# ISO/IEC 19790 and FIPS 140-3 Non-Proprietary

# Security Policy

# for

# Dell™ BSAFE™ Crypto Module for Java 7.0

Software Version: 7.0

Last Updated: November 14, 2024, Version 1.2

Table of Content



List of Figures



List of Tables

# 1 General

This is Dell Australia Pty Limited non-proprietary security policy for the Dell™ BSAFE™ Crypto Module for Java 7.0 (hereinafter referred to as the Module or JCM) with software version 7.0. The following details how this Module meets the security requirements of FIPS 140-3, SP800-140, 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.

| ISO/IEC 24759:2017 Section 6 | ISO/IEC 24759:2017 and FIPS 140-3 Section Title | 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[1] | 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 |

**Table 1 Security Levels**

[1]The module relies on the physical security provided by the host on which it runs.

The Module has an overall security level of 1.

# 2 Cryptographic module specification

The Module is a multi-chip standalone software module intended to be used as part of a software system, providing cryptographic services to that system. The module's operational environment is non-modifiable. The module is provided as a Java Archive (jar) file. It 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()`.

The FIPS 140-3 validation certificate can be located on the NIST Cryptographic Module Validation Program (CMVP) page using the module name and version reported.

The Module has been tested on the following Operational Environments:

| # | Operating System | Hardware Platform | Processor | PAA/Acceleration |
|---|---|---|---|---|
| 1 | SUSE® Linux Enterprise Server 15 SP3 (64-bit) with OpenJDK 11 | Dell PowerEdge™ R6525 | AMD EPYC™ 7513 | N/A |
| 2 | SUSE® Linux Enterprise Server 15 SP3 (64-bit) with OpenJDK 8 | Dell PowerEdge™ R6525 | AMD EPYC™ 7513 | N/A |
| 3 | Windows Server® 2019 (64-bit) with Oracle® JRE 8 | Dell PowerEdge™ R6525 | AMD EPYC™ 7513 | N/A |
| 4 | Windows Server® 2016 (64-bit) with Oracle® JRE 8 | Dell PowerEdge™ T130 | Intel® Xeon® CPU E3-1230 | N/A |

**Table 2 Tested Operational Environment**

In addition to the platforms listed in Table 2, Dell has also tested the module on the following platforms and claims vendor affirmation on them:

| # | Operating System | Hardware Platform |
|---|---|---|
| 1 | Apple® MacOS® 10.15 (x86_64) with Oracle JDK 11 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 2 | Apple MacOS 10.15 (x86_64) with Oracle JDK 8 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 3 | Canonical® Ubuntu® 16.04 (x86) with OpenJDK 8 (32-bit) | Generic Hardware Platform with Intel x86 (32-bit) |
| 4 | Canonical Ubuntu 16.04 (x86) with OpenJDK 8 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 5 | CentOS™ 7.9 (x86_64) with OpenJDK 8 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 6 | Dell PowerProtect™ Data Domain™ OS (x86_64) with Oracle JDK 8 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 7 | Dell PowerStoreOS™ 4.0 (x86_64) with OpenJDK 11 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 8 | FreeBSD® Foundation 12 (x86_64) with OpenJDK 8 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |

| | | |
|---|---|---|
| 9 | Hewlett Packard Enterprise HP-UX 11.31 with HP JDK 8 (64-bit) | Generic Hardware Platform with Itanium® 2 |
| 10 | IBM AIX® 7.2 with IBM JDK 8 (64-bit) | Generic Hardware Platform with PowerPC® (64-bit) |
| 11 | Microsoft® Windows 10 Enterprise (x86_64) with Oracle JDK 11 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 12 | Microsoft Windows 10 Enterprise (x86_64) with Oracle JDK 8 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 13 | Microsoft Windows 10 Enterprise (x86_64) with Oracle JDK 7 (32-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 14 | Microsoft Windows Server 2019 (x86_64) with Oracle JDK 11 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 15 | Microsoft Windows Server 2016 (x86_64) with Oracle JDK 11 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 16 | Oracle Solaris® 11.4 with Oracle JDK 11 (64-bit) | Generic Hardware Platform with SPARC® v9 |
| 17 | Oracle Solaris 11.4 with Oracle JDK 8 (64-bit) | Generic Hardware Platform with SPARC v9 |
| 18 | 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) |
| 19 | 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) |
| 20 | 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) |
| 21 | 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) |
| 22 | Oracle Linux 7 64-bit on Oracle A Series Servers with Oracle JDK 8 (64-bit) | Generic Hardware Platform with ARMv8 (64-bit) |
| 23 | Oracle Linux 7 64-bit on Oracle A Series Servers with Oracle JDK 11 (64-bit) | Generic Hardware Platform with ARMv8 (64-bit) |
| 24 | 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) |
| 25 | 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) |

| | | |
|---|---|---|
| 26 | 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) |
| 27 | 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) |
| 28 | Oracle Linux 8 64-bit on Oracle A Series Servers with Oracle JDK 8 (64-bit) | Generic Hardware Platform with ARMv8 (64-bit) |
| 29 | Oracle Linux 8 64-bit on Oracle A Series Servers with Oracle JDK 11 (64-bit) | Generic Hardware Platform with ARMv8 (64-bit) |
| 30 | Red Hat® Enterprise Linux 8.6 (x86_64) with Oracle JDK 11 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 31 | Red Hat Enterprise Linux 8.6 (x86_64) with Oracle JDK 8 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 32 | Red Hat Enterprise Linux 7.9 (x86_64) with Oracle JDK 11 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 33 | Red Hat Enterprise Linux 7.9 (x86_64) with Oracle JDK 8 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 34 | SUSE Linux Enterprise Server 15 SP4 (x86_64) with OpenJDK 17 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 35 | SUSE Linux Enterprise Server 15 SP4 (x86_64) with OpenJDK 11 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 36 | SUSE Linux Enterprise Server 15 SP4 (x86_64) with OpenJDK 8 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 37 | SUSE Linux Enterprise Server 15 SP2 (x86_64) with OpenJDK 11 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 38 | SUSE Linux Enterprise Server 15 SP2 (x86_64) with OpenJDK 8 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 39 | SUSE Linux Enterprise Server 12 SP5 (x86_64) with IBM JDK 8 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 40 | SUSE Linux Enterprise Server 12 SP5 (x86_64) with IBM JDK 7 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 41 | SUSE Linux Enterprise Server 12 SP5 (x86_64) with OpenJDK 7 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 42 | SUSE Linux Enterprise Server 12 SP5 (x86_64) with Oracle JDK 11 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |

| | | |
|---|---|---|
| 43 | SUSE Linux Enterprise Server 12 SP5 (x86_64) with Oracle JDK 8 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |
| 44 | SUSE Linux Enterprise Server 12 SP5 (x86_64) with Oracle JDK 7 (64-bit) | Generic Hardware Platform with Intel x86_64 (64-bit) |

**Table 3 Vendor Affirmed Operational Environments**

The CMVP makes no statement about the correct operation of the module or the security strengths of the generated keys when ported to an operational environment which is not listed on the validation certificate.

**Mode of operation**

The Module supports both approved and non-approved modes of operation. It can operate 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. If any self-test fails, the cryptographic services of the module are disabled for both Approved and Non-Approved modes of operation. Section 4 provides details on the service indicator implemented by the Module.

Table 4 below 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.

There are some algorithm modes that were tested but not implemented by the Module. Only the algorithms, modes, and key sizes that are implemented by the Module are shown in this table. 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 Module 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.

| CAVP Cert | Algorithm and Standard | Mode/Method | Description / Key Size(s) / Key Strength(s) | Use/Function |
|---|---|---|---|---|
| A2314 | AES<br>[FIPS 197; | CBC | Key Length: 128, 192, and 256 bits | Symmetric encryption and decryption |

| CAVP Cert | Algorithm and Standard | Mode/Method | Description / Key Size(s) / Key Strength(s) | Use/Function |
|---|---|---|---|---|
| | SP800-38A] | | | |
| A2314 | AES [FIPS 197; SP800-38A, addendum] | CBC-CS1 | Key Length: 128, 192, and 256 bits | Symmetric encryption and decryption |
| A2314 | AES [FIPS 197; SP800-38A, addendum] | CBC-CS2 | Key Length: 128, 192, and 256 bits | Symmetric encryption and decryption |
| A2314 | AES [FIPS 197; SP800-38A, addendum] | CBC-CS3 | Key Length: 128, 192, and 256 bits | Symmetric encryption and decryption |
| A2314 | AES [FIPS 197; SP800-38C] | CCM | Key Length: 128, 192, and 256 bits | Authenticated encryption and decryption |
| A2314 | AES [FIPS 197; SP800-38A, addendum] | CFB128 | Key Length: 128, 192, and 256 bits | Symmetric encryption and decryption |
| A2314 | AES [FIPS 197; SP800-38B] | CMAC | Key Length: 128, 192, and 256 bits | Message authentication |
| A2314 | AES [FIPS 197; SP800-38A] | CTR | Key Length: 128, 192, and 256 bits | Symmetric encryption and decryption |
| A2314 | AES [FIPS 197; SP800-38A] | ECB | Key Length: 128, 192, and 256 bits | Symmetric encryption and decryption |
| A2314 | AES [FIPS 197; SP800-38D] | GCM | Key Length: 128, 192, and 256 bits | Authenticated encryption and decryption |
| A2314 | AES [FIPS 197; SP800-38F] | KW, KWP | Key Length: 128, 192, and 256 bits<br><br>Key establishment methodology provides between 128 and 256 bits of encryption strength | Key wrapping and unwrapping |
| A2314 | AES [FIPS 197; SP800-38A] | OFB | Key Length: 128, 192, and 256 bits | Symmetric encryption and decryption |
| A2314 | AES [FIPS 197; SP800-38E] | XTS | Key Length: 128 and 256 bits | Symmetric encryption and decryption |
| A2314 | CTR_DRBG [SP800-90Arev1] | AES-128<br>AES-196<br>AES-256<br>Derivation Function Enabled;<br>Prediction Resistance: Yes, No | N/A | Random bit generation |
| A2314 | DSA [FIPS 186-4] | DSA KeyGen:<br>-N: 224/256<br>-2048/3072 Modulus | L: 2048/3072 bits | DSA keypair generation |
| A2314 | DSA [FIPS 186-4] | DSA PQGGen:<br>-P/Q Generation Methods: Probable<br>-G Generation Methods: Unverifiable<br>-N: 224/256<br>-2048/3072 Modulus with | L: 2048/3072 bits | DSA PQG generation |

| CAVP Cert | Algorithm and Standard | Mode/Method | Description / Key Size(s) / Key Strength(s) | Use/Function |
|---|---|---|---|---|
| | | SHA2-256, SHA2-384, SHA2-512 | | |
| A2314 | DSA<br>[FIPS 186-4] | DSA PQGVer:<br>-P/Q Generation Methods: Probable<br>-G Generation Methods: Unverifiable<br>-N: 160/224/256<br>-1024/2048/3072 Modulus with SHA1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 | L: 1024/2048/3072 bits | DSA PQG verification |
| A2314 | DSA<br>[FIPS 186-4] | DSA SigGen:<br>-N: 224/256<br>-2048/3072 Modulus with SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 | L: 2048/3072 bits | DSA signature generation |
| A2314 | DSA<br>[FIPS 186-4] | DSA SigVer:<br>-N: 160/224/256<br>-1024/2048/3072 Modulus with SHA1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 | L: 1024/2048/3072 bits | DSA signature verification<br><br>Please note that DSA 1024 bits are only used for signature verification |
| A2314 | ECDSA<br>[FIPS 186-4] | ECDSA KeyGen | Curves: B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 | ECDSA keypair generation |
| A2314 | ECDSA<br>[FIPS 186-4] | ECDSA KeyVer | Curves: 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 | ECDSA keypair verification |
| A2314 | ECDSA<br>[FIPS 186-4] | ECDSA SigGen | Curves: B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 | ECDSA signature generation |
| A2314 | ECDSA<br>[FIPS 186-4] | ECDSA SigVer | Curves: 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 | ECDSA signature verification |
| A2314 | HASH_DRBG<br>[SP800-90Arev1] | SHA-1<br>SHA2-224<br>SHA2-256<br>SHA2-384<br>SHA2-512<br>SHA2-512/224<br>SHA2-512/256<br>Prediction Resistance: Yes<br>No | N/A | Random bit generation |
| A2314 | HMAC_DRBG<br>[SP800-90Arev1] | HMAC-SHA-1<br>HMAC-SHA2-224<br>HMAC-SHA2-256<br>HMAC-SHA2-384<br>HMAC-SHA2-512<br>HMAC-SHA2-512/224<br>HMAC-SHA2-512/256<br>Prediction Resistance: Yes, No | N/A | Random bit generation |

| CAVP Cert | Algorithm and Standard | Mode/Method | Description / Key Size(s) / Key Strength(s) | Use/Function |
|---|---|---|---|---|
| A2314 | HMAC<br>[FIPS 198-1] | HMAC-SHA-1 | Key Length: 112 bits or greater | Message authentication |
| A2314 | HMAC<br>[FIPS 198-1] | HMAC-SHA2-224 | Key Length: 112 bits or greater | Message authentication |
| A2314 | HMAC<br>[FIPS 198-1] | HMAC-SHA2-256 | Key Length: 112 bits or greater | Message authentication |
| A2314 | HMAC<br>[FIPS 198-1] | HMAC-SHA2-384 | Key Length: 112 bits or greater | Message authentication |
| A2314 | HMAC<br>[FIPS 198-1] | HMAC-SHA2-512 | Key Length: 112 bits or greater | Message authentication |
| A2314 | HMAC<br>[FIPS 198-1] | HMAC-SHA2-512/224 | Key Length: 112 bits or greater | Message authentication |
| A2314 | HMAC<br>[FIPS 198-1] | HMAC-SHA2-512/256 | Key Length: 112 bits or greater | Message authentication |
| A2314 | HMAC<br>[FIPS 198-1] | HMAC-SHA3-224 | Key Length: 112 bits or greater | Message authentication |
| A2314 | HMAC<br>[FIPS 198-1] | HMAC-SHA3-256 | Key Length: 112 bits or greater | Message authentication |
| A2314 | HMAC<br>[FIPS 198-1] | HMAC-SHA3-384 | Key Length: 112 bits or greater | Message authentication |
| A2314 | HMAC<br>[FIPS 198-1] | HMAC-SHA3-512 | Key Length: 112 bits or greater | Message authentication |
| A2314 | KAS-ECC CDH Component (CVL)<br>[SP800-56Arev3] | KAS-ECC CDH-Component:<br>Function: Full Public Key Validation, Key Pair Generation, Partial Public Key Validation | Curves: B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521<br><br>Key establishment methodology provides between 128 and 256 bits of encryption strength | Key agreement primitive |
| A2314 | KAS-ECC-SSC (CVL)<br>[SP800-56Arev3] | KAS-ECC-SSC: Scheme:<br>ephemeralUnified:<br>KAS Role: initiator, responder<br>staticUnified:<br>KAS Role: initiator, responder | Curves: B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521<br>Key establishment methodology provides between 128 and 256 bits of encryption strength | Key agreement primitive |
| A2314 | KAS-FFC-SSC (CVL)<br>[SP800-56Arev3] | KAS-FFC-SSC:<br>Scheme:<br>dhEphem:<br>KAS Role: initiator, responder<br>dhOneFlow:<br>KAS Role: initiator, responder<br>dhStatic:<br>KAS Role: initiator, responder | Domain Parameter Generation Methods: FB, FC, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192<br><br>Key establishment methodology provides between 112 and 200 bits of encryption strength | Key agreement primitive |
| A2314 | KAS-IFC-SSC (CVL)<br>[SP800-56Brev2] | KAS-IFC-SSC<br>Scheme:<br>KAS1:<br>KAS Role: initiator, responder | Modulus: 2048, 3072, 4096, 6144, 8192 | Key agreement primitive |
| A2314 | KAS KC (CVL)<br>[SP800-56] | Key Confirmation Directions: Bilateral, Unilateral<br>KAS Role: Initiator, Responder | N/A | Key agreement primitive |

| CAVP Cert | Algorithm and Standard | Mode/Method | Description / Key Size(s) / Key Strength(s) | Use/Function |
|---|---|---|---|---|
| | | Key Confirmation Roles: Provider, Recipient<br>Key Confirmation MAC Methods:<br>HMAC-SHA-1<br>HMAC-SHA2-224<br>HMAC-SHA2-256<br>HMAC-SHA2-384<br>HMAC-SHA2-512<br>HMAC-SHA2-512/224<br>HMAC-SHA2-512/256<br>HMAC-SHA3-224<br>HMAC-SHA3-256<br>HMAC-SHA3-384<br>HMAC-SHA3-512 | | |
| A2314 | KDA<br>[SP800-56Crev1] | OneStep:<br>SHA-1<br>SHA2-224<br>SHA2-256<br>SHA2-384<br>SHA2-512<br>SHA2-512/224<br>SHA2-512/256<br>SHA3-224<br>SHA3-256<br>SHA3-384<br>SHA3-512 | N/A | Key agreement primitive |
| A2314 | KDF<br>[SP800-108] | KDF Mode: Feedback<br>MAC Mode:<br>HMAC-SHA-1<br>HMAC-SHA2-224<br>HMAC-SHA2-256<br>HMAC-SHA2-384<br>HMAC-SHA2-512<br>HMAC-SHA2-512/224<br>HMAC-SHA2-512/256 | N/A | Key derivation |
| A2314 | PBKDF [1]<br>[SP800-132] | | N/A | Key derivation |
| A2314 | RSA Decryption Primitive (CVL)<br>[SP800-56Crev2] | decryptionPrimitive | 2048 bit key size | Key transport primitive |
| A2314 | RSA<br>[FIPS 186-4] | RSA KeyGen:<br>-Mode: B.3.6<br>- 2048/3072/4096 Modulus | Modulus: 2048/3072/4096 bits | RSA keypair generation |
| A2314 | RSA<br>[FIPS 186-4] | RSA SigGen:<br>-Mode: ANSI X9.31<br>-2048/3072/4096 Modulus with SHA2-224/ SHA2-256/ SHA2-384/ SHA2-512/ SHA2-512-224/ SHA2-512-256;<br>-Mode: PKCS 1.5<br>-2048/3072/4096 Modulus with SHA2-224/ SHA2-256/ SHA2-384/ SHA2-512/ SHA2-512-224/ SHA2-512-256;<br>-Mode: PKCSPSS<br>-2048/3072/4096 Modulus with SHA2-224/ SHA2-256/ SHA2-384/ SHA2-512/ SHA2-512-224/ SHA2-512- | Modulus: 2048/3072/4096 bits | RSA signature generation |

| CAVP Cert | Algorithm and Standard | Mode/Method | Description / Key Size(s) / Key Strength(s) | Use/Function |
|---|---|---|---|---|
| | | 256; | | |
| A2314 | RSA<br>[FIPS 186-4] | RSA SigVer:<br>-Mode: ANSI X9.31<br>-2048/3072/4096 Modulus with SHA-1, SHA2-224/ SHA2-256/ SHA2-384/ SHA2-512/ SHA2-512-224/ SHA2-512-256;<br>-Mode: PKCS 1.5<br>-2048/3072/4096 Modulus with SHA-1, SHA2-224/ SHA2-256/ SHA2-384/ SHA2-512/ SHA2-512-224/ SHA2-512-256;<br>-Mode: PKCSPSS<br>-2048/3072/4096 Modulus with SHA-1, SHA2-224/ SHA2-256/ SHA2-384/ SHA2-512/ SHA2-512-224/ SHA2-512-256; | Modulus:<br>1024/2048/3072/4096 bits | RSA signature verification |
| A2314 | Safe Primes Key Generation<br>[SP800-56Arev3] | KeyGen for KAS-FFC-SSC | Safe Prime Groups:<br>ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 | KAS-FFC Keypair domain parameters generation |
| A2314 | Safe Primes Key Verification<br>[SP800-56Arev3] | KeyVer for KAS-FFC-SSC | Safe Prime Groups:<br>ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 | KAS-FFC Keypair domain parameters verification |
| A2314 | SHS<br>[FIPS 180-4] | SHA-1 | N/A | Message digest<br><br>Note: SHA-1 is not used for digital signature generation |
| A2314 | SHS<br>[FIPS 180-4] | SHA2-224 | N/A | Message digest |
| A2314 | SHS<br>[FIPS 180-4] | SHA2-256 | N/A | Message digest |
| A2314 | SHS<br>[FIPS 180-4] | SHA2-384 | N/A | Message digest |
| A2314 | SHS<br>[FIPS 180-4] | SHA2-512 | N/A | Message digest |
| A2314 | SHS<br>[FIPS 180-4] | SHA2-512/224 | N/A | Message digest |
| A2314 | SHS<br>[FIPS 180-4] | SHA2-512/256 | N/A | Message digest |
| A2314 | SHA3<br>[FIPS 202] | SHA3-224 | N/A | Message digest |
| A2314 | SHA3<br>[FIPS 202] | SHA3-256 | N/A | Message digest |
| A2314 | SHA3<br>[FIPS 202] | SHA3-384 | N/A | Message digest |
| A2314 | SHA3<br>[FIPS 202] | SHA3-512 | N/A | Message digest |
| A2314 | SHAKE<br>[FIPS 202] | SHAKE-128 | N/A | Message digest |

| CAVP Cert | Algorithm and Standard | Mode/Method | Description / Key Size(s) / Key Strength(s) | Use/Function |
|---|---|---|---|---|
| A2314 | SHAKE [FIPS 202] | SHAKE-256 | N/A | Message digest |
| A2314 | TLS v1.2 KDF RFC7627 [RFC7627] (CVL) | TLS v1.2 KDF RFC7627 | N/A | Key derivation |
| A2314 | TLS v1.3 KDF [RFC8446] (CVL) | TLS v1.3 KDF | N/A | Key derivation |
| Vendor Affirmed | Cryptographic Key Generation (CKG)[2] [SP800-133rev2] | N/A | N/A | Cryptographic Key Generation; SP800-133rev2 and IG D.H<br><br>Note: The cryptographic module performs Cryptographic Key Generation (CKG) for symmetric and asymmetric keys as per sections 5 and 6 in SP800-133rev2 (vendor affirmed). A seed (i.e., the random value) used in asymmetric key generation is a direct output from SP800-90Arev1 DRBG |

**Table 4 - Approved Algorithms**

[1]Password-based key derivation function 2 (PBKDF2). As defined in NIST Special Publication 800-132, PBKDF2 can be used in Approved mode when used with Approved symmetric key and message digest algorithms. For more information, see Crypto Officer Guidance

[2]The module supports cryptographic key generation as described in section 4 of SP800-133rev2 where V is a constant string of binary zeroes. The module also supports symmetric key generation as described in sections 6.1 and 6.2 of SP800-133rev2.

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

| Algorithm / Function | Use / Function |
|---|---|
| TDES in CBC, CFB64, ECB, OFB modes and CBC_CS1, CBC_CS2 or CBC_CS3 mode for CTS | Symmetric encryption / decryption |

**Table 5 - Non-Approved Algorithms Not Allowed in the Approved Mode of Operation**

As there are no non-Approved algorithms allowed in the approved mode of operation, the tables defined in SP800-140B for the following categories are missing from this document:

- Non-Approved Algorithms Allowed in Approved Mode of Operation
- Non-Approved Algorithms Allowed in Approved Mode of Operation with No Security Claimed

## Cryptographic boundary

The 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.

Data Input and Output are provided in the variables passed with the API calls. Status Output is provided through the returns and exceptions documented for each call. This is illustrated in Figure 1, which depicts the Module's cryptographic boundary and physical perimeter. 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.

Physical Perimeter

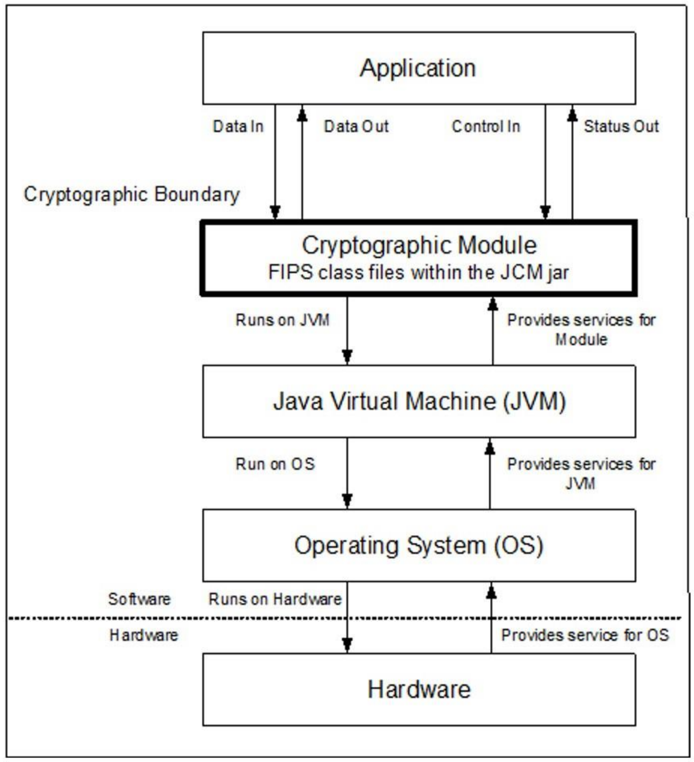


**Figure 1 Module's Block Diagram**

# 3 Cryptographic module 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) as follows:

| Physical Port | Logical Interface | Data that passes over port/interface |
|---|---|---|
| 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 6 Ports and Interfaces**

# 4 Roles, services, and authentication

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.

| Role | Service | Input | Output |
|---|---|---|---|
| Crypto Officer | Asymmetric Encryption | Plaintext | Ciphertext, Status |
| Crypto Officer | Asymmetric Decryption | Ciphertext | Plaintext, Status |
| Crypto Officer | Digital Signature Generation | Message Digest | Signature, Status |
| Crypto Officer | Digital Signature Verification | Message Digest, Signature | Verify Status, Status |
| Crypto Officer | Key Assurance | - | Validation Status, Status |
| Crypto Officer | Key Confirmation | MAC | Verify status |
| Crypto Officer | Key Deletion | - | - |
| Crypto Officer | Key Derivation | Secret | Key text, Status |
| Crypto Officer | Key Export | - | Key text, Status |
| Crypto Officer | Key Generation | - | Status |
| Crypto Officer | Key Import | Key text | Status |
| Crypto Officer | Key Parameter Generation | - | Status |
| Crypto Officer | Key Wrap | - | Wrapped Key text, Status |
| Crypto Officer | Key Unwrap | Wrapped key text | Status |
| Crypto Officer | Message Digest | Message | Message Digest, Status |
| Crypto Officer | MAC Generation | Secret, Message | MAC, Status |
| Crypto Officer | MAC Verification | Secret, Message, MAC | Verify Status, Status |
| Crypto Officer | Random Number Generation | - | Random bytes, Status |
| Crypto Officer | Self-test | - | Status |
| Crypto Officer | Symmetric Encryption | Plaintext | Ciphertext, Status |
| Crypto Officer | Symmetric Decryption | Ciphertext | Plaintext, Status |

**Table 7 Roles, Service Commands, Input, and Output**

The abbreviations of the access rights to keys and SSPs have the following interpretation:

**G = Generate**: The module generates or derives the SSP.
**R = Read**: The SSP is read from the module.
**W = Write**: The SSP is updated, imported, or written to the module.
**E = Execute**: The module uses the SSP in performing a cryptographic operation.
**Z = Zeroize**: The module zeroizes the SSP.
**N/A = Not applicable:** The service does not access any SSP during its operation.

Table 8 below lists all approved services that can be used in the approved mode of operation.

| Service | Description | Approved Security Functions | Keys and/or SSPs | Roles | Access rights to Keys and/or SSPs | Indicator |
|---|---|---|---|---|---|---|
| Asymmetric Encryption | Perform asymmetric encryption operation | RSA Encryption Primitive | RSA Public Key | CO | E | API call |
| Asymmetric Decryption | Perform asymmetric decryption operation | RSA Decryption Primitive | RSA Private Key | CO | E | API call |
| Digital Signature Generation | Perform digital signature generation | DSA PQGGen<br>DSA SigGen<br>ECDSA SigGen<br>RSA SigGen | DSA Private Key<br>ECDSA Private Key<br>RSA Private Key | CO | E | API call |
| Digital Signature Verification | Perform digital signature verification | DSA PQGVer<br>DSA SigVer<br>ECDSA SigVer<br>RSA SigVer | DSA Public Key<br>ECDSA Public Key<br>RSA Public Key | CO | E | API call |
| Key Assurance | Perform key assurance operation | N/A | DSA Public Key<br>DSA Private Key<br>ECDSA Public Key<br>ECDSA Private Key<br>EC Diffie-Hellman Private Key<br>EC Diffie-Hellman Public Key<br>Diffie-Hellman Public Key<br>Diffie-Hellman Private Key<br>RSA Public Key<br>RSA Private Key | CO | R | API call |
| Key Confirmation | Perform key confirmation operation | KAS KC | HMAC Key | CO | E | API call |
| Key Deletion | Perform key deletion operation | N/A | AES Key<br>DSA Public Key<br>DSA Private Key<br>ECDSA Public Key<br>ECDSA Private Key<br>CMAC Key<br>HMAC Key<br>RSA Public Key | CO | Z | API call |

| Service | Description | Approved Security Functions | Keys and/or SSPs | Roles | Access rights to Keys and/or SSPs | Indicator |
|---|---|---|---|---|---|---|
| | | | RSA Private Key | | | |
| Key Derivation | Perform key derivation operation | KDA OneStep<br>PBKDF<br>TLS v1.2 KDF RFC 7627<br>TLS v1.3 KDF | KBKDF Key Derivation Key<br>KBKDF Derived Key<br>OneStep KDF Key Derivation Key<br>OneStep KDF Derived Key<br>PBKDF Password<br>PBKDF Derived Key<br>TLS Master Secret<br>TLS Session Key<br>TLS Session Integrity Key | CO | G, R | API call |
| Key Export | Perform key export operation | N/A | AES Key<br>DSA Public Key<br>DSA Private Key<br>ECDSA Public Key<br>ECDSA Private Key<br>CMAC Key<br>HMAC Key<br>RSA Public Key<br>RSA Private Key | CO | R | API call |
| Key Generation | Perform key generation operation | DSA KeyGen<br>ECDSA KeyGen<br>KAS-ECC-SCC<br>KAS-FFC-SCC<br>RSA KeyGen | AES Key<br>DSA Public Key<br>DSA Private Key<br>ECDSA Public Key<br>ECDSA Private Key<br>EC Diffie-Hellman Private Key<br>EC Diffie-Hellman Public Key<br>Peer EC Diffie-Hellman Public Key<br>EC Diffie-Hellman Shared Secret | CO | G | API call |

| Service | Description | Approved Security Functions | Keys and/or SSPs | Roles | Access rights to Keys and/or SSPs | Indicator |
|---|---|---|---|---|---|---|
| | | | Diffie-Hellman Public Key<br>Diffie-Hellman Private Key<br>Diffie-Hellman Peer Diffie-Hellman Public Key<br>Shared Secret<br>RSA Public Key<br>RSA Private Key | | | |
| Key Import | Perform key import operation | | AES Key<br>DSA Public Key<br>DSA Private Key<br>ECDSA Public Key<br>ECDSA Private Key<br>CMAC Key<br>HMAC Key<br>RSA Public Key RSA Private Key | CO | W | API call |
| Key Parameter Generation | Perform key parameter generation operation | KAS-FFC-SCC | DSA Public Key<br>DSA Private Key<br>Diffie-Hellman Public Key<br>Diffie-Hellman Private Key<br>Diffie-Hellman Shared Secret | CO | G | API call |
| Key Unwrap | Perform key unwrap operation | AES-KW<br>AES-KWP | AES Key | CO | | API call |
| Key wrap | Perform key wrap operation | AES-KW<br>AES-KWP | AES Key<br>AES Key Wrap Key | CO | E | API call |
| Message Digest | Perform message digest operation | SHA-1<br>SHA2-224<br>SHA2-256<br>SHA2-384<br>SHA2-512<br>SHA2-512/224<br>SHA2-512/256<br>SHA3-224<br>SHA3-256<br>SHA3-384<br>SHA3-512<br>SHAKE-128<br>SHAKE-256 | | CO | | API call |
| MAC Generation | Perform MAC generation operation | CMAC-AES<br>HMAC-SHA-1 | CMAC Key<br>HMAC Key | CO | R | API call |

| Service | Description | Approved Security Functions | Keys and/or SSPs | Roles | Access rights to Keys and/or SSPs | Indicator |
|---|---|---|---|---|---|---|
| | | HMAC-SHA2-224<br>HMAC-SHA2-256<br>HMAC-SHA2-384<br>HMAC-SHA2-512<br>HMAC-SHA2-512/224<br>HMAC-SHA2-512/256<br>HMAC-SHA3-224<br>HMAC-SHA3-256<br>HMAC-SHA3-384<br>HMAC-SHA3-512 | | | | |
| MAC Verification | Perform MAC verification operation | CMAC-AES<br>HMAC-SHA-1<br>HMAC-SHA2-224<br>HMAC-SHA2-256<br>HMAC-SHA2-384<br>HMAC-SHA2-512<br>HMAC-SHA2-512/224<br>HMAC-SHA2-512/256<br>HMAC-SHA3-224<br>HMAC-SHA3-256<br>HMAC-SHA3-384<br>HMAC-SHA3-512 | CMAC Key<br>HMAC Key | CO | R | API call |
| Random Number Generation | Perform random number generation | CTR_DRBG<br>HASH_DRBG<br>HMAC_DRBG | DRBG Entropy Input<br>DRBG Seed<br>DRBG Internal State V value<br>DRBG Key | CO | | API call |
| Symmetric Encryption | Perform symmetric encryption operation | AES-CBC<br>AES-CBC-CS1<br>AES-CBC-CS2<br>AES-CBC-CS3<br>AES-CCM<br>AES-CFB<br>AES-CTR<br>AES-ECB<br>AES-GCM<br>AES-OFB<br>AES-XTS | AES Key<br>AES GCM IV | CO | E | API call |
| Symmetric Decryption | Perform symmetric decryption operation | AES-CBC<br>AES-CBC-CS1<br>AES-CBC-CS2<br>AES-CBC-CS3<br>AES-CCM<br>AES-CFB<br>AES-CTR<br>AES-ECB<br>AES-GCM<br>AES-OFB<br>AES-XTS | AES Key<br>AES GCM IV | CO | E | API call |

**Table 8  Approved Services**

| Service | Description | Algorithms Accessed | Role | Indicator |
|---|---|---|---|---|
| Asymmetric Encryption | Perform asymmetric encryption operation | ECIES<br>RSA-KEM-KWS | N/A | API call |
| Asymmetric Decryption | Perform asymmetric decryption operation | ECIES<br>RSA-KEM-KWS | N/A | API call |
| Digital Signature Generation | Perform digital signature generation | Deterministic DSA<br>Deterministic ECDSA | N/A | API call |
| Digital Signature Verification | Perform digital signature verification | Deterministic DSA<br>Deterministic ECDSA | N/A | API call |
| Key Derivation | Perform key derivation operation | KDFTLS10 (For use with TLS versions 1.0 and 1.1)<br>PKCS #5 KDF<br>PKCS #12 KDF<br>scrypt | N/A | API call |
| Key Generation | Perform key generation operation | DES<br>DESX<br>RC2<br>RC4<br>RC5<br>Shamir Secret Sharing<br>TDES | N/A | API call |
| Message Digest | Perform message digest operation | MD2<br>MD5<br>RIPEMD160 | N/A | API call |
| MAC Generation | Perform MAC generation operation | HMAC-MD5<br>PBHMAC (PKCS #12, PKIX)<br>Poly1305 | N/A | API call |
| MAC Verification | Perform MAC verification operation | HMAC-MD5<br>PBHMAC (PKCS #12, PKIX)<br>Poly1305 | N/A | API call |
| Random Number Generation | Perform random number generation | FIPS 186-2 PRNG (Change Notice General) | N/A | API call |
| Symmetric Encryption | Perform symmetric encryption operation | AES in BPS mode for FPE<br>ChaCha20<br>ChaCha20/Poly1305<br>DES<br>DESX<br>RC2<br>RC4<br>RC5<br>PBE (PKCS #12, PKCS #5, SSLCPBE)<br>TDES in CBC, CFB64, ECB, OFB modes and CBC_CS1, CBC_CS2 or CBC_CS3 mode for CTS | N/A | API call |
| Symmetric Decryption | Perform symmetric decryption operation | AES in BPS mode for FPE<br>ChaCha20<br>ChaCha20/Poly1305<br>DES<br>DESX<br>RC2<br>RC4<br>RC5<br>PBE (PKCS #12, PKCS #5, SSLCPBE)<br>TDES in CBC, CFB64, ECB, OFB modes and CBC_CS1, CBC_CS2 or CBC_CS3 mode for CTS | N/A | API call |

**Table 9 Non-Approved Services**

The Module doesn't support self-initiated cryptographic output capability and cryptographic Bypass capability services.

# 5 Software/Firmware security

**Integrity techniques**

Module integrity check is implemented by first calculating a MAC over each of the files listed in *module.files*, using HMAC-SHA-1 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.

**Integrity test 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

The 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 based on an Intel CPU.

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 FIPS 140-3 physical security requirements do not apply to the Module since it is a software module.

# 8 Non-invasive security

Currently, non-invasive security is not required by FIPS 140-3 (see NIST SP800-140F). The requirements of this area are not applicable to the Module.

# 9 Sensitive security parameters management

The following table summarizes the keys and Sensitive Security Parameters (SSPs) that are used by the cryptographic services implemented in the Module:

| Key/SSP Name/ Type | Strength | Security Function and Cert. Number | Generation | Import /Export | Establish-ment | Storage | Zero-isation | Use & related keys |
|---|---|---|---|---|---|---|---|---|
| DRBG entropy input (CSP) | 256 bits | CTR_DRBG HASH_DRBG HMAC_DRBG #A2314 | Obtained from the Entropy Source within TOEPP (GPS INT Pathways). | Import to the module via Module's API Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Random Number Generation |
| DRBG Seed (CSP) | 256 bits | CTR_DRBG HASH_DRBG HMAC_DRBG #A2314 | Internally Derived from entropy input string as defined by SP800-90Arev1. | Import: No Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Random Number Generation |
| DRBG Internal State V value (CSP) | 256 bits | CTR_DRBG HASH_DRBG HMAC_DRBG #A2314 | Internally Derived from entropy input string as defined by SP800-90Arev1. | Import: No Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Random Number Generation |
| DRBG Key (CSP) | 256 bits | CTR_DRBG HASH_DRBG HMAC_DRBG #A2314 | Internally Derived from entropy input string as defined by SP800-90Arev1. | Import: No Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down. | Random Number Generation |
| Diffie-Hellman Private Key (CSP) | MODP-2048 | CKG CTR_DRBG HASH_DRBG HMAC_DRBG KAS-FFC-SSC Safe Primes Key Generation #A2314 | Internally generated conformant to SP800-133rev2 (CKG) using SP800-56A rev3 Diffie-Hellman key generation method, and the random value used in key generation is generated using SP800-90ARev1 DRBG. | Import: No Export: No | N/A | N/A: The module does not provide persistent keys/ SSPs storage. | Automatic zeroization when the tested platform is powered down | Used to derive Diffie-Hellman Shared Secret |
| Diffie-Hellman Public Key (PSP) | MODP-2048 | KAS-FFC-SSC Safe Primes Key Generation #A2314 | Internally derived per the Diffie-Hellman key agreement (SP800-56Arev3). | Import: No Export: Yes | N/A | N/A: The module does not provide persistent keys/ SSPs storage. | Automatic zeroization when the tested platform is powered down | Used to derive Diffie-Hellman Shared Secret |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Peer Diffie-Hellman Public Key (PSP) | MODP-2048 | N/A | N/A | Import: Yes<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Used to derive Diffie-Hellman Shared Secret |
| Diffie-Hellman Shared Secret (CSP) | MODP-2048 | KAS-FFC-SSC<br><br>#A2314 | Internally generated using SP800-56Arev3 DH shared secret computation. | Import: No<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Used to derive TLS session related keys |
| EC Diffie-Hellman Private Key (CSP) | P-256/P-384/P-521 | CKG<br>CTR_DRBG<br>HASH_DRBG<br>HMAC_DRBG<br>KAS-ECC-SSC<br><br>#A2314 | Internally generated conformant to SP800-133rev2 (CKG) using SP800-56A rev3 EC Diffie-Hellman key generation method, and the random value used in key generation is generated using SP800-90Arev1 DRBG | Import: No<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/ SSPs storage. | Automatic zeroization when the tested platform is powered down | Used to derive EC Diffie-Hellman Shared Secret |
| EC Diffie-Hellman Public Key (PSP) | P-256/P-384/P-521 | KAS-ECC-SSC<br><br>#A2314 | Internally derived per the EC Diffie-Hellman key agreement (SP800-56Arev3). | Import: No<br><br>Export: Yes | N/A | N/A: The module does not provide persistent keys/ SSPs storage. | Automatic zeroization when the tested platform is powered down | Used to derive EC Diffie-Hellman Shared Secret |
| Peer EC Diffie-Hellman Public Key (PSP) | P-256/P-384/P-521 | N/A | N/A | Import: Yes<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Used to derive EC Diffie-Hellman Shared Secret |
| EC Diffie-Hellman Shared Secret (CSP) | P-256/P-384/P-521 | KAS-ECC-SSC<br><br>#A2314 | Internally derived using SP800-56Arev3 ECDH shared secret computation. | Import: No<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Used to derive TLS session related keys |
| DSA Private Key (CSP) | 2048/3072 bits<br><br>#A2314 | CKG<br>CTR_DRBG<br>HASH_DRBG<br>HMAC_DRBG<br>DSA PQGGen<br>DSA SigGen<br>DSA KeyGen<br><br>#A2314 | Internally generated conformant to SP800-133r2 (CKG) using FIPS 186-4 DSA key generation method, and the random | Import: Yes<br><br>Export: Yes | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Signature generation and Verification used in TLS |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | value used in key generation is generated using SP800-90Arev1 DRBG<br><br>Or externally generated | | | | | |
| DSA Public Key (PSP) | 1024/2048/3072 bits<br><br>#A2314 | DSA PQGVer<br>DSA SigVer<br>DSA KeyGen<br><br>#A2314 | Internally derived per the FIPS 186-4 DSA key generation method<br><br>Or externally generated | Import: Yes<br><br>Export: Yes | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Signature generation and Verification used in TLS |
| ECDSA Private Key (CSP) | P-256/P-384/P-521 | CKG<br>CTR_DRBG<br>HASH_DRBG<br>HMAC_DRBG<br>ECDSA KeyGen<br>ECDSA KeyVer<br>ECDSA SigGen<br><br>#A2314 | Internally generated conformant to SP800-133rev2 (CKG) using FIPS 186-4 ECDSA key generation method, and the random value used in key generation is generated using SP800-90Arev1 DRBG.<br><br>Or externally generated | Import: Yes<br><br>Export: Yes | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Signature generation and verification used in TLS |
| ECDSA Public Key (PSP) | P-256/P-384/P-521 | ECDSA KeyGen<br>ECDSA KeyVer<br>ECDSA SigVer<br><br>#A2314 | Internally derived per the FIPS 186-4 ECDSA key generation method.<br><br>Or externally generated | Import: Yes<br><br>Export: Yes | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Signature generation and verification used in TLS |
| RSA Private Key (CSP) | 2048/3072/4096 bits | CKG<br>CTR_DRBG<br>HASH_DRBG<br>HMAC_DRBG<br>RSA KeyGen<br>RSA SigGen<br><br>#A2314 | Internally generated conformant to SP800-133rev2 (CKG) using FIPS 186-4 RSA key generation method, and the random value used in the key generation is generated using SP800-90Arev1 | Import: Yes<br><br>Export: Yes | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Signature generation and verification used in TLS |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | DRBG.<br><br>Or externally generated | | | | | |
| RSA Public Key<br>(PSP) | 1024/2048/3072/4096 bits | RSA KeyGen<br>RSA SigVer<br><br>#A2314 | Internally derived per the FIPS 186-4 RSA key generation method.<br><br>Or externally generated | Import: Yes<br><br>Export: Yes | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Signature generation and verification used in TLS |
| TLS Master Secret<br>(CSP) | 48 Bytes | Keying Material | Internally Derived per the key derivation function defined in SP800-135 KDF (KDF-TLS v1.2 RFC7627) | Import: No<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when TLS session is terminated or when the tested platform is powered down | Keying material used to derive other TLS keys |
| TLS Session Key<br>(CSP) | 128/256 bits | AES-CBC<br>AES-GCM<br>TLS v1.2 KDF RFC7627<br>TLS v1.3 KDF<br><br>#A2314 | Internally Derived per the key derivation function defined in SP800-135 KDF (KDF-TLS v1.2 RFC7627). | Import: No<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when TLS session is terminated or when the tested platform is powered down | Used for TLS session confidentiality protection |
| TLS Session Integrity Key<br>(CSP) | 256-384 bits | TLS v1.2 KDF RFC7627<br>TLS v1.3 KDF<br>HMAC-SHA2-256<br>HMAC-SHA2-384<br><br>#A2314 | Internally Derived per the key derivation function defined in SP800-135 KDF (KDF-TLS v1.2 RFC7627). | Import: No<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when TLS session is terminated or when the tested platform is powered down | Used for TLS session integrity protection |
| AES Key<br>(CSP) | 128/192/256 bits | AES-CBC<br>AES-CBC-CS1<br>AES-CBC-CS2<br>AES-CBC-CS3<br>AES-CCM<br>AES-CFB<br>AES-CTR<br>AES-ECB<br>AES-GCM<br>AES-OFB<br>AES-XTS<br><br>#A2314 | Internally generated per the key generation function defined in SP800-133rev2 using random value generated using SP800-90A DRBG<br><br>Or externally generated | Import: Yes<br><br>Export: Yes | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Symmetric encryption and decryption |
| AES GCM IV<br>(CSP) | N/A | AES-GCM<br><br>#A2314 | Internally Derived per the key derivation function defined in: | Import: No<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is | Authenticated symmetric encryption and decryption |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | SP800-38D using random value generated using SP800-90A DRBG | | | | powered down | |
| CMAC Key (CSP) | 128/192/256 bits | CMAC-AES<br><br>#A2314 | Internally Derived per the key derivation function defined in: SP800-133rev2 using random value generated using SP800-90A DRBG<br><br>Or externally generated | Import: Yes<br><br>Export: Yes | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Message authentication |
| HMAC Key (CSP) | 112-256 bits | HMAC-SHA-1<br>HMAC-SHA2-224<br>HMAC-SHA2-256<br>HMAC-SHA2-384<br>HMAC-SHA2-512<br>HMAC-SHA2-512/224<br>HMAC-SHA2-512/256<br>HMAC-SHA3-224<br>HMAC-SHA3-256<br>HMAC-SHA3-384<br>HMAC-SHA3-512<br>#A2314 | Internally Derived per the key derivation function defined in: SP800-133rev2 using random value generated using SP800-90A DRBG<br><br>Or externally generated | Import: Yes<br><br>Export: Yes | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Message authentication |
| KBKDF Key Derivation Key (CSP) | N/A | KDF SP800-108<br><br>#A2314 | N/A | Import: Yes<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Key derivation |
| KBKDF Derived Key (CSP) | N/A | KDF SP800-108<br><br>#A2314 | N/A | Import: No<br><br>Export: Yes | SP800-108 | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Derived from key derivation key |
| OneStep KDF Key Derivation Key (CSP) | N/A | KDA OneStep<br><br>#A2314 | N/A | Import: Yes<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Key derivation |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| OneStep KDF Derived Key (CSP) | N/A | KDA OneStep<br><br>#A2314 | N/A | Import: No<br><br>Export: Yes | SP800-56C Rev.1 | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Derived from key derivation key |
| PBKDF Password (CSP) | N/A | PBKDF<br><br>#A2314 | N/A | Import: Yes<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Key derivation |
| PBKDF Derived Key (CSP) | N/A | PBKDF<br><br>#A2314 | N/A | Import: No<br><br>Export: Yes | SP800-132 | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Derived from key derivation key |
| AES Key Wrap Key (CSP) | 128/192/256 bits | AES-KW<br>AES-KWP<br><br>#A2314 | Internally generated per the key generation function defined in: SP800-133rev2 using random value generated using SP800-90A DRBG<br><br>Or externally generated | Import: Yes<br><br>Export: No | N/A | N/A: The module does not provide persistent keys/SSPs storage. | Automatic zeroization when the tested platform is powered down | Key wrapping and unwrapping |

**Table 10 SSPs**

## RBG entropy source

| Entropy sources | Minimum number of bits of entropy | Details |
|---|---|---|
| Entropy within the TOEPP was passively loaded into the Module to seed the 800-90Arev1 DRBG by the Operating System. | At least 112 bits | While operating in the approved mode, the entropy and seeding material for the SP800-90Arev1 DRBG are provided by the external calling application (and not by the Module) which is outside the Module's cryptographic boundary but contained within the Module's Tested Operational Environment's Physical Perimeter (TOEPP) boundary. The module receives a LOAD command with entropy obtained from the entropy source (Intel CPU processor with instructions RDRand) inside the TOEPP. The minimum effective strength of the SP800-90Arev1 DRBG seed is required to be at least 112 bits when used in an approved mode of operation, therefore the minimum number of bits of entropy requested when the Module makes a call to the SP800-90Arev1 DRBG is at least 112 bits.<br><br>Per the IG 9.3.A Entropy Caveats, the following caveat applies: *No assurance of the minimum strength of generated SSPs (e.g., keys)* |

**Table 11 Non-Deterministic Random Number Generation Specification**

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()`

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.

The following table lists each of the keys that can be generated by the JCM, with the key sizes available, security strengths for each key size, and the security strength required to initialize the DRBG:

| Key Type | Key Size | Security Strength | Required DRBG Security Strength |
|---|---|---|---|
| AES Key | 128, 192, 256 | 128, 192, 256 | 128, 192, 256 |
| RSA Key Pair | 2048, 3072, 4096 | 112, 128, 152 | 112, 128, 152 |
| DSA Key Pair | 2048, 3072 | 112, 128 | 112, 128 |
| EC Key Pair | 224, 256, 384, 521 | 112, 128, 192, 256 | 112, 128, 192, 256 |

**Table 12 Generated Key Sizes and Strength**

# 10 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, the Module 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

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`.

**Pre-operational software integrity test**

- HMAC-SHA-1 KAT
- Software Integrity Test (using HMAC-SHA-1)

Note: The Module conducts HMAC-SHA-1 KAT self-test before the integrity test is performed.

**Conditional self-tests**

- Cryptographic Algorithm Self-Tests (CASTs)
  - AES-CBC 256 bits Encrypt KAT
  - AES-CBC 256 bits Decrypt KAT
  - AES-GCM 256 bits Authenticated Encrypt KAT
  - AES-GCM 256 bits Authenticated Decrypt KAT
  - CMAC 128 bits KAT
  - CTR_DRBG Instantiate KAT
  - CTR_DRBG Generate KAT
  - CTR_DRBG Reseed KAT
    Note: CTR_DRBG Health Tests: Generate, Reseed, Instantiate functions per Section 11.3 of SP800-90Arev1
  - DSA SigGen with SHA-256 KAT
  - DSA SigVer with SHA-256 KAT
  - ECDSA P-256 with SHA-256 SigGen KAT
  - ECDSA P-256 with SHA-256 SigVer KAT
  - HASH-DRBG with SHA-1 KAT
  - HMAC-DRBG with SHA-1 KAT

- HMAC-SHA-1 KAT
- HMAC-SHA-256 KAT
- HMAC-SHA-384 KAT
- HMAC-SHA-512 KAT
- KAS-ECC-SSC Primitive Z KAT
- KAS-FFC-SSC Primitive Z KAT
- KDFTLS12 SHA-256 KAT
- OneStepKDF KAT
- PBKDF2 with SHA-1 KAT
- RSA 2048 bits modulus with SHA-256 SigGen KAT
- RSA 2048 bits modulus with SHA-256 SigVer KAT
- SHA-1 KAT
- SHA2-512 KAT
- SHA3-512 KAT
- SHAKE256 KAT
- SP800-108 KDF KAT

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.

**Pair-wise consistency tests (PCTs):**

- ECDSA PCT
- DSA PCT
- RSA PCT
- KAS-ECC PCT
- KAS-FFC PCT

**Periodic self-tests**

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.

**Error handling**

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

## Installation, Initialization, Startup, Operation and Maintenance

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

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 `securityLevel`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.

## Crypto Officer Guidance

For details of the administrative functions, security parameters, and logical interfaces available to the Crypto Officer. The following table details the requirements for algorithm use in the Approved mode operation:

| Algorithm | Guidance |
|---|---|
| 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 key 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.<br>• 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. |
| GCM Mode Ciphers | • 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.<br>• Where the Module is powered down, a new key must be used for AES GCM encryption/decryption.<br>• 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.<br>• 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:<br>– 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:<br>– The fixed field bytes are derived from the Module name, version information, and memory address of a Java class within the Module.<br>– 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.<br>– 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.<br>• The AES-GCM cipher used for the TLS protocol as the cipher implementation complies with SP800-52 and is compatible with RFC 5288 with the following conditions:<br>– The IV is configured as follows:<br>– The four-byte salt derived from the TLS handshake process 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 FIPS 140-3 IG C.H and is positioned in the name field of the IV as required in FIPS 140-3 IG C.H, TLS/DTLS 1.2 protocol IV generation.<br>– The remaining eight bytes of IV, referred to as `nonce_explicit` in RFC 5288, are generated deterministically by the module using the 64-bit counter used for the invocation field described above.<br>– When the 64-bit counter exhausts the maximum number of |

| | |
|---|---|
| | possible values for a given session key, the Module will throw a SecurityException.<br>– Whichever party, the client or the server, that encounters this condition must trigger a handshake to establish a new encryption key.<br>– 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. |
| HMAC | • The key length for an HMAC generation or verification must be between 112 and 4096 bits, inclusive.<br>• 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.<br>• A particular key-derivation key must only be used for a single key-expansion step. For more information, see SP800-56C Rev. 1.<br>• The derived key must be used only as a secret key.<br>• The derived key shall not be used as a key stream for a stream cipher.<br>• When selecting an HMAC hash, the output block size must be equal to or greater than the desired security strength of the derived key.<br>• 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.<br>• When selecting a hash algorithm, the output block size must be equal to or greater than the desired security strength of the derived key.<br>• The derived key must be used only as a secret key.<br>• The derived key shall not be used as a key stream for a stream cipher.<br>• The secret data input into this KDF must have a length equal to or greater than the desired security strength of the derived key. |
| TLS PRF Key Derivation Function | • TLS v1.2 PRF KDF is allowed only when the following conditions are satisfied:<br>– The KDF is performed in the context of the TLS protocol.<br>– HMAC is as specified in FIPS 198-1.<br>– P_HASH uses either SHA-256, SHA-384, or SHA-512. For more information, see SP800-135 Rev. 1.<br>• The TLS protocols have not been tested by the CAVP and CMVP. |
| 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:**<br>• For schemes using FFC, use one of the FFC safe-prime groups as defined in SP800-56A rev. 3 Appendix D.<br>• For schemes using ECC, use one of the approved curves as defined in SP800-56A rev. 3 Appendix D.<br><br>**Obtain a key pair from domain parameters:**<br>• For all schemes: |

| | |
|---|---|
| | – Both parties must use validated parameters to generate a key pair.<br>– The Module generates the key establishment key pair according to the required standards.<br>– Choose a FIPS Approved DRBG like HMAC DRBG to generate the key pair.<br>– Both parties validate the key pair:<br>The Module provides the following APIs to explicitly validate the public and private keys according to SP800-56A Rev.3:<br>`com.rsa.crypto.PublicKey.isValid(SecureRandom random) com.rsa.crypto.PrivateKey.isValid().`<br>The module provides the APIs to explicitly validate the key pair according to the<br>pairwise consistency requirements in SP800-56A Rev. 3:<br>`com.rsa.crypto.KeyPair.validate(SecureRandom random)`<br>`com.rsa.crypto.KeyPair.validate(AlgorithmParams params, SecureRandom random)`<br>If the key pair is generated with an approved method, then validation is assumed.<br>• For schemes that use static key pairs, a public identifier must be:<br>– Authoritatively associated with the key pair.<br>– Associated with the public key to allow any peer to recognize the key pair.<br>• For schemes that use ephemeral keys, the key pair must be:<br>– Used only for a single agreement transaction.<br>– Destroyed after use.<br>• For schemes that generate an FFC key pair from selected parameters, the key pair must not be used to generate a digital signature.<br><br>**Receive the peer's public key:**<br>• For all schemes, the receiving party must validate the peer's public key.<br>• For schemes that use static keys, the receiving party must have assurance of:<br>– The peer's ownership of the private key.<br>– The identifier is bound to the public key.<br><br>**Generate the Shared Secret:**<br>• For all schemes, the shared secret must be:<br>– Used only as input to an approved KDF.<br>– Treated as a CSP and destroyed after use.<br>• If the shared secret generation fails then the party must destroy all intermediate values.<br><br>**Generate and Confirm Secret Key Material:**<br>• For all schemes:<br>– Approved key-derivation method(s), including the format of `FixedInfo` as specified in SP800-56A Rev. 3.<br>– 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.<br>– If key generation fails then the party must destroy all calculated values.<br>– The shared secret, and any key material, is destroyed.<br>• For schemes that use key confirmation:<br>– Both parties must use a common, approved MAC to generate confirmation values.<br>– The MAC key will be generated as one of the key material elements.<br>– The input values for MAC tag generation must be formatted as per SP800-56A Rev. 3.<br>– The MAC key and tag lengths must satisfy the requirements of SP800-56A Rev. 3.<br>– The MAC key must be destroyed after use.<br>– If confirmation fails then destroy all calculated values.<br>All key material is destroyed before it is used for any other purpose.<br>– Approved key confirmation technique(s) as specified in SP800-56A Rev. 3. |
| 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. For details about the comparable security strengths of symmetric block ciphers and asymmetric key algorithms refer to Table 2 of NIST SP800-57 Part 1 Rev. 5.<br>• 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.<br>• 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.<br>• The SHA-1 digest is disallowed for the generation of digital signatures.<br>• 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.`<br>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). |
| Password-based Key Derivation | • Keys generated using PBKDF2 shall only be used in data storage applications.<br>• Minimum Password Length:<br>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:<br>$L = \lceil \log_2 S / \log_2 N \rceil$ |

| | |
|---|---|
| | The following provides examples for a password used by PBKDF2 where $S = 4.32 \times 10^{20}$:<br>**Character Set** **N L**<br>Case sensitive (a-z, A-Z) 52 13<br>Case sensitive alpha numeric 62 12<br>All ASCII printable<br>characters except space 94 11<br>• A password of the strength `S` can be guessed at random with the probability of 1 in $2^S$.<br>• The minimum length of the randomly-generated portion of the salt is 16 bytes.<br>• The iteration count is as large as possible, with a minimum of 10,000 iterations recommended.<br>• The maximum key length is $(2^{32} - 1)*b$, where `b` is the digest size of the message digest function in bytes.<br>• Derived keys can be used as specified in NIST SP800-132, Section 5.4, options 1 and 2. |
| XTS Mode Ciphers | • AES in XTS mode is approved only for hardware storage applications.<br>• The two keys used for XTS must be checked to ensure they are different. This check is performed automatically by the module. |

**Table 13 Algorithm Requirements for Approved Mode of Operation**

## 12 Mitigation of other attacks

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.

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.