tests Performed - Yes	SP 800-38A
AES-ECB	A2314	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A2314	Direction - Decrypt, Encrypt IV Generation - External, Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 8-1024 Increment 8 Payload Length - Payload Length: 128, 136, 272, 384 AAD Length - AAD Length: 0, 128, 136, 272, 384	SP 800-38D
AES-KW	A2314	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 128-4096 Increment 64	SP 800-38F
AES-KWP	A2314	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 128-4096 Increment 64	SP 800-38F
AES-OFB	A2314	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-XTS Testing Revision 2.0	A2314	Direction - Decrypt, Encrypt Key Length - 128, 256 Payload Length - Payload Length: 128-65536	SP 800-38E

Algorithm	CAVP Cert	Properties	Reference
		Increment 128 Tweak Mode - Number Data Unit Length Matches Payload Length - Yes	
Counter DRBG	A2314	Prediction Resistance - No, Yes Supports Reseed - Yes Mode - AES-128, AES-192, AES-256 Derivation Function Enabled - Yes Additional Input - Additional Input: 0, 128, Additional Input: 0, 256 Entropy Input - Entropy Input: 128, Entropy Input: 192, Entropy Input: 256 Nonce - Nonce: 128, Nonce: 64, Nonce: 96 Personalization String Length - Personalization String Length: 0, 128, Personalization String Length: 0, 256 Returned Bits - 512	SP 800-90A Rev. 1
DSA KeyGen (FIPS186-4)	A2314	L - 2048, 3072 N - 224, 256	FIPS 186-4
DSA PQGGen (FIPS186-4)	A2314	P/Q Generation Methods - Probable G Generation Methods - Unverifiable L - 2048, 3072 N - 224, 256 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
DSA PQGVer (FIPS186-4)	A2314	P/Q Generation Methods - Probable G Generation Methods - Unverifiable 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)	A2314	L - 2048, 3072 N - 224, 256 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	FIPS 186-4
DSA SigVer (FIPS186-4)	A2314	L - 1024, 2048, 3072 N - 160, 224, 256 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	FIPS 186-4
ECDSA KeyGen (FIPS186-4)	A2314	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)	A2314	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)	A2314	Component - No 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	FIPS 186-4

Algorithm	CAVP Cert	Properties	Reference
		Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	
ECDSA SigVer (FIPS186-4)	A2314	Component - No 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, SHA2-512/224, SHA2-512/256	FIPS 186-4
Hash DRBG	A2314	Prediction Resistance - No, Yes Supports Reseed - Yes Mode - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 Entropy Input - Entropy Input: 128, Entropy Input: 192, Entropy Input: 256 Nonce - Nonce: 128, Nonce: 64, Nonce: 96 Personalization String Length - Personalization String Length: 0, 128, Personalization String Length: 0, 192, Personalization String Length: 0, 256 Additional Input - Additional Input: 0, 128, Additional Input: 0, 192, Additional Input: 0, 256 Returned Bits - 1024, 1536, 2048, 896	SP 800-90A Rev. 1
HMAC DRBG	A2314	Prediction Resistance - No, Yes Supports Reseed - Yes Mode - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 Entropy Input - Entropy Input: 128, Entropy Input: 192, Entropy Input: 256 Nonce - Nonce: 128, Nonce: 64, Nonce: 96 Personalization String Length - Personalization String Length: 0, 128, Personalization String Length: 0, 192, Personalization String Length: 0, 256 Additional Input - Additional Input: 0, 128, Additional Input: 0, 192, Additional Input: 0, 256 Returned Bits - 1024, 1536, 2048, 896	SP 800-90A Rev. 1
HMAC-SHA-1	A2314	MAC - MAC: 160 Key Length - Key Length: 8-51200 Increment 8	FIPS 198-1
HMAC-SHA2-224	A2314	MAC - MAC: 224 Key Length - Key Length: 8-51200 Increment 8	FIPS 198-1
HMAC-SHA2-256	A2314	MAC - MAC: 256 Key Length - Key Length: 8-51200 Increment 8	FIPS 198-1
HMAC-SHA2-384	A2314	MAC - MAC: 384 Key Length - Key Length: 8-51200 Increment 8	FIPS 198-1
HMAC-SHA2-512	A2314	MAC - MAC: 512 Key Length - Key Length: 8-51200 Increment 8	FIPS 198-1

Algorithm	CAVP Cert	Properties	Reference
HMAC-SHA2-512/224	A2314	MAC - MAC: 224 Key Length - Key Length: 8-51200 Increment 8	FIPS 198-1
HMAC-SHA2-512/256	A2314	MAC - MAC: 256 Key Length - Key Length: 8-51200 Increment 8	FIPS 198-1
HMAC-SHA3-224	A2314	MAC - MAC: 224 Key Length - Key Length: 8-51200 Increment 8	FIPS 198-1
HMAC-SHA3-256	A2314	MAC - MAC: 256 Key Length - Key Length: 8-51200 Increment 8	FIPS 198-1
HMAC-SHA3-384	A2314	MAC - MAC: 384 Key Length - Key Length: 8-51200 Increment 8	FIPS 198-1
HMAC-SHA3-512	A2314	MAC - MAC: 512 Key Length - Key Length: 8-51200 Increment 8	FIPS 198-1
KAS-ECC CDH-Component (CVL)	A2314	Function - Full Public Key Validation, Key Pair Generation, Partial Public Key Validation 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	SP 800-56A Rev. 3
KAS-ECC-SSC Sp800-56Ar3	A2314	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 staticUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-FFC-SSC Sp800-56Ar3	A2314	Domain Parameter Generation Methods - FB, FC, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Scheme - dhEphem - KAS Role - initiator, responder dhOneFlow - KAS Role - initiator, responder dhStatic - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-IFC-SSC	A2314	Modulo - 2048, 3072, 4096, 6144, 8192 Key Generation Methods - rsakpg1-basic Scheme - KAS1 - KAS Role - initiator, responder Fixed Public Exponent - 010001	SP 800-56A Rev. 3
KAS-KC SP800-56 (CVL)	A2314	KAS Role - Initiator, Responder Key Confirmation Methods - Key Confirmation Directions - Bilateral, Unilateral Key Confirmation Roles - Provider, Recipient Key Confirmation MAC Methods - HMAC-SHA-1 - Key Length - 128	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
		MAC Length - 128 HMAC-SHA2-224 - Key Length - 128 MAC Length - 128 HMAC-SHA2-256 - Key Length - 128 MAC Length - 128 HMAC-SHA2-384 - Key Length - 128 MAC Length - 128 HMAC-SHA2-512224 - Key Length - 128 MAC Length - 128 HMAC-SHA2-512 - Key Length - 128 MAC Length - 128 HMAC-SHA2-512256 - Key Length - 128 MAC Length - 128 HMAC-SHA3-224 - Key Length - 128 MAC Length - 128 HMAC-SHA3-256 - Key Length - 128 MAC Length - 128 HMAC-SHA3-384 - Key Length - 128 MAC Length - 128 HMAC-SHA3-512 - Key Length - 128 MAC Length - 128	
KDA OneStep Sp800-56Cr1	A2314	Auxiliary Function Methods - Auxiliary Function Name - SHA-1 MAC Salting Methods - default Fixed Info Pattern - context||uPartyInfo||vPartyInfo Fixed Info Encoding - concatenation Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-8192 Increment 8	SP 800-56C Rev. 2
KDF SP800-108	A2314	KDF Mode - Feedback MAC Mode - HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA2-512/224, HMAC-SHA2-512/256 Supported Lengths - Supported Lengths: 8-4096 Increment 8 Fixed Data Order - After Fixed Data Counter Length - 8 Supports Empty IV - Yes	SP 800-108 Rev. 1

Algorithm	CAVP Cert	Properties	Reference
		Requires Empty IV - No Custom Key In Length - 0	
PBKDF	A2314	Iteration Count - Iteration Count: 1-10000000 Increment 1 HMAC Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Password Length - Password Length: 8-128 Increment 1 Salt Length - Salt Length: 128-4096 Increment 8 Key Data Length - Key Data Length: 112-4096 Increment 8	SP 800-132
RSA Decryption Primitive (CVL)	A2314	Modulus Length - 2048	FIPS 186-4
RSA KeyGen (FIPS186-4)	A2314	Key Generation Mode - B.3.6 Modulo - 2048, 3072, 4096 Primality Tests - Table C.3 Info Generated By Server - Yes Public Exponent Mode - Fixed Fixed Public Exponent - 010001 Private Key Format - Chinese Remainder Theorem	FIPS 186-4
RSA SigGen (FIPS186-4)	A2314	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA2-224	FIPS 186-4
RSA SigVer (FIPS186-4)	A2314	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA-1 Public Exponent Mode - Fixed Fixed Public Exponent - 010001	FIPS 186-4
Safe Primes Key Generation	A2314	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3
Safe Primes Key Verification	A2314	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3
SHA-1	A2314	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-224	A2314	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4

Algorithm	CAVP Cert	Properties	Reference
SHA2-256	A2314	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A2314	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A2314	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512/224	A2314	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512/256	A2314	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA3-224	A2314	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-256	A2314	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-384	A2314	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-512	A2314	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHAKE-128	A2314	Supports Bit-Oriented Messages - No Supports Empty Message - Yes Supports Bit-Oriented Output - No Output Length - Output Length: 16-65536 Increment 8	FIPS 202
SHAKE-256	A2314	Supports Bit-Oriented Messages - No Supports Empty Message - Yes Supports Bit-Oriented Output - No Output Length - Output Length: 16-65536 Increment 8	FIPS 202
TLS v1.2 KDF RFC7627 (CVL)	A2314	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
TLS v1.3 KDF (CVL)	A2314	HMAC Algorithm - SHA2-256, SHA2-384 KDF Running Modes - DHE, PSK, PSK-DHE	SP 800-135 Rev. 1

Table 5: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG-1	Key Type:Asymmetric	N/A	NIST SP800-133r2 Section 4: Using the Output of a Random Bit Generator; Section 5.1: Key Pairs for Digital Signature Schemes; Section 5.2: Key Pairs for Key Establishment
CKG-2	Key Type:Symmetric	N/A	NIST SP800-133r2 Sections 4, Using the Output of a Random Bit Generator; Section 6.1: Direct Generation of Symmetric Keys; Section 6.2: Derivation of Symmetric keys

Name	Properties	Implementation	Reference
CKG-XTS	Key Type:Symmetric	N/A	SP 800-133r2 and IG D.H: Per Section 6.3, approved method 1. Applicable to AES-XTS compliant to IG C.I because Key_1 and Key_2 are concatenated prior to usage

Table 6: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
AES in BPS mode for FPE	Symmetric encryption
ChaCha20	Symmetric encryption
ChaCha20/Poly1305	Symmetric encryption
DES	Symmetric encryption
DESX	Symmetric encryption
Deterministic DSA	Digital signatures
Deterministic ECDSA (FIPS 186-5)	Digital signatures
ECIES	Asymmetric encryption
FIPS 186-2 PRNG (Change Notice General)	Random bit generation
HMAC-MD5	Message authentication
KDFTLS10	Key Derivation (For use with TLS versions 1.0 and 1.1)
MD2	Secure hashing
MD5	Secure hashing
PBE (PKCS #12, PKCS #5, SSLCPBE)	Symmetric encryption
PBHMAC (PKCS #12, PKIX)	Message authentication
Poly1305	Message authentication
RC2	Symmetric encryption
RC4	Symmetric encryption
RC5	Symmetric encryption
RIPEMD160	Secure hashing
RSA-KEM-KWS	Asymmetric encryption
scrypt	Key Derivation
Shamir Secret Sharing	Key Generation

Name	Use and Function
TDES in CBC, CFB64, ECB, OFB modes and CBC_CS1, CBC_CS2 or CBC_CS3 mode for CTS	Symmetric encryption

Table 7: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Random Number Generation	DRBG	Perform random number generation		Counter DRBG: (A2314) HMAC DRBG: (A2314) Hash DRBG: (A2314)
Cryptographic Key Generation (CKG)	CKG	Direct generation of symmetric keys per NIST SP 800- 133r2		CKG-2: () Key Type: Symmetric
AES-XTS KeyGen (CKG)	CKG	AES XTS Key generated to comply with the approved key generation guidelines of NIST SP 800-133rev2, Section 6.3		CKG-XTS: () Key Type: Symmetric
Asymmetric Key Generation/Verification	AsymKeyPair-DomPar AsymKeyPair-KeyGen AsymKeyPair-KeyVer AsymKeyPair-PubKeyVal	Perform DSA, ECDSA, RSA, SafePrime keypair generation/verification		DSA KeyGen (FIPS186-4): (A2314) DSA PQGGen (FIPS186-4): (A2314) DSA PQGVer (FIPS186-4): (A2314) ECDSA KeyGen (FIPS186-4): (A2314) ECDSA KeyVer (FIPS186-4): (A2314) RSA KeyGen

Name	Type	Description	Properties	Algorithms
				(FIPS186-4): (A2314) Counter DRBG: (A2314) Hash DRBG: (A2314) HMAC DRBG: (A2314) Safe Primes Key Generation: (A2314) Safe Primes Key Verification: (A2314) CKG-1: () Key Type: Asymmetric
KAS-ECC Keypair Generation	KAS-KeyGen	Perform KAS-ECC keypair generation		Counter DRBG: (A2314) Hash DRBG: (A2314) HMAC DRBG: (A2314) CKG-1: () Key Type: Asymmetric
KAS-FFC Keypair Generation	KAS-KeyGen	Perform KAS-FFC keypair generation		Safe Primes Key Generation: (A2314) Counter DRBG: (A2314) Hash DRBG: (A2314) HMAC DRBG: (A2314) CKG-1: () Key Type: Asymmetric Safe Primes Key

Name	Type	Description	Properties	Algorithms
				Verification: (A2314)
KAS-IFC Keypair Generation	KAS-KeyGen	Perform KAS-IFC keypair generation		Counter DRBG: (A2314) Hash DRBG: (A2314) HMAC DRBG: (A2314) CKG-1: () Key Type: Asymmetric RSA KeyGen (FIPS186-4): (A2314)
Shared Secret Calculation (KAS-ECC-SSC)	KAS-SSC	Perform Shared Secret Computation for KAS-ECC		KAS-ECC-SSC Sp800-56Ar3: (A2314) KAS-ECC CDH-Component: (A2314)
Shared Secret Calculation (KAS-FFC-SSC)	KAS-SSC	Perform shared secret computation for KAS-FFC		KAS-FFC-SSC Sp800-56Ar3: (A2314)
Shared Secret Calculation (KAS-IFC-SSC)	KAS-SSC	Perform shared secret computation for RSA IFC		KAS-IFC-SSC: (A2314)
KAS Key Confirmation	KAS-KC	Perform KAS key confirmation		KAS-KC SP800-56: (A2314)
Key Derivation	KAS-135KDF KAS-56CKDF KBKDF PBKDF	Perform key derivation function		KDA OneStep Sp800-56Cr1: (A2314) KDF SP800-108: (A2314) PBKDF: (A2314) TLS v1.2 KDF RFC7627: (A2314) TLS v1.3

Name	Type	Description	Properties	Algorithms
				KDF: (A2314)
Key Wrap/Unwrap	KTS-Wrap	Perform key wrap operation		AES-KW: (A2314) AES-KWP: (A2314)
Unauthenticated Symmetric Encryption and Decryption	BC-UnAuth	Unauthenticated Symmetric Encryption and Decryption		AES-CBC: (A2314) AES-CBC-CS1: (A2314) AES-CBC-CS2: (A2314) AES-CBC-CS3: (A2314) AES-CFB128: (A2314) AES-CTR: (A2314) AES-ECB: (A2314) AES-OFB: (A2314) AES-XTS Testing Revision 2.0: (A2314)
Authenticated Symmetric Encryption/Decryption	BC-Auth	Perform authenticated symmetric encryption operation		AES-CCM: (A2314) AES-GCM: (A2314)
Message Digest	SHA	Perform message digest operation		SHA-1: (A2314) SHA2-224: (A2314) SHA2-256: (A2314) SHA2-384: (A2314) SHA2-512: (A2314) SHA2-512/224: (A2314) SHA2-512/256:

Name	Type	Description	Properties	Algorithms
				(A2314) SHA3-224: (A2314) SHA3-256: (A2314) SHA3-384: (A2314) SHA3-512: (A2314) SHAKE-128: (A2314) SHAKE-256: (A2314)
MAC Generation/Verification	MAC	Perform MAC generation or verification		AES-CMAC: (A2314) HMAC-SHA-1: (A2314) HMAC-SHA2-224: (A2314) HMAC-SHA2-256: (A2314) HMAC-SHA2-384: (A2314) HMAC-SHA2-512: (A2314) HMAC-SHA2-512/224: (A2314) HMAC-SHA2-512/256: (A2314) HMAC-SHA3-224: (A2314) HMAC-SHA3-256: (A2314) HMAC-SHA3-384: (A2314) HMAC-SHA3-512: (A2314)

Name	Type	Description	Properties	Algorithms
Digital Signature Generation	DigSig-SigGen	Perform DSA, ECDSA, RSA signature generation		DSA SigGen (FIPS186-4): (A2314) keysizes: 2048 or 3072 bits ECDSA SigGen (FIPS186-4): (A2314) RSA SigGen (FIPS186-4): (A2314)
Digital Signature Verification (legacy)	DigSig-SigVer	Perform signature verification with DSA 1024 bits		DSA SigVer (FIPS186-4): (A2314) keysize: 1024 bits
RSADP Primitive	UNK	RSADP primitive generation		RSA Decryption Primitive: (A2314)
Digital Signature Verification	DigSig-SigVer	Perform DSA, ECDSA, RSA signature verification		DSA SigVer (FIPS186-4): (A2314) : ECDSA SigVer (FIPS186-4): (A2314) RSA SigVer (FIPS186-4): (A2314)
TLS Session Encrypt/Decrypt	BC-Auth BC-UnAuth	TLSv1.2/v1.3 Session Encrypt/Decrypt		AES-CBC: (A2314) AES-GCM: (A2314)
TLS Session Authentication	MAC	TLSv1.2/v1.3 session authentication		HMAC-SHA-1: (A2314) HMAC-SHA2-224: (A2314) HMAC-SHA2-256: (A2314) HMAC-SHA2-384: (A2314) HMAC-

Name	Type	Description	Properties	Algorithms
				SHA2-512: (A2314) HMAC-SHA2-512/224: (A2314) HMAC-SHA2-512/256: (A2314) SHA-1: (A2314) SHA2-224: (A2314) SHA2-256: (A2314) SHA2-384: (A2314) SHA2-512: (A2314) SHA2-512/224: (A2314) SHA2-512/256: (A2314)

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

AES-GCM

The module supports AES-GCM for symmetric encryption in compliance with SP800-38D and is compatible with TLS 1.2 and TLS 1.3 protocols. While the module does not implement TLS itself, it provides the cryptographic functions required for implementing these protocols, including AES-GCM cipher suites specified in Section 3.3.1 of SP800-52r2.

TLS Usage

For TLS 1.2, IV construction aligns with IG C.H scenario 1 and RFC5288:

- A 4-byte salt derived from the TLS handshake is input using the parameter PARTIAL_IV during cipher initialization. This is used as the first four bytes of IV. This 32-bit part of the IV is also referred to as the nonce value in FIPS140-3 IG C.H and is positioned in the name field of the IV as required in FIPS140-3 IG C.H, TLS/DTLS 1.2 protocol IV generation.
- The remaining eight bytes of IV, referred to as nonce_explicit in RFC5288, are generated deterministically by the module using a 64-bit counter.

- The counter portion of the IV is strictly increasing. When the 64-bit counter exhausts the maximum number of possible values for a given session key, the module will throw a `SecurityException`.
- When this occurs, encryption fails, and a handshake to establish a new encryption key is required. The first party encountering this condition (client or server) must trigger the handshake in accordance with RFC5246.

For TLS 1.3, IV construction aligns with IG C.H scenario 5 and RFC8446.

The TLS session is aborted if the keys for the client and server negotiated in the handshake process, `client_write_key` and `server_write_key`, are identical.

AES-GCM Outside TLS
When using GCM feedback mode for symmetric encryption, the authentication tag length and authenticated data length may be specified as input parameters, but the IV must not be specified. It must be generated internally.

Where the module is powered down, a new key must be used for AES GCM encryption/decryption.

GCM with a partial IV supplied to the module is approved only when used within a TLS v 1.2 or 1.3 protocol implementation.

The module supports internal IV generation by the module's approved DRBGs, in alignment with IG C.H scenario 2. The IV is at least 96 bits in length per SP800-38D, Section 8.2.2. The AES-GCM cipher, when used for symmetric encryption purposes other than TLS, must use an IV in one of the two possible ways, to comply with SP800-38D:

- Allow the module to generate the IV deterministically by not supplying any IV parameters during cipher initialization. The generated 96-bit (12-byte) IV consists of a 32-bit fixed field followed by a 64-bit invocation field where:
    - The fixed field bytes are derived from the module name, version information, and memory address of a Java class within the module.
    - The invocation field is a 64-bit counter that is initialized, on module startup, to a value consisting of the 42 bits of current time, as milliseconds since Epoch, followed by 22 bits of zero. This counter value is incremented by one each time a new IV is requested. By using the current time to prefix the counter start value, in the event of module restart, the counter will be ahead of any previous module states, ensuring that IV values cannot be reused. The module user must ensure the system time is valid to prevent repetition of IVs.
- Generate at least 12 bytes of IV using an Approved DRBG, and input the IV to the cipher at initialization time using the RAW_IV parameter.

XTS-AES
In accordance with SP800-38E, the XTS-AES algorithm is to be used for confidentiality on storage devices. The module complies with FIPS 140-3 IG C.I by:

- Generating Key_1 and Key_2 independently according to the rules for component symmetric keys from SP800-133r2, Section 6.3.

- Explicitly checking that Key_1 ≠ Key_2 before using the keys in the XTS-AES algorithm to process data with them

DSA

DSA KeyGen (FIPS186-4) and DSA PQGGen (FIPS186-4) are only implemented for use as a part of an approved SP800-56Ar3 FFC scheme. In accordance with this, only the FIPS186-type parameter sets FB (2048, 224) and FC (2048, 256) from SP800-56Ar3 are supported by the module. For DSA signatures, only DSA PQGVer (FIPS186-4) and DSA SigVer (FIPS186-4) are only implemented. Please refer to Security Policy Section 9.5 - Transitions for additional context.

PBKDF

The PBKDF aligns with Option 1a in Section 5.4 of SP800-132. Keys derived from passwords using the PBKDF may only be used in storage applications. The PBKDF function can be called using the Key Derivation service, but it does not establish keys into the module. The PBKDF function supports passwords from 8 to 128 bytes and iteration counts from 1 to the maximum integer value. SP800-132 Section 5.2 recommends a minimum iteration count of 1,000. Operators should select an appropriate password length and iteration count for their use case, bearing in mind that both should be as large as is feasible for the application.

- Keys generated using PBKDF2 shall only be used in data storage applications.
- Minimum Password Length:
  The minimum length (L) of a password generated using a cryptographically secure random password generator to provide a search space of S entries depends on the size (N) of the character set:
  $L = \lceil \log_2 S / \log_2 N \rceil$
  The following provides examples for a password used by PBKDF2 where S = 4.32×10^{20}:

Character Set	N	L
Case sensitive (a-z, A-Z)	52	13
Case sensitive alpha numeric	62	12
All ASCII printable characters except space	94	11

- A password of the strength S can be guessed at random with the probability of 1 in 2^S.
- The minimum length of the randomly-generated portion of the salt is 16 bytes.
- The iteration count is as large as possible, with a minimum of 1,000 iterations recommended.
- The maximum key length is $(2^{32} - 1) * b$, where b is the digest size of the message digest function in bytes.
- Derived keys can be used as specified in NIST SP800-132, Section 5.4, options 1 and 2.

TLS PRF Key Derivation Function

- TLS v1.2 PRF KDF is allowed only when the following conditions are satisfied:
  - The KDF is performed in the context of the TLS protocol.
  - HMAC is as specified in FIPS198-1.

    - P_HASH uses either SHA-256, SHA-384, or SHA-512. For more information, see SP800-135r1.
- The TLS protocols have not been tested by the CAVP and CMVP.

DRBG

- When an approved algorithm requires a DRBG to perform an operation, an approved DRBG algorithm must be used. For example, when initializing an approved signature algorithm, an approved DRBG such as HMAC DRBG must be used.
- When using an approved DRBG, the number of bytes of seed input must be equivalent to or greater than the security strength of the keys the caller wishes to generate. For example, a 256-bit or higher seed key input when generating 256-bit AES keys.
- Since the module does not modify the output of an Approved DRBG, any generated symmetric keys or seed values are created directly from the output of the Approved DRBG.

HMAC

- The key length for an HMAC generation or verification must be between 112 and 4096 bits, inclusive.
- For HMAC verification, a key length greater than or equal to 80 and less than 112 is allowed for legacy-use.

HMAC-Based Extract-and-Expand Key Derivation Function

- An approved HMAC must be used for extract and expand operations.
- A particular key-derivation key must only be used for a single key-expansion step. For more information, see SP800-56Cr1.
- The derived key must be used only as a secret key.
- The derived key shall not be used as a key stream for a stream cipher.
- When selecting an HMAC hash, the output block size must be equal to or greater than the desired security strength of the derived key.
- The pseudo-random key input to the expansion and the keying material output from the expansion must have lengths that are equal to or greater than the desired security strength of the derived key.

One-Step Key Derivation Function

- An approved hash function must be used to derive key materials.
- When selecting a hash algorithm, the output block size must be equal to or greater than the desired security strength of the derived key.
- The derived key must be used only as a secret key.
- The derived key shall not be used as a key stream for a stream cipher.
- The secret data input into this KDF must have a length equal to or greater than the desired security strength of the derived key.

Parameter Generation

When using an Approved DRBG to generate DH or DSA parameters, the requested DRBG must have a security strength at least as great as the security strength of the parameters being generated. That means that an Approved DRBG with an appropriate strength must be used. For more information on requesting the DRBG security strength, see the relevant API Javadoc.

Key Agreement

Obtain domain parameters and assurance of the domain parameter validity:

- For schemes using FFC, use one of the FFC safe-prime groups as defined in SP800-56Ar3, Appendix D
- For schemes using ECC, use one of the approved curves as defined in SP800-56Ar3, Appendix D.

Obtain a key pair from domain parameters:

- For all schemes:
  - Both parties must use validated parameters to generate a key pair.
  - The module generates the key establishment key pair according to the required standards.
  - Choose a FIPS Approved DRBG like HMAC DRBG to generate the key pair.
  - Both parties validate the key pair:
    The module provides the following APIs to explicitly validate the public and private keys according to SP 800-56Ar3:
    `com.rsa.crypto.PublicKey.isValid(SecureRandom random)`
    `com.rsa.crypto.PrivateKey.isValid().`
    The module provides the APIs to explicitly validate the key pair according to the pairwise consistency requirements in SP800-56Ar3:
    `com.rsa.crypto.KeyPair.validate(SecureRandom random)`
    `com.rsa.crypto.KeyPair.validate(AlgorithmParams params, SecureRandom random)`
    If the key pair is generated with an approved method, then validation is assumed.
- For schemes that use static key pairs, a public identifier must be:
  - Authoritatively associated with the key pair.
  - Associated with the public key to allow any peer to recognize the key pair.
- For schemes that use ephemeral keys, the key pair must be:
  - Used only for a single agreement transaction.
  - Destroyed after use.
- For schemes that generate an FFC key pair from selected parameters, the key pair must not be used to generate a digital signature.

Receive the peer's public key:

- For all schemes, the receiving party must validate the peer's public key.
- For schemes that use static keys, the receiving party must have assurance of:
  - The peer's ownership of the private key.
  - The identifier is bound to the public key.

Generate the Shared Secret:

- For all schemes, the shared secret must be:
  - Used only as input to an approved KDF.
  - Treated as a CSP and destroyed after use.
- If the shared secret generation fails, then the party must destroy all intermediate values.

Generate and Confirm Secret Key Material:

- For all schemes:
  - Approved key-derivation method(s), including the format of `FixedInfo` as specified in SP 800-56Ar3.
  - When the shared secret is used as input to the KDF the outputs must be used as secret keys.
  - All key material must be generated before any of the keys are used.
  - If key generation fails, then the party must destroy all calculated values.
  - The shared secret, and any key material, is destroyed.
- For schemes that use key confirmation:
  - Both parties must use a common, approved MAC to generate confirmation values.
  - The MAC key will be generated as one of the key material elements.
  - The input values for MAC tag generation must be formatted as per SP800-56Ar3.
  - The MAC key and tag lengths must satisfy the requirements of SP800-56Ar3.
  - The MAC key must be destroyed after use.
  - If confirmation fails, then destroy all calculated values.

  All key material is destroyed before it is used for any other purpose.
  Approved key confirmation technique(s) as specified in SP800-56Ar3.

Key Generation

- When using an approved DRBG to generate keys, the security strength of the DRBG must be at least as great as the security strength of the key being generated.

- When generating key pairs using the `KeyPairGenerator` object, the `generate(boolean pairwiseConsistency)` method must not be invoked with an argument of false. Use of the no-argument `generate()` method is recommended.

Digital Signatures

- Keys used for digital signature generation and verification shall not be used for any other purpose. The module generates keys with a particular purpose that is either signing or encryption. The same purpose must always be used for a given key when exported and loaded into the module again.
- The length of an RSA key pair for digital signature generation must be greater than or equal to 2048 bits. For digital signature verification, the length must be greater than or equal to 2048 bits. However, 1024 bits is allowed for legacy-use only. RSA keys shall have a public exponent of an odd number, equal to or greater than 65537.
- The SHA-1 digest is disallowed for the generation of digital signatures.
- For RSASSA-PSS: If `nLen` is 1024 bits, and the output length of the approved hash function output block is 512 bits, then the length of the salt (`sLen`) shall `be 0<= sLen <= hLen –`

2. Otherwise, the length of the salt shall be 0 <= sLen <= hLen, where hLen is the length of the hash function output block (in bytes or octets).

XTS Mode Ciphers

- AES in XTS mode is approved only for hardware storage applications.
- The two keys used for XTS must be checked to ensure they are different. This check is performed automatically by the module.

2.8 RBG and Entropy

N/A for this module.

N/A for this module.

The module is passively receiving the entropy while exercising no control over the amount or the quality of the obtained entropy. Therefore, it is the user's responsibility to supply the entropy to seed an RBG to provide the required security strength, and to ensure the security strength of a DRBG is equal to or greater than the security strength of any SSPs generated using that DRBG. Entropy can be supplied to the module using the following APIs:

- `com.rsa.crypto.SecureRandom.setSeed()`
- `com.rsa.crypto.ModuleConfig.setEntropySource()`

The module does not include an entropy source. The module aligns with IG 9.3.A, scenario 2b, therefore the module's certificate includes the caveat "No assurance of the minimum strength of generated SSPs (e.g., keys)."

The module accepts input from entropy sources external to the cryptographic boundary for use as seed material for the module's approved DRBG implementations. Entropy is supplied to the module by means of callback functions. Those functions return an error if the minimum entropy strength is not met. Entropy strength requirements are per NIST SP800-90Ar1, Table 2 (Hash_DRBG, HMAC_DRBG) and Table 3 (CTR_DRBG). At a minimum, the entropy source shall provide at least 128 bits of entropy to the DRBG.

All random values used by the module for approved algorithms are provided by the module's approved DRBGs.

The module includes Counter DRBG, Hash DRBG, and HMAC DRBG, all of which are approved RBGs. The output of these approved RBGs is used to generate random data, symmetric keys, and asymmetric keys, as indicated in Security Policy Section 2.5.2 - Vendor-Affirmed Algorithms.

When generating SSPs, the DRBG used in key generation must be seeded with a number of bits of entropy that is equal to or greater than the security strength of the SSP being generated. The entropy supplied to the DRBG is referred to as the DRBG security strength which represents the minimum amount of entropy that should be provided to the DRBG prior to generating the SSP.

2.9 Key Generation

Any generated SSPs are passed out to the calling application and are not stored in the module. Additional details are provided in Security Policy Section 2.6 - Security Function Implementations and Section 4.3 - Approved Services.

Random values for key generation are provided by the module's approved DRBGs.
The output of the module's approved DRBGs may be used to generate symmetric and asymmetric keys per SP800-133r1, as indicated in Security Policy Section 2.5.2 - Vendor-Affirmed Algorithms. The module is a software library that provides a service (called Random Number Generation) for direct output of the approved DRBG (U). This output is approved for generating keys or SSPs.

Symmetric keys are generated per SP800-133r2 Section 6.1 using the Random Number Generation service; additionally, Section 6.3 is applicable for AES-XTS keys. Asymmetric keys are generated per SP 800-133r2 Section 5 per FIPS186-4 using the Asymmetric Key Generation service.

2.10 Key Establishment

SSPs used for services are passed in by the calling application. Established SSPs are passed out to the calling application and are not stored in the module. Additional details are provided in Security Policy Section 2.6 - Security Function Implementations and Section 4.3 - Approved Services.

The module does not establish SSPs using an approved key agreement scheme (KAS). However, it does offer some or all of the underlying KAS cryptographic functionality to be used by an external operator/application as part of an approved KAS.

- The module provides ECC and FFC shared secret computation that is conformant to SP800-56Ar3 in alignment with IG D.F scenario 2 (path 1).
    - For ECC, the module supports the (Cofactor) Ephemeral Unified Model, and staticUnified Scheme described in SP800-56Ar3, Section 6.1.2.2. The module also implements ECC CDH-Component.
    - For FFC, the module supports the dhEphem, dhOneFlow and dhStatic Schemes described in SP800-56Ar3.
    - The appropriate public key validation assurances are implemented. For ECC, full public key validation is implemented (SP800-56Ar3, Section 5.6.2.3.3). For FFC, both full public key validation (per SP800-56Ar3, Section 5.6.2.3.1) and partial public key validation (per SP800-56Ar3, Section 5.6.2.3.2) are implemented.

- The module provides RSA shared secret computation (KAS-IFC-SSC) that is conformant to SP800-56Br2 in alignment with IG D.F scenario 1 (path 1) via the Key Agreement (RSA) service. The module supports the KAS1 basic scheme.

The module supports various key derivation functions separately via the Key Derivation service. Supported KDFs are conformant to SP800-108r1 (KBKDF), SP800-132 (PBKDF), SP800-56Cr2 (OneStep KDA), SP800-135r1 (TLS v1.2 KDF RFC7627), and RFC 8446 (TLS 1.3 KDF).

The module provides RSA Decryption Primitive (CVL) component that is conformant to SP800-56Br2.

The module provides AES key wrapping (AES KW, AES KWP) that is conformant to SP800-38F via the Key Wrapping service.

2.11 Industry Protocols

The module implements TLS v1.2 KDF RFC7627 and the TLS 1.3 KDF industry protocols. These KDFs have been validated by the CAVP and received CVL certificates (A2314). No parts of these protocols, other than the approved cryptographic algorithms and the KDFs, have been tested by the CAVP and CMVP.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	Plaintext, Ciphertext, Message Digest, Signature, MAC, Secret, Key text, Wrapped key text, Message, Secret
N/A	Data Output	Status, Ciphertext, Plaintext, Verify status, Validation status, Wrapped key text, Message digest, MAC, Random bytes
N/A	Control Input	Configuration parameters for the API interface ModuleConfig which sets the mode of operation
N/A	Status Output	Mode of operation indicator from the API CryptoModule.isFIPS140Approved(). The state of the module from the API CryptoModule.getState()
N/A	Control Output	N/A

Table 9: Ports and Interfaces

The module's physical perimeter encompasses the case of the tested platform mentioned in Table 2 Tested Operational Environment. The module provides its logical interfaces via API calls. The logical interfaces provided by the module are mapped onto the FIPS 140-3 logical interfaces (Data Input, Data Output, Control Input, Control Output, and Status Output).

4 Roles, Services, and Authentication

4.1 Authentication Methods

The Module meets all FIPS 140-3 Security Level 1 requirements for Roles, Services; and Authentication, implementing a Crypto Officer Role. As allowed by FIPS 140-3, the module does not support identification or authentication for this role. The Crypto Officer Role is implicitly assumed once the module is loaded, and the role is cleared on module unload. There is no maintenance role, cryptographic bypass capability, or self-initiated cryptographic output. The module does not allow concurrent operators.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 10: Roles

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show Status	Provide Module's current status (return codes and/or syslog messages)	N/A	Request of show Module's Status	Module's operational status	None	Crypto Officer
Show Version	Provide Module's name and version information	N/A	Request of show version	Module's ID and versioning information	None	Crypto Officer
Perform Self-Tests	Perform Self-Tests	N/A	Request of Self-Test	Status of Self-Tests	None	Crypto Officer
Perform Zeroization	Perform zeroization	N/A	Request of keys zeroization	Status of Keys Zeroization	None	Crypto Officer - DRBG

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Entropy Input: Z - CTR_DRBG Seed: Z - CTR_DRBG V: Z - CTR_DRBG Key: Z - Hash_DRBG Seed: Z - Hash_DRBG V: Z - Hash_DRBG C: Z - HMAC_DRBG Seed: Z - HMAC_DRBG V: Z - HMAC_DRBG Key: Z - Diffie-Hellman

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Private Key: Z - Diffie-Hellman Public Key: Z - Diffie-Hellman Shared Secret: Z - EC Diffie-Hellman Private Key: Z - EC Diffie-Hellman Public Key: Z - EC Diffie-Hellman Shared Secret: Z - DSA SGK: Z - DSA SVK: Z - ECDSA SGK: Z - ECDSA SVK: Z - RSA SGK:

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Z - RSA SVK: Z - KAS-IFC Private Key: Z - KAS-IFC Public Key: Z - KAS-IFC-SSC Shared Secret: Z - RSADP Primitive Private Key: Z - RSADP Primitive Public Key: Z - AES EDK: Z - AES-XTS key: Z - AES key wrapping key: Z - AES-CCM key: Z - AES-GCM Key: Z - AES-

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						CMAC key: Z - HMAC Key: Z - KBKDF Key Derivation Key: Z - KBKDF Derived Key: Z - OneStep KDF Key Derivation Key: Z - OneStep KDF Derived Key: Z - PBKDF Password: Z - PBKDF Derived Key: Z - TLS Master Secret: Z - TLS

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Session Encryption Key: Z - TLS Session Integrity Key: Z
Random Number Generation	Perform Random Number Generation	API function CryptoModule.isFips140Mode() returns true, plus successful completion of service	Request of Random Number Generation	Status of Random Number Generation	Random Number Generation	Crypto Officer - DRBG Entropy Input: G,E - CTR_DRBG Seed: W,E - CTR_DRBG V: W,E - CTR_DRBG Key: W,E - Hash_DRBG Seed: W,E - Hash_DRBG V: W,E - Hash_DRBG C: W,E - HMAC

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						_DRBG Seed: W,E - HMAC_DRBG V: W,E - HMAC_DRBG Key: W,E
KAS Keypair Generation (ECC/FFC/IFC)	KAS keypair generation	API function CryptoModule.isFips140Mode() returns true, plus successful completion of service	Request of KAS keypair generation (ECC/FFC/IFC)	Status of KAS keypair generation (ECC/FFC/IFC)	KAS-ECC Keypair Generation KAS-FFC Keypair Generation KAS-IFC Keypair Generation	Crypto Officer - EC Diffie-Hellman Private Key: G,W,E - EC Diffie-Hellman Public Key: G,W,E - Diffie-Hellman Private Key: G,W,E - Diffie-Hellman Public Key: G,W,E - KAS-IFC Private Key: G,W,E - KAS-

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						IFC Public Key: G,W,E
Shared Secret generation (KAS-ECC-SSC)	KAS-ECC shared secret generation	API function CryptoModule.isFips140Mode() returns true, plus successful completion of service	Request of shared secret generation (KAS-ECC-SSC)	Status of shared secret generation (KAS-ECC-SSC)	Shared Secret Calculation (KAS-ECC-SSC)	Crypto Officer - EC Diffie-Hellman Shared Secret: W,E
Shared Secret generation (KAS-FFC-SSC)	KAS-FFC shared secret generation	API function CryptoModule.isFips140Mode() returns true, plus successful completion of service	Request of shared secret generation (KAS-FFC-SSC)	Status of shared secret generation (KAS-FFC-SSC)	Shared Secret Calculation (KAS-FFC-SSC)	Crypto Officer - Diffie-Hellman Shared Secret: G,E
Shared Secret generation (KAS-IFC-SSC)	KAS-IFC shared secret generation	API function CryptoModule.isFips140Mode() returns true, plus successful completion of service	Request of shared secret generation (KAS-IFC-SSC)	Status of shared secret generation (KAS-IFC-SSC)	Shared Secret Calculation (KAS-IFC-SSC)	Crypto Officer - KAS-IFC-SSC Shared Secret: G,E
Asymmetric Keypair Generation and Verification	Perform asymmetric keypair generation and verification	API function CryptoModule.isFips140Mode() returns true, plus successful completion of service	Request of asymmetric keypair generation and verification	Status of asymmetric keypair generation and verification	Asymmetric Key Generation /Verification	Crypto Officer - DSA SGK: G,W,E - DSA SVK: G,W,E - ECDSA SGK: G,W,E - ECDSA SVK: G,W,E - RSA

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						SGK: G,W,E - RSA SVK: G,W,E
RSADP Primitive	Perform RSADP primitive operation	API function CryptoModule.isFips140Mode() returns true, plus successful completion of service	Request of RSADP Primitive	Status of RSADP Primitive	RSADP Primitive	Crypto Officer - RSADP Primitive Private Key: G,E - RSADP Primitive Public Key: E
Key Confirmation	Perform key confirmation	API function CryptoModule.isFips140Mode() returns true, plus successful completion of service	Request of key confirmation	Status of key confirmation	KAS Key Confirmation	Crypto Officer - Diffie-Hellman Shared Secret: E - EC Diffie-Hellman Shared Secret: E - HMAC Key: E
Key Derivation	Perform Key Derivation	API function CryptoModule.isFips140Mode() returns true, plus successful completion of service	Request of key derivation	Status of key derivation	Key Derivation	Crypto Officer - KBKDF Key Derivation

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Key: E - KBKDF Derived Key: E - OneStep KDF Key Derivation Key: E - OneStep KDF Derived Key: E - PBKDF Password: E - PBKDF Derived Key: E - TLS Session Encryption Key: E - TLS Session Integrity Key: E
Key Wraping	Perform key wrap	API function CryptoModule.isFips140Mode(	Request of key wrapping	Status of key wrapping	Key Wrap/Unwrap	Crypto Officer - AES

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	and unwrap	) returns true, plus successful completion of service				key wrapping key: G,R,E
Symmetric Encryption/ Decryption	Perform symmetric encryption operations	API function CryptoModule.isFips140Mode( ) returns true, plus successful completion of service	Request of symmetric encryption/ decryption	Status of symmetric encryption/ decryption	Unauthenticated Symmetric Encryption and Decryption Authenticated Symmetric Encryption/ Decryption	Crypto Officer - AES EDK: G,R,E - AES-CCM key: G,R,E - AES-GCM Key: G,R,E
Digital Signature Generation	Perform digital signature generation	API function CryptoModule.isFips140Mode( ) returns true, plus successful completion of service	Request of digital signature verification	Status of digital signature verification	Digital Signature Generation	Crypto Officer - DSA SGK: R,W,E - ECDSA SGK: R,W,E - RSA SGK: R,W,E
Digital Signature Verification	Perform digital signature verification	API function CryptoModule.isFips140Mode( ) returns true, plus successful completion of service	Request of digital signature verification (legacy)	Status of digital signature verification (legacy)	Digital Signature Verification (legacy) Digital Signature Verification	Crypto Officer - DSA SVK: R,W,E - ECDSA SVK: R,W,E - RSA SVK: R,W,E
Message Digest	Perform message digest operation	API function CryptoModule.isFips140Mode( ) returns true, plus successful	Request of message digest	Status of message digest	Message Digest	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		completion of service				
MAC Generation /Verification	Perform MAC Generation	API function CryptoModule.isFips140Mode() returns true, plus successful completion of service	Request of MAC generation /verification	Status of MAC generation /verification	MAC Generation /Verification	Crypto Officer - AES-CMAC key: G,R,W,E - HMAC Key: G,R,W,E
Keypair Assurance	Perform keypair assurance	API function CryptoModule.isFips140Mode() returns true, plus successful completion of service	Request of keypair assurance	Status of keypair assurance	Asymmetric Key Generation /Verification	Crypto Officer - DSA SGK: E - DSA SVK: E - ECDSA SGK: E - ECDSA SVK: E - RSA SGK: E - RSA SVK: E
Symmetric Key Generation	Perform symmetric key generation	API function CryptoModule.isFips140Mode() returns true, plus successful completion of service	Request of symmetric key generation	Status of symmetric key generation	Cryptographic Key Generation (CKG) AES-XTS KeyGen (CKG)	Crypto Officer - AES EDK: - AES-XTS key: - AES key wrapping key: - AES-

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						CCM key: - AES-GCM Key: - AES-CMAC key:
Asymmetric Key Generation	Perform asymmetric key generation	API function CryptoModule.isFips140Mode() returns true, plus successful completion of service	Request of asymmetric key generation	Status of Request of asymmetric key generation	Asymmetric Key Generation /Verification	Crypto Officer - DSA SGK: - DSA SVK: - ECDSA SGK: - ECDSA SVK: - RSA SGK: - RSA SVK:
TLS session operation	Perform TLS session encryption /decryption	API function CryptoModule.isFips140Mode() returns true, plus successful completion of service	Request of TLS session encrypt/decrypt	Status of TLS session encrypt/decrypt	TLS Session Encrypt/Decrypt TLS Session Authentication	Crypto Officer - TLS Master Secret: - TLS Session Encryption Key: - TLS Session Integrity Key:

Table 11: Approved Services

4.4 Non-Approved Services

Name	Description	Algorithms	Role
Asymmetric Encryption	Perform Asymmetric Encryption Operation	ECIES RSA-KEM-KWS	CO
Asymmetric Decryption	Perform Asymmetric Decryption Operation	ECIES RSA-KEM-KWS	CO
Digital Signature Generation	Perform Digital Signature Generation	Deterministic DSA Deterministic ECDSA (FIPS 186-5)	CO
Digital Signature Verification	Perform Digital Signature Verification	Deterministic DSA Deterministic ECDSA (FIPS 186-5)	CO
Key Derivation	Perform Key Derivation Operation	KDFTLS10 PBE (PKCS #12, PKCS #5, SSLCPBE) PBHMAC (PKCS #12, PKIX) scrypt	CO
Key Generation	Perform Key Generation Operation	DES DESX RC2 RC4 RC5 Shamir Secret Sharing	CO
Message Digest	Perform Message Digest Operation	MD2 MD5 RIPEMD160	CO
MAC Generation	Perform MAC Generation	HMAC-MD5 PBHMAC (PKCS #12, PKIX) Poly1305	CO
MAC Verification	Perform MAC Verification	HMAC-MD5 PBHMAC (PKCS #12, PKIX) RIPEMD160	CO
Random Number Generation	Perform Random Number Generation	FIPS 186-2 PRNG (Change Notice General)	CO
Symmetric Encryption	Perform Symmetric Encryption Operation	AES in BPS mode for FPE ChaCha20 ChaCha20/Poly1305 DES DESX PBE (PKCS #12, PKCS #5, SSLCPBE) RC2 RC4 RC5 TDES in CBC, CFB64, ECB, OFB modes and CBC_CS1, CBC_CS2 or CBC_CS3 mode for CTS	CO
Symmetric Decryption	Perform Symmetric Decryption Operation	AES in BPS mode for FPE ChaCha20 ChaCha20/Poly1305 DES DESX RC2 RC4	CO

Name	Description	Algorithms	Role
		RC5 TDES in CBC, CFB64, ECB, OFB modes and CBC_CS1, CBC_CS2 or CBC_CS3 mode for CTS	

Table 12: Non-Approved Services

4.5 External Software/Firmware Loaded

N/A for this module.

5 Software/Firmware Security

5.1 Integrity Techniques

BSAFE Java Crypto Module integrity check is implemented by first calculating a MAC over each of the files listed in module.files, using HMAC-SHA-1 (Algorithm Cert. #A2314) with a fixed key. Another MAC is then calculated over all file MACs in the order that they are listed, using the same algorithm and key as for the file MACs. This two-step process is intended to allow the jar file to be processed sequentially without having to load the entire jar file into memory even after the order of the jar file entries has been changed. The expected integrity check MAC is stored in the jar file manifest.

During the Integrity Test when the module is loaded, a MAC is again calculated and compared with the pre-computed MAC value contained in the jar file manifest. If these values are equal, then the software integrity check has passed and power-up of the module can continue. Otherwise, the test has failed, and the module is disabled.

5.2 Initiate on Demand

The integrity test is performed as part of the pre-operational self-tests. It is automatically executed at power-on. The module provides the `ModuleConfig.runSelfTests()` API to allow the operator to perform on-demand integrity testing. The operator can also power-cycle or reboot the tested platform to initiate the software integrity test on-demand.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

BSAFE Java Crypto Module is a software module, which is operated in a modifiable operational environment per FIPS 140-3 level 1 specifications. The module is provided for operating systems running on a general-purpose computer platform.

The module has control over its own SSPs. The process and memory management functionality of the host device's OS prevents unauthorized access to plaintext private and secret keys, intermediate key generation values, and other SSPs by external processes during module execution. The module only allows access to SSPs through its well-defined API. The operational 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 environment. Processes that are spawned by the module are owned by the module and are not owned by external processes or operators.

7 Physical Security

The requirements of this section are not applicable. The module is a software module and does not implement any physical security mechanisms.

8 Non-Invasive Security

The requirements of this area are not applicable to BSAFE Java Crypto Module.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
RAM memory	Volatile Memory (RAM) on the tested platform within TOEPP	Dynamic

Table 13: Storage Areas

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
API Input via TOEPP path	Other Applications (App per IG 9.5.A)	RAM memory	Plaintext	Manual	Electronic	
API Output via TOEPP path	RAM memory	Applications (App per IG 9.5.A)	Plaintext	Manual	Electronic	

Table 14: SSP Input-Output Methods

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Zeroization Command	CO runs zeroization service	Zeroization service will erase all SSPs used by the module	API function 'clearSensitiveData' zeroizes all SSPs
Power down	Power down the tested platform	Powering down the tested platform will zeroize all SSPs used by the module	Power down

Table 15: SSP Zeroization Methods

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DRBG Entropy Input	Used to seed the DRBG	384 bits - at least 256 bits	Entropy Input - CSP			Random Number Generation
CTR_DRBG Seed	Used in CTR_DRBG Generation	256 bits - 256 bits	DRBG Seed - CSP			Random Number Generation
CTR_DRBG V	Used in CTR_DRBG Generation	256 bits - 256 bits	DRBG Internal State V value - CSP			Random Number Generation
CTR_DRBG Key	Used in DRBG Generation	256 bits - 256 bits	DRBG Key - CSP			Random Number Generation
Hash_DRBG Seed	Used in DRBG Generation	256 bits - 256 bits	DRBG Seed - CSP			Random Number Generation
Hash_DRBG V	Used in DRBG Generation	256 bits - 256 bits	DRBG Internal State V value - CSP			Random Number Generation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Hash_DRBG C	Used in DRBG Generation	256 bits - 256 bits	DRBG Key - CSP			Random Number Generation
HMAC_DRBG Seed	Used in DRBG Generation	256 bits - 256 bits	DRBG Seed - CSP			Random Number Generation
HMAC_DRBG V	Used in DRBG Generation	256 bits - 256 bits	DRBG Internal State V value - CSP			Random Number Generation
HMAC_DRBG Key	Used in DRBG Generation	256 bits - 256 bits	DRBG Key - CSP			Random Number Generation
Diffie-Hellman Private Key	Used to derive the Diffie-Hellman Shared Secret	MODP-2048, MODP-3072, MODP-4096 - 112-152 bits	Private Key - CSP	KAS-FFC Keypair Generation		Shared Secret Calculation (KAS-FFC-SSC)
Diffie-Hellman Public Key	Used to derive the Diffie-Hellman Shared Secret	MODP-2048, MODP-3072, MODP-4096 - 112-152 bits	Public Key - PSP		KAS-FFC Keypair Generation	

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Diffie-Hellman Shared Secret	Used to derive other session keys	MODP-2048, MODP-3072, MODP-4096 - 112-152 bits	Shared Secret - CSP		Shared Secret Calculation (KAS-FFC-SSC)	
EC Diffie-Hellman Private Key	Used to derive the EC Diffie-Hellman Shared Secret	Curves: 256, 384, 521 bits - 128 to 256 bits	Private Key - CSP	KAS-ECC Keypair Generation		Shared Secret Calculation (KAS-ECC-SSC)
EC Diffie-Hellman Public Key	Used to derive EC Diffie-Hellman Shared Secret	Curves: 256, 384, 521 bits - 128-256 bits	Public Key - PSP		KAS-ECC Keypair Generation	
EC Diffie-Hellman Shared Secret	Used to derive other session keys	Curves: 256, 384, 521 bits - 128 to 256 bits	Shared Secret - CSP		Shared Secret Calculation (KAS-ECC-SSC)	

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DSA SGK	Used for DSA signature generation	2048, 3072 bits - 112 or 128 bits	Private Key - CSP	Asymmetric Key Generation/Verification		Digital Signature Generation
DSA SVK	Used for DSA signature verification	1024, 2048, 3072 bits - 128 or 128 bits	Public Key - PSP		Asymmetric Key Generation/Verification	Digital Signature Verification (legacy) Digital Signature Verification
ECDSA SGK	Used for ECDSA signature generation	Curves: 256, 384, 521 - 128 to 256 bits	Private Key - CSP	Asymmetric Key Generation/Verification		Digital Signature Generation
ECDSA SVK	Used for ECDSA signature verification	Curves: 256, 384, 521 - 128 to 256 bits	Public Key - PSP		Asymmetric Key Generation/Verification	Digital Signature Verification
RSA SGK	Used for RSA signature generation	Modulus: 2048, 3072 and 4096 bits - 112-	Private Key - CSP	Asymmetric Key Generation/Verification		Digital Signature Generation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		152 bits				
RSA SVK	Used for RSA signature verification	Modulus: 2048, 3072 and 4096 bits - 112-152 bits	Public Key - PSP		Asymmetric Key Generation/Verification	Digital Signature Verification
KAS-IFC Private Key	Used for KAS-IFC Shared Secret derivation	Modulus: 2048, 3072 and 4096 bits - 112-152 bits	Private Key - CSP	Asymmetric Key Generation/Verification		Shared Secret Calculation (KAS-IFC-SSC)
KAS-IFC Public Key	Used for KAS-IFC Shared Secret derivation	Modulus: 2048, 3072 and 4096 bits - 112-152 bits	Public Key - PSP		Asymmetric Key Generation/Verification	Shared Secret Calculation (KAS-IFC-SSC)
KAS-IFC-SSC Shared Secret	Used for KAS-IFC-SSC derivation	Modulus: 2048, 3072 and 4096 bits - 112-152 bits	CSP - CSP		Shared Secret Calculation (KAS-IFC-SSC)	

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
RSADP Primitive Private Key	Used for RSADP primitive component	2048 bits - 112 bits	Private Key - CSP	Asymmetric Key Generation/Verification		RSADP Primitive
RSADP Primitive Public Key	Used for RSADP primitive component	2048 bits - 112 bits	Public Key - PSP		Asymmetric Key Generation/Verification	RSADP Primitive
AES EDK	Used for AES Encryption/Decryption	128-256 bits - 128-256 bits	AES Encryption/Decryption Key - CSP	Cryptographic Key Generation (CKG)		Unauthenticated Symmetric Encryption and Decryption
AES-XTS key	Used for AES-XTS encryption and decryption	128-256 bits - 128-256 bits	AES-XTS symmetric key - CSP	AES-XTS KeyGen (CKG)		Unauthenticated Symmetric Encryption and Decryption
AES key wrapping key	Used for AES KW/KWP key wrapping	128-256 bits - 128-256 bits	Symmetric Key - CSP	Cryptographic Key Generation (CKG)		Key Wrap/Unwrap
AES-CCM key	Used for AES-CCM authenticated encryption/decryption	128-256 bits - 128-256 bits	AES-CCM Key - CSP	Cryptographic Key Generation (CKG)		Authenticated Symmetric Encryption/Decryption
AES-GCM Key	Used for AES-CCM authenticated encryption/decryption	128-256 bits - 128-256 bits	AES-GCM Key - CSP	Cryptographic Key Generation (CKG)		Authenticated Symmetric Encryption/Decryption
AES-CMAC key	Used for AES-CMAC Encryption/Decryption	128-256 bits - 128-256 bits	MAC Key - CSP	Cryptographic Key Generation (CKG)		MAC Generation/Verification

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
HMAC Key	Used for HMAC generation	At least 160 bits - At least 112 bits	HMAC Key - CSP			MAC Generation/Verification
KBKDF Key Derivation Key	Used for KBKDF Derived Key	TBD - TBD	TBD - CSP			Key Derivation
KBKDF Derived Key	Used for KBKDF calling	TBD - TBD	TBD - CSP		Key Derivation	
OneStep KDF Key Derivation Key	Used for OneStep KDF Derived Key derivation	TBD - TBD	Keying Material - CSP			Key Derivation
OneStep KDF Derived Key	Used for OneStep KDF calling function	TBD - TBD	Keying material - CSP		Key Derivation	
PBKDF Password	Used for PBKDF Derived Key derivation	TBD - TBD	Authentication data - CSP			Key Derivation
PBKDF Derived Key	Used for PBKDF calling function	TBD - TBD	KDF Key - CSP		Key Derivation	
TLS Master Secret	Used to derive TLS session keys	TBD - TBD	Keying Material - CSP		Key Derivation	Key Derivation
TLS Session Encryption Key	Used for TLS session protection	128-256 bits - 128-256 bits	Symmetric session encryption key - CSP		Key Derivation	TLS Session Encrypt/Decrypt
TLS Session Integrity Key	Used to protect TLS data integrity	at least 112 bits - at	Integrity key - CSP		Key Derivation	TLS Session Encrypt/Decrypt TLS Session

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		least 112 bits				Authentication

Table 16: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
DRBG Entropy Input	API Input via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	CTR_DRBG Seed:Used With CTR_DRBG V:Used With CTR_DRBG Key:Used With Hash_DRBG Seed:Used With Hash_DRBG V:Used With Hash_DRBG C:Used With HMAC_DRBG Seed:Used With HMAC_DRBG V:Used With HMAC_DRBG Key:Used With
CTR_DRBG Seed		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	DRBG Entropy Input:Used With CTR_DRBG V:Used With CTR_DRBG Key:Used With
CTR_DRBG V		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	DRBG Entropy Input:Used With CTR_DRBG Seed:Used With CTR_DRBG Key:Used With
CTR_DRBG Key		RAM memory:Plaintext	All SSPs are temporarily	Zeroization Command	DRBG Entropy Input:Used With CTR_DRBG

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
			stored. Storage duration is for the lifetime of the API call	Power down	Seed:Used With CTR_DRBG V:Used With
Hash_DRBG Seed		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	DRBG Entropy Input:Used With Hash_DRBG V:Used With Hash_DRBG C:Used With
Hash_DRBG V		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	DRBG Entropy Input:Used With Hash_DRBG Seed:Used With Hash_DRBG C:Used With
Hash_DRBG C		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	DRBG Entropy Input:Used With Hash_DRBG Seed:Used With Hash_DRBG V:Used With
HMAC_DRBG Seed		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	DRBG Entropy Input:Used With HMAC_DRBG V:Used With HMAC_DRBG Key:Used With
HMAC_DRBG V		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is	Zeroization Command Power down	DRBG Entropy Input:Used With HMAC_DRBG Seed:Used With HMAC_DRBG Key:Used With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
			for the lifetime of the API call		
HMAC_DRBG Key		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	DRBG Entropy Input:Used With HMAC_DRBG Seed:Used With HMAC_DRBG V:Used With
Diffie-Hellman Private Key		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	Diffie-Hellman Public Key:Paired With Diffie-Hellman Shared Secret:Used to establish
Diffie-Hellman Public Key	API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	Diffie-Hellman Private Key:Paired With
Diffie-Hellman Shared Secret		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	Diffie-Hellman Private Key:Derived From
EC Diffie-Hellman Private Key		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	EC Diffie-Hellman Public Key:Paired With EC Diffie-Hellman Shared Secret:Used to establish

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
EC Diffie-Hellman Public Key	API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	EC Diffie-Hellman Private Key:Paired With
EC Diffie-Hellman Shared Secret		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	EC Diffie-Hellman Private Key:Derived From
DSA SGK	API Input via TOEPP path API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	DSA SVK:Paired With
DSA SVK	API Input via TOEPP path API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	DSA SGK:Paired With
ECDSA SGK		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	ECDSA SVK:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
ECDSA SVK	API Input via TOEPP path API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	ECDSA SGK:Paired With
RSA SGK	API Input via TOEPP path API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	RSA SVK:Paired With
RSA SVK	API Input via TOEPP path API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	RSA SGK:Paired With
KAS-IFC Private Key	API Input via TOEPP path API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	KAS-IFC Public Key:Paired With KAS-IFC-SSC Shared Secret:Used to establish
KAS-IFC Public Key	API Input via TOEPP path API Output via	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the	Zeroization Command Power down	KAS-IFC Private Key:Paired With KAS-IFC-SSC Shared Secret:Used to establish

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
	TOEPP path		lifetime of the API call		
KAS-IFC-SSC Shared Secret			All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	RSA KAS-IFC Private Key:Derived From RSA KAS-IFC Public Key:Derived From
RSADP Primitive Private Key		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	RSADP Primitive Public Key:Paired With
RSADP Primitive Public Key		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	RSADP Primitive Private Key:Paired With
AES EDK	API Input via TOEPP path API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	
AES-XTS key	API Input via TOEPP path API Output via	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
	TOEPP path				
AES key wrapping key	API Input via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	
AES-CCM key	API Input via TOEPP path API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	
AES-GCM Key	API Input via TOEPP path API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	
AES-CMAC key	API Input via TOEPP path API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	
HMAC Key	API Input via TOEPP path API Output	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the	Zeroization Command Power down	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
	via TOEPP path		lifetime of the API call		
KBKDF Key Derivation Key	API Input via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	KBKDF Derived Key:Used to establish
KBKDF Derived Key	API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	KBKDF Key Derivation Key:Derived From
OneStep KDF Key Derivation Key	API Input via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	OneStep KDF Derived Key:Used to establish
OneStep KDF Derived Key	API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	OneStep KDF Key Derivation Key:Derived From
PBKDF Password	API Input via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	PBKDF Derived Key:Used to establish

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
PBKDF Derived Key	API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	PBKDF Password:Derived From
TLS Master Secret	API Input via TOEPP path API Output via TOEPP path	RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	TLS Session Encryption Key:Used to establish TLS Session Integrity Key:Used to establish
TLS Session Encryption Key		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	TLS Session Integrity Key:Used With
TLS Session Integrity Key		RAM memory:Plaintext	All SSPs are temporarily stored. Storage duration is for the lifetime of the API call	Zeroization Command Power down	TLS Session Encryption Key:Used With

Table 17: SSP Table 2

9.5 Transitions

- SHA-1: The module includes an implementation of SHA-1 for hashing and digital signature verification. This implementation will be non-Approved for all uses starting January 1, 2031. At this time, the user should move to SHA2, which is available in this module.

- FIPS186-4/186-5: As of February 5, 2024, the CMVP does not accept module submissions that implement DSA or RSA X9.31 in the approved mode, other than for signature verification which is approved for legacy use. Although this validation was submitted to the CMVP before February 3, 2024, the module only implements the DSA and RSA X9.31 functionality that remains approved for submissions after this transition

    - DSA primes and group generators used exclusively in a SP800-56Ar3-compliant scheme remain approved
    - DSA verification remains approved
    - RSA X9.31 verification remains approved

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA-1 (A2314)	HMAC-SHA-1	KAT	SW/FW Integrity	Module is in normal state	HMAC-SHA-1

Table 18: Pre-Operational Self-Tests

When the module is loaded or instantiated after being power-cycled or rebooted, the module runs pre-operational self-tests. The operating system is responsible for the initialization process and loading the module. The module is designed with a default entry point (DEP) that ensures automatic initiation of the self-tests when the module is loaded. Before the module provides any data output via the data output interface, it performs the pre-operational self-tests, ensuring all pass. A software integrity test is performed on the runtime image of the module with an HMAC-SHA-1 algorithm. Prior to the firmware integrity test, the module conducts an HMAC-SHA-1 Cryptographic Algorithm Self-test (CAST). If the CAST on the HMAC-SHA-1 is successful, the HMAC value of the runtime image is recalculated and compared with the stored HMAC value pre-computed at compilation time. During power-up, and following the successful pre-operational self-tests, the module executes the Conditional CASTs for all approved cryptographic algorithms implemented by the module.

The self-test success or failure messages, for example, Error: Signature RSA test failure or ECDH P-256 test failure, are logged and function as the self-test status indicator.

If any one of the self-tests fails, the module transitions into a `FIPS140State.FAILED` error state and outputs the error message via the module's status output interface, `SecurityException`. While the module is in the error state, all data through the data output interface and all cryptographic operations are disabled. The only method to recover from the error state is to power cycle the device. This results in the module being reloaded into memory and reperforming the pre-operational software integrity test and the Conditional CASTs. The module will only enter the operational state after successfully passing the pre-operational software integrity test and the Conditional CASTs.

Pre-operational self-tests are executed automatically when the module is loaded into memory. They can be re-run manually after the module has loaded, by calling the `ModuleConfig.runSelfTests()` API.

The pre-operational self-tests include the Software Integrity Test. The Software Integrity Test is comprised of an HMAC-SHA-1 verification of the files listed in `fips140/module.files`.

The cryptographic services of the module are disabled when the self-tests are running. When the self-tests are running, the following stands true:

- All cryptographic operations, if called, throw a `CryptoException`.
- The `CryptoModule.getState()` status output interface, if called, returns a state of `com.rsa.crypto.FIPS140State.UNDER_SELF_TEST`.

If any pre-operational self-test fails, all cryptographic services of the module are disabled. When the self-tests fail, the following stands true:

- All cryptographic operations, if called, throw a `CryptoException`.
- The `CryptoModule.getState()` status output interface, if called, returns a state of `com.rsa.crypto.FIPS140State.FAILED`.

If the pre-operational self-tests pass, the cryptographic services of the module are enabled, and the module can be used. The `CryptoModule.getState()` status output interface returns a state of `com.rsa.crypto.FIPS140State.OPERATIONAL`.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC Encrypt KAT (A2314)	256 bits	KAT	CAST	Module is in normal state	Encrypt	Initialization
AES-CBC Decrypt KAT (A2314)	256 bits	KAT	CAST	Module is in normal state	Decrypt	Power Up
AES-GCM Authenticated Encrypt KAT (A2314)	256 bits	KAT	CAST	Module is in normal state	Authenticated Encrypt	Initialization
AES-GCM Authenticated Decrypt KAT (A2314)	256 bits	KAT	CAST	Module is in normal state	Authenticated Decrypt	Initialization
CTR_DRBG Instantiate/Generate/ Reseed KAT (A2314)	AES-128	KAT	CAST	Module is in normal state	Instantiate/Generate/ Reseed KAT	Initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
Hash_DRBG Instantiate/Generate/ Reseed KAT (A2314)	SHA-1	KAT	CAST	Module is in normal state	Hash_DRBG Instantiate/Generate/ Reseed KAT	Initialization
HMAC_DRBG Instantiate/Generate/ Reseed KAT (A2314)	HMAC-SHA-1	CAST	CAST	Module is in normal state	HMAC_DRBG Instantiate/Generate/ Reseed KAT	Initialization
DSA SigGen (FIPS186-4) KAT (A2314)	2048-bit with SHA2-256	KAT	CAST	Module is in normal state	DSA SigGen KAT	Initialization
DSA SigVer (FIPS186-4) KAT (A2314)	2048-bit with SHA2-256	KAT	CAST	Module is in normal state	DSA SigVer KAT	Initialization
ECDSA SigGen (FIPS186-4) KAT (A2314)	P-256 curve with SHA2-256	KAT	CAST	Module is in normal state	ECDSA SigGen KAT	Initialization
ECDSA SigVer (FIPS186-4) KAT (A2314)	P-256 curve with SHA2-256	KAT	CAST	Module is in normal state	ECDSA SigVer KAT	Initialization
HMAC-SHA-1 KAT (A2314)	SHA-1	KAT	CAST	Module is in normal state	HMAC-SHA-1 KAT	Initialization
HMAC-SHA2-256 KAT (A2314)	SHA2-256	KAT	CAST	Module is in normal state	HMAC-SHA2-256 KAT	Initialization
HMAC-SHA2-384 KAT (A2314)	SHA2-384	KAT	CAST	Module is in normal state	HMAC-SHA2-384 KAT	Initialization
HMAC-SHA2-512 KAT (A2314)	SHA2-512	KAT	CAST	Module is in normal state	HMAC-SHA2-512 KAT	Initialization
HMAC-SHA3-512 KAT (A2314)	SHA3-512	KAT	CAST	Module is in normal state	HMAC-SHA3-512	Initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KAS-ECC-SSC Sp800- 56Ar3 KAT (A2314)	P-256 Curve	KAT	CAST	Module is in normal state	Primitive Z KAT	Initialization
KAS-FFC-SSC Sp800- 56Ar3 KAT (A2314)	MODP-2048	KAT	CAST	Module is in normal state	Primitive Z KAT	Initialization
KAS-IFC-SSC KAT (A2314)	N/A	KAT	CAST	Module is in normal state	Primitive Z KAT	Initialization
KDA OneStep Sp800-56Cr1 KAT (A2314)	N/A	KAT	CAST	Module is in normal state	N/A	Initialization
SP800-108 KBKDF KAT (A2314)	N/A	KAT	CAST	Module is in normal state	N/A	Initialization
PBKDF KAT (A2314)	N/A	KAT	CAST	Module is in normal state	N/A	Initialization
RSA SigGen (FIPS186-4) KAT (A2314)	2048 bit modulus with SHA2-256	KAT	CAST	Module is in normal state	RSA SigGen KAT	Initialization
RSA SigVer (FIPS186-4) KAT (A2314)	2048 bit modulus with SHA2-256	KAT	CAST	Module is in normal state	RSA SigVer KAT	Initialization
SHAKE-256 KAT (A2314)	N/A	N/A	Bypass	Module is in normal state	N/A	Initialization
TLS v1.2 KDF RFC7627 KAT (A2314)	N/A	KAT	CAST	Module is in normal state	N/A	Initialization
TLS v1.3 KDF KAT (A2314)	N/A	KAT	CAST	Module is in normal state	N/A	Initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
DSA KeyGen (FIPS186-4) PCT	2048 bits	N/A	PCT	Module is in normal state	N/A	Performs all required pair-wise consistency tests on the newly generated key pairs before the first operational use.
ECDSA KeyGen (FIPS186-4) PCT (A2314)	Curve P-256 with SHA2-256	PCT	PCT	Module is in normal state	N/A	Performs all required pair-wise consistency tests on the newly generated key pairs before the first operational use.
RSA KeyGen (FIPS186-4) PCT (A2314)	2048 bit Modulus	PCT	PCT	Module is in normal state	N/A	Performs all required pair-wise consistency tests on the newly generated key pairs before the first operational use.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KAS-ECC-SSC Sp800-56Ar3 PCT (A2314)	Curve P-256 with SHA2-256	PCT	PCT	Module is in normal state	N/A	Performs all required pair-wise consistency tests on the newly generated key pairs before the first operational use.
KAS-FFC-SSC Sp800-56Ar3 PCT (A2314)	MODP-2048	PCT	PCT	Module is in normal state	N/A	Performs all required pair-wise consistency tests on the newly generated key pairs before the first operational use.

Table 19: Conditional Self-Tests

The module generates RSA, ECDSA, KAS-ECC, and KAS-FFC asymmetric keys and performs all required pair-wise consistency tests on the newly generated key pairs as detailed in the "Pair-wise consistency tests" section below. If the Pair-wise Consistency conditional test fails, the module throws a `SecurityException` and aborts the operation. A Pair-wise Consistency test failure does not disable the module.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA-1 (A2314)	KAT	SW/FW Integrity	Recommend 60 Days	Reboot

Table 20: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC Encrypt KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
AES-CBC Decrypt KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
AES-GCM Authenticated Encrypt KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
AES-GCM Authenticated Decrypt KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
CTR_DRBG Instantiate/Generate/Reseed KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
Hash_DRBG Instantiate/Generate/Reseed KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
HMAC_DRBG Instantiate/Generate/Reseed KAT (A2314)	CAST	CAST	Recommend 60 Days	Reboot
DSA SigGen (FIPS186-4) KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
DSA SigVer (FIPS186-4) KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
ECDSA SigGen (FIPS186-4) KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
ECDSA SigVer (FIPS186-4) KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
HMAC-SHA-1 KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
HMAC-SHA2-256 KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
HMAC-SHA2-384 KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
HMAC-SHA2-512 KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
HMAC-SHA3-512 KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
KAS-ECC-SSC Sp800-56Ar3 KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
KAS-FFC-SSC Sp800-56Ar3 KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
KAS-IFC-SSC KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
KDA OneStep Sp800-56Cr1 KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
SP800-108 KBKDF KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
PBKDF KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
RSA SigGen (FIPS186-4) KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
RSA SigVer (FIPS186-4) KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
SHAKE-256 KAT (A2314)	N/A	Bypass	Recommend 60 Days	Reboot
TLS v1.2 KDF RFC7627 KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
TLS v1.3 KDF KAT (A2314)	KAT	CAST	Recommend 60 Days	Reboot
DSA KeyGen (FIPS186-4) PCT	N/A	PCT	Recommend 60 Days	Reboot
ECDSA KeyGen (FIPS186-4) PCT (A2314)	PCT	PCT	Recommend 60 Days	Reboot
RSA KeyGen (FIPS186-4) PCT (A2314)	PCT	PCT	Recommend 60 Days	Reboot
KAS-ECC-SSC Sp800-56Ar3 PCT (A2314)	PCT	PCT	Recommend 60 Days	Reboot
KAS-FFC-SSC Sp800-56Ar3 PCT (A2314)	PCT	PCT	Recommend 60 Days	Reboot

Table 21: Conditional Periodic Information

The module performs on-demand self-tests initiated by the operator, by power-cycling or rebooting the tested platform. The full suite of self-tests is then executed. The same procedure may be employed by the operator to perform periodic self-tests. In addition, it is recommended for the Crypto Officer to perform the periodic tests a minimum of once every 60 days to ensure all components are functioning correctly.

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Error State	If self-test tests fail, the module is put into an error state	Self-test failure	Reboot the module	System Halt

Table 22: Error States

If any of the above-mentioned self-tests fail, the Module reports the cause of the error and enters a `FIPS140State.FAILED` error state (there is only one error state). In the Error State, no cryptographic services are provided, and data output is prohibited. The only method to recover from the error state is to power-cycle or reboot to reload the Module and perform the self-tests, including the pre-operational software integrity test and the conditional CASTs. The module will only enter the operational state after successfully passing the pre-operational software integrity test and the conditional CASTs.

Note: `FIPS140State.FAILED` is the only error state.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

The module is installed by adding jcmFIPS-7.0.jar to the application's class path.

The module is started by starting the application that references it. The module uses JDK services to perform the module startup when the application loads it.

When loading the module, the `com.rsa.crypto.jcm.ModuleLoader.load()` method extracts arguments from the `com.rsa.cryptoj.jcm.JavaModuleProperties` class, which is created using the `com.rsa.cryptoj.jcm.CryptoJModulePropertiesFactory` class.

The following arguments are extracted:

- The module jar file.
- The security level, specified as the constant `ModuleConfig.LEVEL_1` which should have the value of 1.
- An optional `SelfTestEventListener` argument used for logging power-up self-test events.
- An optional `java.util.concurrent.ExecutorService` argument used for running the power-up self-tests.
- An optional file to be used for reading and writing the status of the algorithm power-up self-tests.

Using the specified security level ensures that the module is loaded for use in an approved mode.

Loading the module runs the integrity tests that must be completed successfully before any cryptographic services are made available by the module. This ensures that the application has made no modification to the module as part of its development or installation. For more information about the Integrity Tests, see Software/Firmware Security.

The module starts in an approved mode and in the Crypto Officer Role by default. Otherwise, to assume a role once the module is operational, construct a `FIPS140Context` object for the desired role using the `FIPS140Context.getFIPS140Context(int mode, int role)` method.

- The mode argument must be the value `FIPS140Context.MODE_FIPS140`. To retrieve the current mode of operation, call `FIPS140Context.getMode()`.
- The available role value is the constant `FIPS140Context.ROLE_CRYPTO_OFFICER`.

No role authentication is required to operate the module in Security Level 1 mode.

This object can then be used to perform cryptographic operations using the module.

The only permitted maintenance operation is to add a signature to the jar file by re-signing with an application certificate. Otherwise, application writers should not attempt to modify the module jar file as the module will refuse to load or perform cryptographic operations.

11.2 Administrator Guidance

For details of the administrative functions, security parameters, and logical interfaces available to the Crypto Officer refer to Section 4.2 - Roles.

11.3 Non-Administrator Guidance

Not applicable for this module.

12 Mitigation of Other Attacks

12.1 Attack List

RSA, EC, and DSA key operations implement blinding by default, a reversible way of modifying the input data, to make the operation immune to timing attacks. Blinding has no effect on the algorithm other than to mitigate attacks on the algorithm. For more information, see Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems.
RSA, EC, and DSA blinding is implemented through blinding modes, for which the following options are available:

- Blinding mode off.
- Blinding mode with no update, where the blinding value is squared for each operation.

12.2 Mitigation Effectiveness

This mitigation is enabled by default. For optimum security, it should not be disabled. RSA signing operations implement a verification step after private key operations. This verification step is in place to prevent potential faults in optimized Chinese Remainder Theorem (CRT) implementations. It has no effect on the signature algorithm. For more information, see Modulus Fault Attacks Against RSA-CRT Signatures and On the Importance of Eliminating Errors in Cryptographic Computations.

This mitigation is enabled by default. For optimum security, it should not be disabled.
RSA PKCS #1 v1.5 encryption padding operations are implemented in constant time in order to make the operation immune to timing attacks. For more information, see Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1.

Time invariant comparisons are also used for HMAC and RSA verify operations. For this mitigation, constant time padding is built-in and cannot be disabled.