



Raytheon Technologies

FIPS 140-2 Non-Proprietary Security Policy

Raytheon Cryptographic Module for Java

Software Version 3.0.2.1

Document Version 1.4

August 31, 2022

Prepared For:



Raytheon Technologies
870 Winter Street
Waltham, MA 02451
www.rtx.com

Prepared By:



SafeLogic Inc.
530 Lytton Ave, Suite 200
Palo Alto, CA 94301
www.safelogic.com

Overview

This document provides a non-proprietary FIPS 140-2 Security Policy for Raytheon Cryptographic Module for Java.

Table of Contents

| | |
|--|-----------|
| Overview | 2 |
| 1 Introduction | 5 |
| 1.1 About FIPS 140 | 5 |
| 1.2 About this Document | 5 |
| 1.3 External Resources | 5 |
| 1.4 Notices | 5 |
| 2 Raytheon Cryptographic Module for Java | 6 |
| 2.1 Cryptographic Module Specification | 6 |
| 2.1.1 Validation Level Detail | 6 |
| 2.1.2 Modes of Operation | 7 |
| 2.1.3 Module Configuration | 7 |
| 2.1.4 Approved Cryptographic Algorithms | 9 |
| 2.1.5 Non-Approved But Allowed Cryptographic Algorithms | 14 |
| 2.1.6 Non-Approved Mode of Operation | 14 |
| 2.2 Critical Security Parameters and Public Keys | 16 |
| 2.2.1 Critical Security Parameters | 16 |
| 2.2.2 Public Keys | 18 |
| 2.3 Module Interfaces | 19 |
| 2.4 Roles, Services, and Authentication | 20 |
| 2.4.1 Assumption of Roles | 20 |
| 2.4.2 Services | 21 |
| 2.5 Physical Security | 26 |
| 2.6 Operational Environment | 26 |
| 2.6.1 Use of External RNG | 26 |
| 2.7 Self-Tests | 27 |
| 2.7.1 Power-Up Self-Tests | 27 |
| 2.7.2 Conditional Self-Tests | 28 |
| 2.8 Mitigation of Other Attacks | 29 |
| 3 Security Rules and Guidance | 30 |
| 3.1 Basic Enforcement | 30 |
| 3.2 Additional Enforcement with a Java SecurityManager | 30 |
| 3.3 Basic Guidance | 30 |
| 3.4 Enforcement and Guidance for AES GCM IVs | 31 |
| 3.5 Enforcement and Guidance for Use of the Approved PBKDF | 31 |
| 3.6 Rules for Setting the N and the S String in cSHAKE | 32 |
| 3.7 Guidance for the Use of DRBGs and Configuring the JVM's Entropy Source | 32 |
| 3.8 Software Installation | 33 |
| 4 References and Acronyms | 34 |
| 4.1 References | 34 |
| 4.2 Acronyms | 36 |

List of Tables

| | |
|--|----|
| Table 1 - Validation Level by FIPS 140-2 Section | 6 |
| Table 2 - Available Java Permissions..... | 7 |
| Table 3 - FIPS Approved Algorithm Certificates..... | 9 |
| Table 4 - Approved Cryptographic Functions Implemented with Vendor Affirmation | 13 |
| Table 5 - Non-Approved But Allowed Cryptographic Algorithms..... | 14 |
| Table 6 - Non-Approved Cryptographic Functions for Use in non-Approved mode Only | 14 |
| Table 7 - Critical Security Parameters..... | 16 |
| Table 8 - Public Keys | 18 |
| Table 9 - Logical Interface / Physical Interface Mapping..... | 20 |
| Table 10 - Description of Roles | 21 |
| Table 11 - Module Services, Descriptions, and Roles | 21 |
| Table 12 - CSP Access Rights within Services..... | 24 |
| Table 13 - Tested Environments | 26 |
| Table 14 - Power-Up Self-Tests..... | 27 |
| Table 15 - Conditional Self-Tests | 28 |
| Table 16 – References | 34 |
| Table 17 - Acronyms and Terms | 36 |

List of Figures

| | |
|--|----|
| Figure 1 – Module Boundary and Interfaces Diagram..... | 19 |
|--|----|

1 Introduction

1.1 About FIPS 140

Federal Information Processing Standards Publication 140-2 — Security Requirements for Cryptographic Modules specifies requirements for cryptographic modules to be deployed in a Sensitive but Unclassified environment. The National Institute of Standards and Technology (NIST) and Canadian Centre for Cyber Security (CCCS) Cryptographic Module Validation Program (CMVP) run the FIPS 140 program. The NVLAP accredits independent testing labs to perform FIPS 140 testing; the CMVP validates modules meeting FIPS 140 validation. *Validated* is the term given to a module that is documented and tested against the FIPS 140 criteria.

More information is available on the CMVP website at <https://csrc.nist.gov/projects/cryptographic-module-validation-program>.

1.2 About this Document

This non-proprietary Cryptographic Module Security Policy for Raytheon Cryptographic Module for Java from Raytheon Technologies (“Raytheon”) provides an overview of the product and a high-level description of how it meets the overall Level 1 security requirements of FIPS 140-2.

Raytheon Cryptographic Module for Java may also be referred to as the “module” in this document.

1.3 External Resources

The Raytheon website (www.rtx.com) contains information on Raytheon services and products. The Cryptographic Module Validation Program website contains links to the FIPS 140-2 certificate and Raytheon contact information.

1.4 Notices

This document may be freely reproduced and distributed in its entirety without modification.

2 Raytheon Cryptographic Module for Java

2.1 Cryptographic Module Specification

Raytheon Cryptographic Module for Java is a standards-based cryptographic engine for native Java environments. The module delivers core cryptographic functions to mobile and server platforms and features robust algorithm support, including Suite B algorithms. Raytheon Cryptographic Module for Java offloads secure key management, data integrity, data at rest encryption, and secure communications to a trusted implementation.

The module's software version is 3.0.2.1. The module's logical cryptographic boundary is the Java Archive (JAR) file (ccj-3.0.2.1.jar).

The module is a software module that relies on the physical characteristics of the host platform. The module's physical cryptographic boundary is defined by the enclosure of the host platform, which is the General Purpose Device that the module is installed on. For the purposes of FIPS 140-2 validation, the module's embodiment type is defined as multi-chip standalone.

All operations of the module occur via calls from host applications and their respective internal daemons/processes. As such there are no untrusted services calling the services of the module.

2.1.1 Validation Level Detail

The following table lists the module's level of validation for each area in FIPS 140-2:

Table 1 - Validation Level by FIPS 140-2 Section

| FIPS 140-2 Section Title | Validation Level |
|--|------------------|
| Cryptographic Module Specification | 1 |
| Cryptographic Module Ports and Interfaces | 1 |
| Roles, Services, and Authentication | 1 |
| Finite State Model | 1 |
| Physical Security | N/A |
| Operational Environment | 1 |
| Cryptographic Key Management | 1 |
| Electromagnetic Interference / Electromagnetic Compatibility | 1 |
| Self-Tests | 1 |
| Design Assurance | 1 |
| Mitigation of Other Attacks | 1 |

2.1.2 Modes of Operation

The module supports two modes of operation: FIPS Approved mode and non-Approved mode. The module will be in FIPS Approved mode when the appropriate transition method is called. To verify that a module is in the FIPS Approved mode of operation, the user can call a FIPS Approved mode status method (*CryptoServicesRegistrar.isInApprovedOnlyMode()*). If the module is configured to allow FIPS Approved mode and non-Approved mode operations, a call to *CryptoServicesRegistrar.setApprovedMode(true)* will switch the current thread of user control into FIPS Approved mode.

In FIPS Approved mode, the module will not provide non-Approved algorithms, therefore, exceptions will be called if the user tries to access non-Approved algorithms in the FIPS Approved mode.

2.1.3 Module Configuration

In default operation, the module will start with both FIPS Approved mode and non-Approved mode enabled.

If the module detects that the system property *com.safelogic.cryptocomply.fips.approved_only* is set to *true* the module will start in FIPS Approved mode and non-Approved mode functionality will not be available.

If the underlying JVM is running with a Java Security Manager installed, the module will be running in FIPS Approved mode with secret and private key export disabled.

Use of the module with a Java Security Manager requires the setting of some basic permissions to allow the module HMAC-SHA-256 software integrity test to take place as well as to allow the module itself to examine secret and private keys. The basic permissions required for the module to operate correctly with a Java Security Manager are indicated by a Y in the **Req** column of Table 2 - Available Java Permissions.

Table 2 - Available Java Permissions

| Permission | Settings | Req | Usage |
|--------------------------|-------------------------------|-----|--|
| RuntimePermission | "getProtectionDomain" | Y | Allows checksum to be carried out on jar |
| RuntimePermission | "accessDeclaredMembers" | Y | Allows use of reflection API within the provider |
| PropertyPermission | "java.runtime.name", "read" | N | Only if configuration properties are used |
| SecurityPermission | "putProviderProperty.BCFIPS" | N | Only if provider installed during execution |
| CryptoServicesPermission | "unapprovedModeEnabled" | N | Only if non-Approved mode algorithms required |
| CryptoServicesPermission | "changeToApprovedModeEnabled" | N | Only if threads allowed to change modes |

| Permission | Settings | Req | Usage |
|--------------------------|---------------------------|-----|--|
| CryptoServicesPermission | "exportSecretKey" | N | To allow export of secret keys only |
| CryptoServicesPermission | "exportPrivateKey" | N | To allow export of private keys only |
| CryptoServicesPermission | "exportKeys" | Y | Required to be applied for the module itself. Optional for any other codebase. |
| CryptoServicesPermission | "tlsNullDigestEnabled" | N | Only required for TLS digest calculations |
| CryptoServicesPermission | "tlsPKCS15KeyWrapEnabled" | N | Only required if TLS is used with RSA encryption |
| CryptoServicesPermission | "tlsAlgorithmsEnabled" | N | Enables both NullDigest and PKCS15KeyWrap |
| CryptoServicesPermission | "defaultRandomConfig" | N | Allows setting of default SecureRandom |
| CryptoServicesPermission | "threadLocalConfig" | N | Required to set a thread local property in the CryptoServicesRegistrar |
| CryptoServicesPermission | "globalConfig" | N | Required to set a global property in the CryptoServicesRegistrar |

2.1.4 Approved Cryptographic Algorithms

2.1.4.1 CAVP Tested Approved Algorithms

The module’s cryptographic algorithm implementations have received the following certificate numbers from the Cryptographic Algorithm Validation Program (CAVP).

Table 3 - FIPS Approved Algorithm Certificates

| CAVP Cert. | Algorithm | Standard | Mode/Method | Key Lengths, Curves or Moduli | Use |
|------------|--|------------------------|--|---|----------------------------|
| A2720 | AES | FIPS 197 SP 800-38A | CBC, ECB, CFB8, CFB128, CTR, OFB | 128, 192, 256 | Encryption, Decryption |
| A2720 | AES CCM | SP 800-38C | CCM | 128, 192, 256 | Generation, Authentication |
| A2720 | AES CMAC | SP 800-38B | CMAC | 128, 192, 256 | Generation, Authentication |
| A2720 | AES GCM/GMAC ¹ | SP 800-38D | GCM/GMAC | 128, 192, 256 | Generation, Authentication |
| A2720 | CVL: KDF, Existing Application-Specific ² | SP 800-135 | TLS v1.0/1.1 KDF, TLS 1.2 KDF, SSH KDF, X9.63 KDF, IKEv2 KDF, SRTP KDF | Various (See #A2720 for details) | KDF Services |
| A2720 | DRBG | SP 800-90A | Hash DRBG HMAC DRBG CTR DRBG | 112, 128, 192, 256 (SHA-1, SHA-2, 3-Key Triple DES, AES) | Random Bit Generation |

¹ GCM encryption with an internally generated IV, see Security Policy section 3.4 concerning external IVs. IV generation is compliant with IG A.5.

² These protocols have not been reviewed or tested by the CAVP and CMVP.

| CAVP Cert. | Algorithm | Standard | Mode/Method | Key Lengths, Curves or Moduli | Use |
|------------|------------------|------------|--|--|---|
| A2720 | DSA ³ | FIPS 186-4 | Key Pair Generation, PQG Generation, PQG Verification, Signature Generation, Signature Verification | (1024, 160) ⁴ (2048, 224) (2048, 256) (3072, 256) | Digital Signature Services |
| A2720 | ECDSA | FIPS 186-4 | Key Generation, Signature Generation, Signature Verification, Public Key Validation, Signature Generation Component (CVL) | P-192 ⁵ , P-224, P-256, P-384, P- 521, K-163 ⁶ , K-233, K-283, K-409, K-571, B-163 ⁷ , B-233, B-283, B-409, B-571 | Digital Signature Services |
| A2720 | HMAC | FIPS 198-1 | HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, HMAC-SHA-512/224, HMAC-SHA-512/256, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512 | Various (KS<BS, KS=BS, KS>BS) | HMAC Generation, HMAC Authentication |

³ DSA signature generation with SHA-1 is only for use with protocols

⁴ Key size only used for Signature Verification

⁵ Curves only used for Signature Verification and Public Key Validation

⁶ Curves only used for Signature Verification and Public Key Validation

⁷ Curves only used for Signature Verification and Public Key Validation

| CAVP Cert. | Algorithm | Standard | Mode/Method | Key Lengths, Curves or Moduli | Use |
|--------------|--|---|--|---|--|
| A2720 | KBKDF, using Pseudorandom Functions | SP 800-108 | Counter Mode, Feedback Mode, Double-Pipeline Iteration Mode | CMAC-based KDF: AES (128, 192, 256), 3-key Triple-DES HMAC-based KDF: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 ⁸ | KDF Services |
| A2720 (AES) | KTS: Key Wrapping Using AES ⁹ | SP 800-38F | AES KW, AES KWP | 128, 192, 256 | Key Transport For AES, the key establishment methodology provides between 128 and 256 bits of encryption strength |
| A2720 (TDES) | KTS: Key Wrapping Using TDES ¹⁰ | SP 800-38F | TKW | 3-key Triple-DES | Key Transport For Triple-DES, key establishment methodology provides 112 bits of encryption strength |
| A2720 | RSA | FIPS 186-4 SP 800-56B Section 7.1.2 | Key Pair Generation: 2048, 3072 Signature Generation (ANSI X9.31, PKCS 1.5, and PKCSPSS): 2048, 3072 Signature Verification (ANSI X9.31, PKCS 1.5, and PKCSPSS): 1024, 2048, 3072, 4096 RSA Signature Primitive Component (CVL): 2048 RSA Decryption Primitive Component (CVL) per SP 800-56B: 2048 | | Digital Signature Services, Key Transport (per SP 800-56B) |
| | | FIPS 186-2 | Signature Verification (ANSI X9.31, PKCS 1.5, and PKCSPSS): 1024, 1536, 2048, 3072, 4096 bits | | |

⁸ Note: CAVP testing is not provided for use of the PRFs SHA-512/224 and SHA-512/256. These must not be used in FIPS Approved mode.

⁹ Keys are not established directly into the module using key unwrapping.

¹⁰ Keys are not established directly into the module using key unwrapping.

| CAVP Cert. | Algorithm | Standard | Mode/Method | Key Lengths, Curves or Moduli | Use |
|------------|-----------------|------------|---|---|--|
| A2720 | SHA-3, SHAKE | FIPS 202 | SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256, | N/A | Digital Signature Generation, Digital Signature Verification, non-Digital Signature Applications |
| A2720 | SHS | FIPS 180-4 | SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256 | N/A | Digital Signature Generation, Digital Signature Verification, non-Digital Signature Applications |
| A2720 | Triple-DES | SP 800-67 | TCBC, TCFB8, TCFB64, TECB, TOFB, CTR | 2-key ¹¹ , 3-key ¹² | Encryption, Decryption |
| A2720 | Triple-DES CMAC | SP 800-38B | Triple-DES | Triple-DES with 2-key ¹³ , 3-key | Generation, Authentication |

¹¹ 2²⁰ block limit is enforced by the module, 2-key encryption is disabled.

¹² 3-key Triple-DES encryption must not be used for more than 2²⁰ blocks for any given key.

¹³ 2²⁰ block limit is enforced by the module. In FIPS Approved mode, the use of 2-key Triple-DES to generate MACs for anything other than verification purposes is non-compliant.

2.1.4.2 Vendor Affirmed Approved Algorithms

The following Approved cryptographic algorithms were implemented with vendor affirmation.

Table 4 - Approved Cryptographic Functions Implemented with Vendor Affirmation

| Algorithm | IG Reference | Use |
|---|------------------------------------|---|
| AES-CBC Ciphertext Stealing (CS) | Vendor Affirmed per IG A.12 | [Addendum to SP 800-38A, Oct 2010] Functions: Encryption, Decryption Modes: CBC-CS1, CBC-CS2, CBC-CS3 Key Sizes: 128, 192, 256 |
| CKG using output from DRBG ¹⁴ | Vendor Affirmed per IG D.12 | [SP 800-133] Section 6.1 (Asymmetric from DRBG) Section 7.1 (Symmetric from DRBG) Using DRBG #A2720 |
| cSHAKE128, cSHAKE256 | Vendor Affirmed per IG A.15 | [SP 800-185] Section 3, cSHAKE Using SHA3 #A2720, SHAKE #A2720 |
| KAS-SSC ¹⁵ | Vendor Affirmed per IG D.1-rev3 | [SP 800-56Ar3] <ul style="list-style-type: none"> • Section 5.6.2.3.1 (Finite Field Cryptography (FFC) Full Public Key Validation Routine) • Section 5.6.2.3.2 (Elliptic Curve Cryptography (ECC) Full Public Key Validation Routine) • Section 5.7 (DLC Primitive) • Section 5.8 (Key Derivation Functions for Key Agreement Schemes) • Section 5.9 (Key Confirmation) • Section 6 (Key Agreement) • Parameter sets/Key sizes <ul style="list-style-type: none"> ○ ECC: Approved P, B, K Curves per Appendix D ○ FFC: Safe primes per Appendix D (safe primes key generation tested under #A2720) |
| KTS: Key Transport ¹⁶ Using RSA | Vendor Affirmed per IG D.4 | [SP 800-56B, Section 7.2.3] RSA-KEM-KWS with, and without, key confirmation Key sizes: 2048, 3072 bits |
| KTS: Key Transport ¹⁷ Using RSA | Vendor Affirmed per IG D.4 | [SP 800-56B, Section 7.2.2] RSA-OAEP with, and without, key confirmation Key sizes: 2048, 3072 bits |

¹⁴ The resulting key or a generated seed is an unmodified output from a DRBG

¹⁵ Keys are not directly established into the module using key agreement or transport techniques.

¹⁶ Keys are not directly established into the module using key agreement or transport techniques.

¹⁷ Keys are not directly established into the module using key agreement or transport techniques.

| Algorithm | IG Reference | Use |
|--------------------------------------|-----------------------------|---|
| PBKDF, password-based key derivation | Vendor Affirmed per IG D.6 | [SP 800-132] Options: PBKDF with Option 1a Functions: HMAC-based KDF using SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 Using HMAC #A2720 Refer also to Security Policy section 3.5 - Enforcement and Guidance for Use of the Approved PBKDF |
| RSA | Vendor Affirmed per IG A.14 | [SP 800-131Ar2] Section 3 Key sizes: 4096 - 16384 bits Using mechanism tested in #A2720 |

2.1.5 Non-Approved But Allowed Cryptographic Algorithms

The module supports the following FIPS 140-2 non-Approved but allowed algorithms that may be used in the FIPS Approved mode of operation.

Table 5 - Non-Approved But Allowed Cryptographic Algorithms

| Algorithm | Use |
|--|--|
| MD5 within TLS | [IG D.2, IG 1.23 example 2a] |
| NDRNG | [IG 7.15, IG 7.14 example 1b] Non-deterministic random number generator. The module generates cryptographic keys whose strengths are modified by available entropy |
| RSA Key Wrapping, Non-SP 800-56B compliant | [IG D.9] RSA may be used by a calling application as part of a key encapsulation scheme. Key sizes: 4096 - 16384 bits (key wrapping; key establishment methodology provides between 150 and 256 bits of encryption strength) |

2.1.6 Non-Approved Mode of Operation

The module supports a non-Approved mode of operation. The algorithms listed in this section are not to be used by the operator in the FIPS Approved mode of operation.

Table 6 - Non-Approved Cryptographic Functions for Use in non-Approved mode Only

| Algorithm | Use |
|------------------------------------|------------------------|
| AES (non-compliant ¹⁸) | Encryption, Decryption |
| ARC4 (RC4) | Encryption, Decryption |
| Blowfish | Encryption, Decryption |

¹⁸ Support for additional modes of operation.

| Algorithm | Use |
|---|--------------------------|
| Camellia | Encryption, Decryption |
| CAST5 | Encryption, Decryption |
| DES | Encryption, Decryption |
| DSA (non-compliant ¹⁹) | Public Key Cryptography |
| DSTU4145 | Public Key Cryptography |
| ECDSA (non-compliant ²⁰) | Public Key Cryptography |
| EdDSA | Public Key Cryptography |
| ElGamal | Public Key Cryptography |
| GOST28147 | Encryption, Decryption |
| GOST3410-1994 | Hashing |
| GOST3410-2001 | Hashing |
| GOST3411 | Hashing |
| HMAC-GOST3411 | Hashing |
| HMAC-MD5 | Hashing |
| HMAC-RIPEMD128 | Hashing |
| HMAC-RIPEMD160 | Hashing |
| HMAC-RIPEMD256 | Hashing |
| HMAC-RIPEMD320 | Hashing |
| HMAC-TIGER | Hashing |
| HMAC-WHIRLPOOL | Hashing |
| IDEA | Encryption, Decryption |
| KAS ²¹ , Diffie-Hellman (non-compliant ²²) | Key Agreement |
| KAS ²³ using SHA-512/224 or SHA-512/256 | Key Agreement |
| KBKDF using SHA-512/224 or SHA-512/256 (non-compliant) | KDF |
| MD5 | Hashing |
| OpenSSL PBKDF (non-compliant) | KDF |
| PKCS#12 PBKDF (non-compliant) | KDF |
| PKCS#5 Scheme 1 PBKDF (non-compliant) | KDF |
| PRNG X9.31 | Random Number Generation |
| RC2 | Encryption, Decryption |
| RIPEMD128 | Hashing |
| RIPEMD160 | Hashing |
| RIPEMD256 | Hashing |
| RIPEMD320 | Hashing |
| RSA (non-compliant ²⁴) | Public Key Cryptography |
| RSA KTS (non-compliant ²⁵) | Public Key Cryptography |
| SCrypt | KDF |

¹⁹ Deterministic signature calculation, support for additional digests, and key sizes.

²⁰ Deterministic signature calculation, support for additional digests, and key sizes.

²¹ Keys are not directly established into the module using key agreement or transport techniques.

²² Support for additional key sizes and the establishment of keys of less than 112 bits of security strength.

²³ Keys are not directly established into the module using key agreement or transport techniques.

²⁴ Support for additional digests and signature formats, PKCS#1 1.5 key wrapping, support for additional key sizes.

²⁵ Support for additional key sizes and the establishment of keys of less than 112 bits of security strength.

| Algorithm | Use |
|---|------------------------|
| SEED | Encryption, Decryption |
| Serpent | Encryption, Decryption |
| SipHash | Hashing |
| SHACAL-2 | Encryption, Decryption |
| TIGER | Hashing |
| Triple DES (non-compliant ²⁶) | Encryption, Decryption |
| Twofish | Encryption, Decryption |
| WHIRLPOOL | Hashing |
| XDH | Key Agreement |

2.2 Critical Security Parameters and Public Keys

2.2.1 Critical Security Parameters

The table below provides a complete list of Critical Security Parameters used within the module:

Table 7 - Critical Security Parameters

| CSP | Description / Usage |
|------------------------|--|
| AES Encryption Key | [FIPS 197, SP 800-38A, SP 800-38C, SP 800-38D, Addendum to SP 800-38A] AES (128/192/256) encrypt key ²⁷ |
| AES Decryption Key | [FIPS 197, SP 800-38A, SP 800-38C, SP 800-38D, Addendum to SP 800-38A] AES (128/192/256) decrypt key |
| AES Authentication Key | [FIPS 197] AES (128/192/256) CMAC/GMAC key |
| AES Wrapping Key | [SP 800-38F] AES (128/192/256) key wrapping key |
| DH Agreement Key | [SP 800-56Ar3] Diffie-Hellman (160 - 512 bits) private key agreement key |
| DRBG (CTR AES) | V (128 bits) and AES key (128/192/256), entropy input (length dependent on security strength) |
| DRBG (CTR Triple-DES) | V (64 bits) and Triple-DES key (192), entropy input (length dependent on security strength) |
| DRBG (Hash) | V (440/888 bits) and C (440/888 bits), entropy input (length dependent on security strength) |
| DRBG (HMAC) | V (160/224/256/384/512 bits) and Key (160/224/256/384/512 bits), entropy input (length dependent on security strength) |
| DSA Signing Key | [FIPS 186-4] DSA (2048/3072) signature generation key |

²⁶ Support for additional modes of operation

²⁷ The AES GCM key and IV are generated randomly per IG A.5, and the Initialization Vector (IV) is a minimum of 96 bits. In the event module power is lost and restored, the consuming application must ensure that any of its AES GCM keys used for encryption or decryption are re-distributed. Refer also to Security Policy section 3.4.

| CSP | Description / Usage |
|---|--|
| EC Agreement Key | [SP 800-56Ar3] EC (P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409 and B-571) private key agreement key |
| EC Signing Key | [FIPS 186-4] ECDSA (P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409 and B-571) signature generation key |
| HMAC Authentication Key | [FIPS 198-1] Keyed-Hash key (SHA-1, SHA-2, SHA-3). Key size determined by security strength required (≥ 112 bits) |
| IKEv2 Derivation Function Secret Value | [SP 800-135] Secret value used in construction of key for the specified IKEv2 PRF |
| PBKDF Secret Value | [SP 800-132] Secret value used in construction of Keyed-Hash key for the specified PRF |
| RSA Signing Key | [FIPS 186-4] RSA (2048 - 16384 bits) signature generation key |
| RSA Key Transport Key | [SP 800-56B] RSA (2048 - 16384 bits) key transport (decryption) key |
| SP 800-56C Concatenation Derivation Function Secret Value | [SP 800-56C] Secret value used in construction of key for underlying PRF |
| SP 800-108 KDF Secret Value | [SP 800-108] Secret value used in construction of key for the specified PRF |
| SRTP Derivation Function Secret Value | [SP 800-135] Secret value used in construction of key for the specified SRTP PRF |
| SSH Derivation Function Secret Value | [SP 800-135] Secret value used in construction of key for the specified SSH PRF |
| TLS KDF Secret Value | [SP 800-135] Secret value used in construction of Keyed-Hash key for the specified TLS PRF |
| Triple-DES Encryption Key | [SP 800-67] Triple-DES (192 bits) encryption key |
| Triple-DES Decryption Key | [SP 800-67] Triple-DES (128/192 bits) decryption key |
| Triple-DES Authentication Key | [SP 800-67] Triple-DES (128/192 bits) CMAC key |
| Triple-DES Wrapping Key | [SP 800-38F] Triple-DES key wrapping (192 bits)/unwrapping key (128/192 bits) |
| X9.63 KDF Secret Value | [SP 800-135] Secret value used in construction of Keyed-Hash key for the specified X9.63 PRF |

2.2.2 Public Keys

The table below provides a complete list of the public keys used within the module:

Table 8 - Public Keys

| Public Key | Description / Usage |
|-----------------------|---|
| DH Agreement Key | [SP 800-56Ar3] Diffie-Hellman (2048 and 3072) public key agreement key |
| DSA Verification Key | [FIPS 186-4] DSA (1024/2048/3072) signature verification key |
| EC Agreement Key | [SP 800-56Ar3] EC (P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409 and B-571) public key agreement key |
| EC Verification Key | [FIPS 186-4] ECDSA (P-192, P-224, P-256, P-384, P-521, K-163, K-233, K-283, K-409, K-571, B-163, B-233, B-283, B-409 and B-571) signature verification key |
| RSA Key Transport Key | [SP 800-56B] RSA (2048 - 16384) key transport (encryption) key |
| RSA Verification Key | [FIPS 186-4] RSA (1024 - 16384) signature verification key |

2.3 Module Interfaces

The figure below shows the module’s physical and logical block diagram:

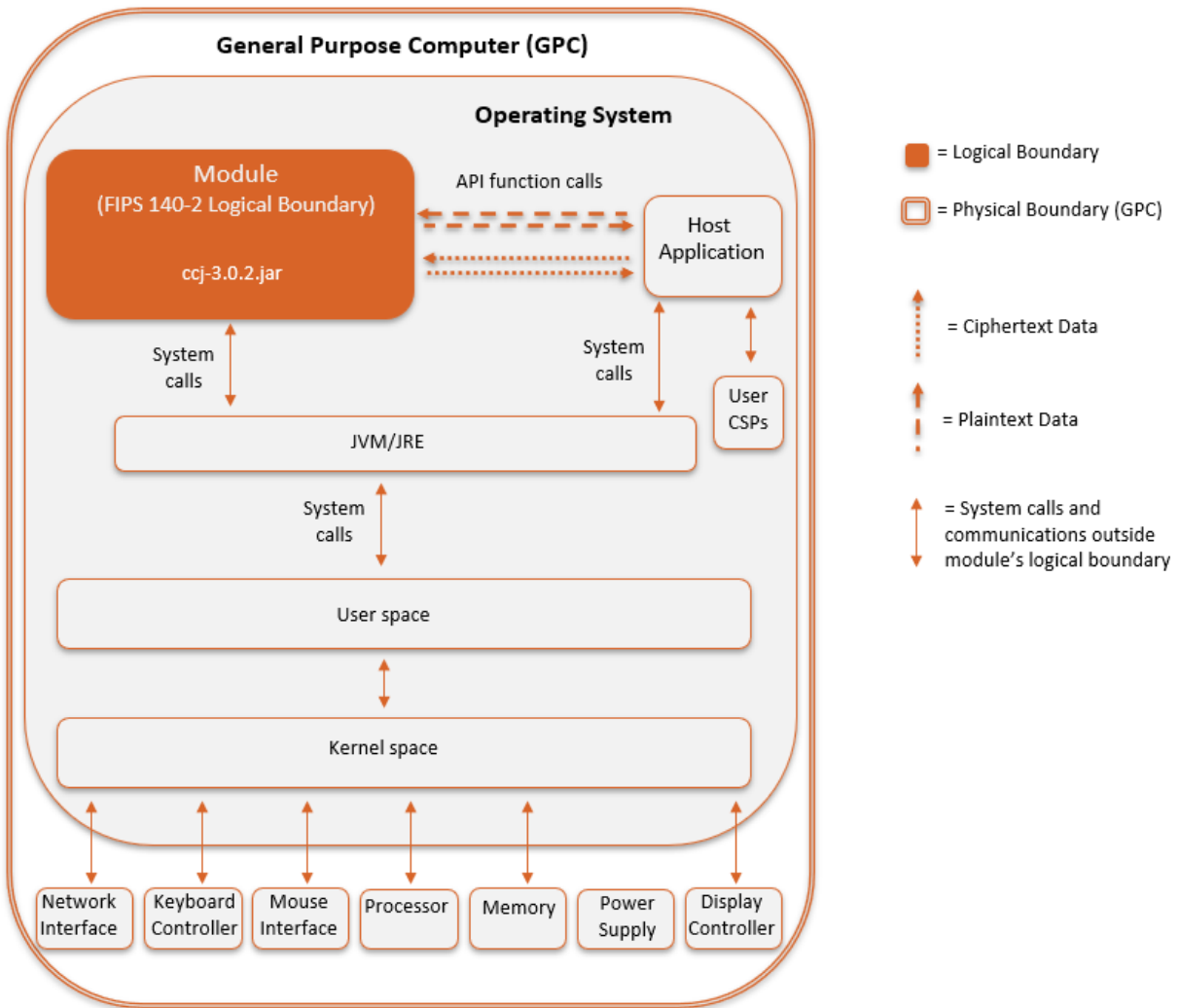


Figure 1 – Module Boundary and Interfaces Diagram

The module’s physical boundary is the boundary of the General Purpose Computer (GPC) that the module is installed on, which includes a processor and memory. The interfaces (ports) for the physical boundary include the computer’s network port, keyboard port, mouse port, power plug, and display. When operational, the module does not transmit any information across these physical ports because it is a software cryptographic module. Therefore, the module’s interfaces are purely logical.

Figure 1 shows the logical relationship of the cryptographic module to the other software and hardware components of the GPC. The module classes are executed on the Java Virtual Machine (JVM) using the classes of the Java Runtime Environment (JRE). The JVM is the interface to the computer’s Operating System (OS), which is the interface to the various physical components of the computer. The logical

interface is provided through an Application Programming Interface (API) that a calling daemon can operate. The API itself defines the module’s logical boundary, i.e. all access to the module is through this API. The API provides functions that may be called by an application (see Section 2.4 – Roles, Services, and Authentication for the list of available functions). The module distinguishes between logical interfaces by logically separating the information according to the defined API.

The API provided by the module is mapped onto the FIPS 140- 2 logical interfaces, which relate to the module’s callable interface as follows:

Table 9 - Logical Interface / Physical Interface Mapping

| FIPS 140-2 Interface | Logical Interface | Module Physical Interface |
|----------------------|---|--|
| Data Input | API input parameters – plaintext and/or ciphertext data | Network Interface |
| Data Output | API output parameters and return values – plaintext and/or ciphertext data | Network Interface |
| Control Input | API method calls – method calls, or input parameters, that specify commands and/or control data used to control the operation of the module | Network Interface, Keyboard Interface, Mouse Interface |
| Status Output | API output parameters and return/error codes that provide status information used to indicate the state of the module | Display Controller, Network Interface |
| Power | None | Power Supply |

When the module performs self-tests, is in an error state, is generating keys, or performing zeroization, the module prevents all output on the logical data output interface as only the thread performing the operation has access to the data. The module is single-threaded, and in an error state, the module does not return any output data, only an error value.

2.4 Roles, Services, and Authentication

2.4.1 Assumption of Roles

The module supports two distinct operator roles, which are the User and Crypto Officer (CO), as indicated in Table 10 - Description of Roles. The cryptographic module implicitly maps the two roles to the services. A user is considered the owner of the thread that instantiates the module and, therefore, only one concurrent user is allowed.

The module does not support a Maintenance role or bypass capability. The module does not support authentication.

Table 10 - Description of Roles

| Role | Role Description | Authentication Type |
|------|---|--|
| CO | Crypto Officer – Powers the module on and off | N/A – Authentication is not a requirement for FIPS 140 Level 1 |
| User | User – The user of the complete API | N/A – Authentication is not a requirement for FIPS 140 Level 1 |

2.4.2 Services

All services implemented by the module are listed in Table 11 - Module Services, Descriptions. The second column provides a description of each service, and availability to the Crypto Officer and User is indicated in columns three and four, respectively. Table 12 - CSP Access Rights within Services describes all CSP usage by services.

Table 11 - Module Services, Descriptions, and Roles

| Service | Description | CO | User |
|--|--|----|------|
| Initialize Module and Run Self-Tests on Demand | The JRE will call the static constructor for self-tests on module initialization. | X | |
| Show Status | A user can call <i>FipsStatus.IsReady()</i> at any time to determine if the module is ready. <i>CryptoServicesRegistrar.IsInApprovedOnlyMode()</i> can be called to determine the FIPS mode of operation. | | X |
| Zeroize / Power-off | The module uses the JVM garbage collector on thread termination. | | X |
| Data Encryption | Used to encrypt data. | | X |
| Data Decryption | Used to decrypt data. | | X |
| MAC Calculation | Used to calculate data integrity codes with CMAC. | | X |
| Signature Generation | Used to generate digital signatures (DSA, ECDSA, RSA). | | X |
| Signature Verification | Used to verify digital signatures (DSA, ECDSA, RSA). | | X |
| DRBG (SP 800-90A) output | Used for random number, IV and key generation. | | X |
| Message Hashing | Used to generate a SHA-1, SHA-2, or SHA-3 message digest, SHAKE output. | | X |
| Keyed Message Hashing | Used to calculate data integrity codes with HMAC. | | X |
| TLS Key Derivation Function | (secret input) (outputs secret) Used to calculate a value suitable to be used for a master secret in TLS from a pre-master secret and additional input. | | X |
| SP 800-108 KBKDF | (secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input. | | X |
| SSH Derivation Function | (secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input. | | X |

| Service | Description | CO | User |
|--|---|----|------|
| X9.63 Derivation Function | (secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input. | | X |
| SP 800-56C Concatenation Derivation Function | (secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input. | | X |
| IKEv2 Derivation Function | (secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input. | | X |
| SRTP Derivation Function | (secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input. | | X |
| PBKDF | (secret input) (outputs secret) Used to generate a key using an encoding of a password and an additional function such as a message hash. | | X |
| Key Agreement Schemes | Used to calculate key agreement values (SP 800-56Ar3, key agreement in non-Approved mode) | | X |
| Key Wrapping/Transport | Used to encrypt a key value. (RSA, AES, Triple-DES) | | X |
| Key Unwrapping | Used to decrypt a key value. (RSA, AES, Triple-DES) | | X |
| NDRNG Callback | Gathers entropy in a passive manner from a user-provided function. | | X |
| Utility | Miscellaneous utility functions, does not access CSPs. | | X |

Note: The module services are the same in the FIPS Approved and non-Approved modes of operation. The only difference is the function(s) used (Approved/allowed or non-Approved/non-allowed).

Services in the module are accessed via the public APIs of the Jar file. The ability of a thread to invoke non-Approved services depends on whether it has been registered with the module as FIPS Approved mode only. In FIPS Approved only mode, no non-Approved services are accessible. In the presence of a Java SecurityManager FIPS Approved mode services specific to a context (such as DSA and ECDSA for use in TLS) require specific permissions to be configured in the JVM configuration by the Crypto Officer or User.

In the absence of a Java SecurityManager specific services related to protocols such as TLS are available, however must only be used in relation to those protocols.

Table 12 - CSP Access Rights within Services defines the relationship between access to CSPs and the different module services. The modes of access shown in the table are defined as:

- **G** = Generate: The module generates the CSP.
- **R** = Read: The module reads the CSP. The read access is typically performed before the module uses the CSP.
- **E** = Execute: The module executes using the CSP.

- **W** = Write: The module writes the CSP. The write access is typically performed after a CSP is imported into the module, when the module generates a CSP, or when the module overwrites an existing CSP.
- **Z** = Zeroize: The module zeroizes the CSP.

Table 12 - CSP Access Rights within Services

| Services | CSPs | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|--------------------|--------------------|------------------------|------------------|------------------|----------------|-----------------------|-------------|-------------|-----------------|------------------|----------------|-------------------------|-----------------|--------------|-----------------|-----------------------|------------------------------|-----------------------|----------------|---------------------|----------------|---------------------------|---------------------------|-------------------------------|-------------------------|------------------------|---|--|
| | AES Encryption Key | AES Decryption Key | AES Authentication Key | AES Wrapping Key | DH Agreement Key | DRBG (CTR AES) | DRBG (CTR Triple-DES) | DRBG (Hash) | DRBG (HMAC) | DSA Signing Key | EC Agreement Key | EC Signing Key | HMAC Authentication Key | IKEv2 DF Secret | PBKDF Secret | RSA Signing Key | RSA Key Transport Key | SP 800-56C Concat. DF Secret | SP 800-108 KDF Secret | SRTP DF Secret | SSH DF Secret Value | TLS KDF Secret | Triple-DES Encryption Key | Triple-DES Decryption Key | Triple-DES Authentication Key | Triple-DES Wrapping Key | X9.63 KDF Secret Value | | |
| Initialize Module and Run Self-Tests on Demand | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Show Status | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Zeroize / Power-off | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | |
| Data Encryption | R | | | | | | | | | | | | | | | | | | | | | | R | | | | | | |
| Data Decryption | | R | | | | | | | | | | | | | | | | | | | | | | R | | | | | |
| MAC Calculation | | | R | | | | | | | | | | | | | | | | | | | | | | R | | | | |
| Signature Generation | | | | | | | | | | R | | R | | | | | R | | | | | | | | | | | | |
| Signature Verification | | | | | | | | | | R | | R | | | | | R | | | | | | | | | | | | |
| DRBG (SP 800-90A) output | G | G | G | G | G | G | G | G | G | G | G | G | | | | G | G | | | | | | G | G | G | G | | | |
| Message Hashing | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Keyed Message Hashing | | | | | | | | | | | | | R | | | | | | | | | | | | | | | | |
| TLS Key Derivation Function | | | | | | | | | | | | | | | | | | | | | | R | | | | | | | |
| SP 800-108 KBKDF | | | | | | | | | | | | | | | | | | | R | | | | | | | | | | |

| Services | CSPs | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|--------------------|--------------------|------------------------|------------------|------------------|----------------|-----------------------|-------------|-------------|-----------------|------------------|----------------|-------------------------|-----------------|--------------|-----------------|-----------------------|------------------------------|-----------------------|----------------|---------------------|----------------|---------------------------|---------------------------|-------------------------------|-------------------------|------------------------|---|
| | AES Encryption Key | AES Decryption Key | AES Authentication Key | AES Wrapping Key | DH Agreement Key | DRBG (CTR AES) | DRBG (CTR Triple-DES) | DRBG (Hash) | DRBG (HMAC) | DSA Signing Key | EC Agreement Key | EC Signing Key | HMAC Authentication Key | IKEv2 DF Secret | PBKDF Secret | RSA Signing Key | RSA Key Transport Key | SP 800-56C Concat. DF Secret | SP 800-108 KDF Secret | SRTP DF Secret | SSH DF Secret Value | TLS KDF Secret | Triple-DES Encryption Key | Triple-DES Decryption Key | Triple-DES Authentication Key | Triple-DES Wrapping Key | X9.63 KDF Secret Value | |
| SSH Derivation Function | | | | | | | | | | | | | | | | | | | | | R | | | | | | | |
| X9.63 Derivation Function | | | | | G | | | | | G | | | | | | G | | | | | | | | | | | | R |
| SP 800-56C Concatenation Derivation Function | | | | | G | | | | | G | | | | | | G | | R | | | | | | | | | | |
| IKEv2 Derivation Function | | | | | | | | | | | | | | R | | | | | | | | | | | | | | |
| SRTP Derivation Function | | | | | | | | | | | | | | | | | | | | R | | | | | | | | |
| PBKDF | | | | | | | | | | | | | G R | | R | | | | | | | | | | | | | |
| Key Agreement Schemes | G | G | G | G | R | | | | | R | | G | | | | | R | | | | | | G | G | G | G | | |
| Key Wrapping/ Transport | | | | R | | | | | | | | | R | | | | R | | | | | | | | | | R | |
| Key Unwrapping | | | | R | | | | | | | | | R | | | | R | | | | | | | | | | R | |
| NDRNG Callback | | | | | | G | G | G | G | | | | | | | | | | | | | | | | | | | |
| Utility | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

2.5 Physical Security

The module is a software-only module and does not have physical security mechanisms.

2.6 Operational Environment

The module operates in a modifiable operational environment under the FIPS 140-2 definitions.

The module runs on a GPC running one of the operating systems specified in the approved operational environment list in this section. Each approved operating system manages processes and threads in a logically separated manner. The module's user is considered the owner of the calling application that instantiates the module within the process space of the Java Virtual Machine.

The module optionally uses the Java Security Manager and starts in FIPS Approved mode by default when used with the Java Security Manager. When the module is not used within the context of the Java Security Manager, it will start by default in the non-Approved mode.

The module was tested on the following platforms:

Table 13 - Tested Environments

| Operating System | Hardware Platform | Processor (CPU) |
|---|---------------------|-----------------|
| VMware Photon OS 2.0 with JDK 11 on VMware ESXi 6.7 | Dell PowerEdge R830 | Intel Xeon E5 |

FIPS 140-2 validation compliance is maintained for other compatible operating systems (in single user mode) where the module source code is unmodified, and the requirements outlined in NIST IG G.5 are met. No claim can be made as to 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.

The module, when compiled from the same unmodified source code, is vendor-affirmed to be FIPS 140-2 compliant when running one of the Java SE Runtime environments on any of the following on the following supported single-user operating systems for which operational testing and algorithm testing were not performed:

- Microsoft Windows 2016
- Red Hat Enterprise Linux 7

2.6.1 Use of External RNG

The module makes use of the JVM's configured SecureRandom entropy source to provide entropy when required. The module will request entropy as appropriate to the security strength and seeding configuration for the DRBG that is using it and for the default DRBG will request a minimum of 256 bits

of entropy. In approved mode the minimum amount of entropy that can be requested by a DRBG is 112 bits. The module will wait until the `SecureRandom.generateSeed()` returns the requested amount of entropy, blocking if necessary.

2.7 Self-Tests

Each time the module is powered up, it tests that the cryptographic algorithms still operate correctly and that sensitive data has not been damaged. Power-up self-tests are available on demand by power cycling the module.

On power-up or reset, the module performs the self-tests that are described in Table 14 - Power-Up Self-Tests. All KATs must be completed successfully prior to any other use of cryptography by the module. If one of the KATs fails, the module enters the Self-Test Failure error state. The module will output a detailed error message when `FipsStatus.isReady()` is called. The error state can only be cleared by reloading the module and calling `FipsStatus.isReady()` again to confirm successful completion of the KATs.

2.7.1 Power-Up Self-Tests

Table 14 - Power-Up Self-Tests

| Test Target | Description |
|--------------------------|--|
| Software Integrity Check | HMAC-SHA-256 (HMAC Cert. #A2720) |
| AES | KATs: Encryption, Decryption Modes: ECB Key sizes: 128 bits |
| AES CCM | KATs: Generation, Verification Key sizes: 128 bits |
| AES CMAC | KATs: Generation, Verification Key sizes: 128 bits |
| AES GCM/GMAC | KATs: Generation, Verification Key sizes: 128 bits |
| DRBG | KATs: HASH_DRBG, HMAC_DRBG, CTR_DRBG Security Strengths: 256 bits |
| DSA | KAT: Signature Generation, Signature Verification Key sizes: 2048 bits |
| ECDSA | KAT: Signature Generation, Signature Verification Curves/Key sizes: P-256 |
| HMAC | KATs: Generation, Verification SHA sizes: SHA-256, SHA-512, SHA3-256 |
| KAS: FFC ²⁸ | KATs: Per IG 9.6 – Primitive “Z” Computation Parameter Sets/Key sizes: FB |

²⁸ Implemented by the module, though not required per IG D.1-rev3. The KAS is vendor affirmed to SP 800-56Ar3.

| | |
|-----------------------------------|---|
| KAS: ECC ²⁹ | KATs: Per IG 9.6 – Primitive “Z” Computation Parameter Sets/Key sizes: EC |
| KBKDF (SP 800-108) | KATs: Per IG 9.4 – Output Verification Modes: Counter, Feedback, Double Pipeline PRFs: AES-CMAC, Triple-DES-CMAC, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256 |
| RSA | KATs: Signature Generation, Signature Verification Key sizes: 2048 bits |
| RSA, Key Transport | KATs: SP 800-56B specific KATs per IG D.4 Key sizes: 2048 bits |
| RSA, Key Wrapping | KATs: SP 800-56B specific KATs per IG D.4 Key sizes: 2048 bits |
| SHS | KATs: Output Verification SHA sizes: SHA-1, SHA-256, SHA-512 |
| Triple-DES | KATs: Encryption, Decryption Modes: ECB Key sizes: 3-Key |
| Triple-DES CMAC | KATs: Generation, Verification Key sizes: 3-Key |
| XOF (Extendable-Output functions) | KATs: Output Verification XOFs: SHAKE256 |

2.7.2 Conditional Self-Tests

The module implements the following conditional self-tests upon key generation, or random number generation (respectively):

Table 15 - Conditional Self-Tests

| Test Target | Description |
|--------------------|--|
| DRBG | DRBG Continuous Test performed when a random value is requested from the DRBG. |
| DRBG Health Checks | Performed conditionally on DRBG, per SP 800-90A Section 11.3. |
| DSA | DSA Pairwise Consistency Test performed on every DSA key pair generation. |
| ECDSA | ECDSA Pairwise Consistency Test performed on every EC key pair generation. |
| KAS: DH | DH Pairwise Consistency Test performed on every DH key pair generation. |
| KAS: ECDH/ECCDH | EC DH Pairwise Consistency Test performed on every ECDH/ECCDH key pair generation. |

²⁹ Implemented by the module, though not required per IG D.1-rev3. The KAS is vendor affirmed to SP 800-56Ar3.

| Test Target | Description |
|--|--|
| KAS: SP 800-56A Assurances ³⁰ | Performed conditionally per SP 800-56A Sections 5.5.2, 5.6.2, and/or 5.6.3 |
| NDRNG | NDRNG Continuous Test performed when a random value is requested from the NDRNG. |
| RSA | RSA Pairwise Consistency Test performed on every RSA key pair generation. |

2.8 Mitigation of Other Attacks

The module implements basic protections to mitigate against timing-based attacks against its internal implementations. There are two countermeasures used.

The first countermeasure is Constant Time Comparisons, which protect the digest and integrity algorithms by strictly avoiding “fast fail” comparison of MACs, signatures, and digests so the time taken to compare a MAC, signature, or digest is constant regardless of whether the comparison passes or fails.

The second countermeasure is made up of Numeric Blinding and decryption/signing verification which both protect the RSA algorithm.

Numeric Blinding prevents timing attacks against RSA decryption and signing by providing a random input into the operation which is subsequently eliminated when the result is produced. The random input makes it impossible for a third party observing the private key operation to attempt a timing attack on the operation as they do not have knowledge of the random input and consequently the time taken for the operation tells them nothing about the private value of the RSA key.

Decryption/signing verification is carried out by calculating a primitive encryption or signature verification operation after a corresponding decryption or signing operation before the result of the decryption or signing operation is returned. The purpose of this is to protect against Lenstra's CRT attack by verifying the correctness of the private key calculations involved. Lenstra's CRT attack takes advantage of undetected errors in the use of RSA private keys with CRT values and, if exploitable, can be used to discover the private value of the RSA key.

³⁰ Implemented by the module, though not required per IG D.1-rev3. The KAS is vendor affirmed to SP 800-56Ar3.

3 Security Rules and Guidance

3.1 Basic Enforcement

The module design corresponds to the module security rules. This section documents the security rules enforced by the cryptographic module to implement the security requirements of this FIPS 140-2 Level 1 module.

1. The module provides two distinct operator roles: User and Crypto Officer.
2. The module does not provide authentication.
3. The operator may command the module to perform the power up self-tests by cycling power or resetting the module.
4. Power-up self-tests do not require any operator action.
5. Data output is inhibited during self-tests, zeroization, and error states. Output related to keys and their use is inhibited until the key concerned has been fully generated.
6. Status information does not contain CSPs or sensitive data that if misused could lead to a compromise of the module.
7. There are no restrictions on which keys or CSPs are zeroized by the zeroization service.
8. The module does not support concurrent operators.
9. The module does not have any external input/output devices used for entry/output of data.
10. The module does not enter or output plaintext CSPs from the module's physical boundary.
11. The module does not output intermediate key values.

3.2 Additional Enforcement with a Java SecurityManager

In the presence of a Java SecurityManager FIPS Approved mode services specific to a context (such as DSA and ECDSA for use in TLS) require specific policy permissions to be configured in the JVM configuration by the Crypto Officer or User. The Java SecurityManager can also be used to restrict the ability of particular code bases to examine CSPs. See Section 2.1.3 –Module Configuration for further advice on this.

In the absence of a Java SecurityManager specific services related to protocols such as TLS are available, however must only be used in relation to those protocols.

3.3 Basic Guidance

The jar file representing the module needs to be installed in a JVM's class path in a manner appropriate to its use in applications running on the JVM.

Functionality in the module is provided in two ways. At the lowest level there are distinct classes that provide access to the FIPS Approved and non-Approved services provided by the module. A more

abstract level of access can also be gained using strings providing operation names passed into the module's Java cryptography provider through the APIs described in the Java Cryptography Architecture (JCA) and the Java Cryptography Extension (JCE).

When the module is being used in FIPS Approved-only mode, classes providing implementations of algorithms which are not FIPS Approved, or allowed, are explicitly disabled.

3.4 Enforcement and Guidance for AES GCM IVs

IVs for GCM can be generated randomly or via a `FipsNonceGenerator`. Where an IV is not generated within the module, the module supports the importing of GCM IVs.

In FIPS Approved mode, when a GCM IV is generated randomly, the module enforces the use of an approved DRBG in line with Section 8.2.2 of SP 800-38D.

In FIPS approved mode, when a GCM IV is generated using the `FipsNonceGenerator`, a counter is used as the basis for the nonce. Rollover of the counter in the `FipsNonceGenerator` will result in an `IllegalStateException` indicating the `FipsNonceGenerator` is exhausted. Per IG A.5, where the AES GCM IV is used for TLS, rollover will terminate any TLS session in process using the current key. The exception can only be recovered from by using a new handshake and creating a new `FipsNonceGenerator`.

In FIPS Approved mode, importing a GCM IV for encryption that originates from outside the module is non-conformant unless the source of the IV is also FIPS approved for GCM IV generation.

Per IG A.5, Section 2.2.1 of this Security Policy also states that in the event module power is lost and restored, the consuming application must ensure that any of its AES GCM keys used for encryption or decryption are re-distributed.

3.5 Enforcement and Guidance for Use of the Approved PBKDF

In line with the requirements for SP 800-132, keys generated using the approved PBKDF must only be used for storage applications. Any other use of the approved PBKDF is non-conformant.

In FIPS Approved mode the module enforces that any password used must encode to at least 14 bytes (112 bits) and that the salt is at least 16 bytes (128 bits) long. The iteration count associated with the PBKDF should be as large as practical.

As the module is a general purpose software module, it is not possible to anticipate all the levels of use for the PBKDF, however a user of the module should also note that a password should at least contain enough entropy to be unguessable and also contain enough entropy to reflect the security strength required for the key being generated. In the event a password encoding is simply based on ASCII, a 14-byte password is unlikely to contain sufficient entropy for most purposes. Users are referred to

Appendix A, "Security Considerations" in SP 800-132 for further information on password, salt, and iteration count selection.

For users interested in introducing memory hardness as a layer on top of the PBKDF, the script augmentation to PBKDF based on HMAC-SHA-256 (as described in RFC 7914) is also available.

3.6 Rules for Setting the N and the S String in cSHAKE

To customize the output of the cSHAKE function, the cSHAKE algorithm permits the operator to input strings for the Function-Name input (N) and the Customization String (S).

The Function-Name input (N) is reserved for values specified by NIST and should only be set to the appropriate NIST specified value. Any other use of N is non-conformant.

The Customization String (S) is available to allow users to customize the cSHAKE function as they wish. The length of S is limited to the available size of a byte array in the JVM running the module.

3.7 Guidance for the Use of DRBGs and Configuring the JVM's Entropy Source

A user can instantiate the default Approved DRBG for the module explicitly by using `SecureRandom.getInstance("DEFAULT", "CCJ")`, or by using a `CryptoComplyFipsProvider` object instead of the provider name as appropriate. This will seed the Approved DRBG from the live entropy source of the JVM, for example `/dev/random` on the tested Linux operational environments, with an appropriate number of bits of entropy for the security level of the default Approved DRBG configured for the module.

An additional option is available using the Approved Hash_DRBG and the process outlined in SP 800-90A, Section 8.6.5. The provider can be configured to use an DRBG chain based on a SHA-512 SP 800-90A DRBG as the internal (source) DRBG providing a seed generation for the external (target) DRBG. To configure this use: `"C:HYBRID;ENABLE{All};"`

The two DRBGs are instantiated in a chain as a "Source DRBG" to seed the "Target DRBG" in accordance with Section 7 of Draft NIST SP 800-90C, where the Target DRBG is the default Approved DRBG used by the module.

The initial seed and the subsequent reseeds for the DRBG chain come from the live entropy source configured for the JVM. The DRBG chain will reseed automatically by pausing for 20 requests (which will usually equate to 5120 bits). An entropy gathering thread reseeds the DRBG chain when it has gathered sufficient entropy (currently 256 bits) from the live entropy source. Once reseeded, the request counter is reset and the reseed process begins again.

The “Source DRBG” in the chain is internal to the module and inaccessible to the user to ensure it is only used for generating seeds for the default Approved DRBG of the module.

The user shall ensure that the Approved entropy source is configured per Section 6.1 of this Security Policy and will block, or fail, if it is unable to provide the amount of entropy requested.

3.8 Software Installation

The module is provided directly to solution developers and is not available for direct download to the general public. Only the compiled module is provided to solution developers. The module and its host application are to be installed on an operating system specified in Section 2.6 or on an operating system where portability is maintained.

4 References and Acronyms

4.1 References

Table 16 – References

| Abbreviation | Full Specification Name |
|--------------|---|
| ANSI X9.31 | X9.31-1998, Digital Signatures using Reversible Public Key Cryptography for the Financial Services Industry (rDSA), September 9, 1998 |
| FIPS 140-2 | Security Requirements for Cryptographic modules, May 25, 2001 |
| FIPS 180-4 | Secure Hash Standard (SHS) |
| FIPS 186-2 | Digital Signature Standard (DSS) |
| FIPS 186-4 | Digital Signature Standard (DSS) |
| FIPS 197 | Advanced Encryption Standard |
| FIPS 198-1 | The Keyed-Hash Message Authentication Code (HMAC) |
| FIPS 202 | SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions |
| IG | Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program |
| PKCS#1 v2.1 | RSA Cryptography Standard |
| PKCS#5 | Password-Based Cryptography Standard |
| PKCS#12 | Personal Information Exchange Syntax Standard |
| SP 800-38A | Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode |
| SP 800-38B | Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication |
| SP 800-38C | Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality |
| SP 800-38D | Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC |
| SP 800-38F | Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping |
| SP 800-56Ar3 | Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography |
| SP 800-56B | Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography |
| SP 800-56C | Recommendation for Key Derivation through Extraction-then- Expansion |
| SP 800-67 | Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher |
| SP 800-89 | Recommendation for Obtaining Assurances for Digital Signature Applications |
| SP 800-90A | Recommendation for Random Number Generation Using Deterministic Random Bit Generators |
| SP 800-108 | Recommendation for Key Derivation Using Pseudorandom Functions |

| | |
|---------------|---|
| SP 800-131Ar2 | Transitioning the Use of Cryptographic Algorithms and Key Lengths |
| SP 800-132 | Recommendation for Password-Based Key Derivation |
| SP 800-133 | Recommendation for Cryptographic Key Generation |
| SP 800-135 | Recommendation for Existing Application–Specific Key Derivation Functions |
| SP 800-185 | SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash |

4.2 Acronyms

The following table defines acronyms found in this document:

Table 17 - Acronyms and Terms

| Acronym | Term |
|----------|---|
| AES | Advanced Encryption Standard |
| API | Application Programming Interface |
| CBC | Cipher-Block Chaining |
| CCM | Counter with CBC-MAC |
| CCCS | Canadian Centre for Cyber Security |
| CDH | Computational Diffie-Hellman |
| CFB | Cipher Feedback Mode |
| CMAC | Cipher-based Message Authentication Code |
| CMVP | Cryptographic Module Validation Program |
| CO | Crypto Officer |
| CPU | Central Processing Unit |
| CS | Ciphertext Stealing |
| CSP | Critical Security Parameter |
| CTR | Counter Mode |
| CVL | Component Validation List |
| DES | Data Encryption Standard |
| DH | Diffie-Hellman |
| DRAM | Dynamic Random Access Memory |
| DRBG | Deterministic Random Bit Generator |
| DSA | Digital Signature Algorithm |
| DSTU4145 | Ukrainian DSTU-4145-2002 Elliptic Curve Scheme |
| EC | Elliptic Curve |
| ECB | Electronic Code Book |
| ECC | Elliptic Curve Cryptography |
| ECDSA | Elliptic Curve Digital Signature Algorithm |
| EdDSA | Edwards Curve DSA using Ed25519, Ed448 |
| EMC | Electromagnetic Compatibility |
| EMI | Electromagnetic Interference |
| FIPS | Federal Information Processing Standard |
| GCM | Galois/Counter Mode |
| GMAC | Galois Message Authentication Code |
| GOST | Gosudarstvennyi Standard Soyuz SSR/Government Standard of the Union of Soviet Socialist Republics |
| GPC | General Purpose Computer |
| HMAC | (Keyed-) Hash Message Authentication Code |
| IG | Implementation Guidance |
| IV | Initialization Vector |
| JAR | Java ARchive |
| JCA | Java Cryptography Architecture |

| | |
|--------|---|
| JCE | Java Cryptography Extension |
| JDK | Java Development Kit |
| JRE | Java Runtime Environment |
| JVM | Java Virtual Machine |
| KAS | Key Agreement Scheme |
| KAT | Known Answer Test |
| KDF | Key Derivation Function |
| KW | Key Wrap |
| KWP | Key Wrap with Padding |
| MAC | Message Authentication Code |
| MD5 | Message Digest algorithm MD5 |
| N/A | Not Applicable |
| NDRNG | Non Deterministic Random Number Generator |
| OCB | Offset Codebook Mode |
| OFB | Output Feedback |
| OS | Operating System |
| PBKDF | Password-Based Key Derivation Function |
| PKCS | Public-Key Cryptography Standards |
| PQG | Diffie-Hellman Parameters P, Q and G |
| RC | Rivest Cipher, Ron's Code |
| RIPEMD | RACE Integrity Primitives Evaluation Message Digest |
| RSA | Rivest, Shamir, and Adleman |
| SHA | Secure Hash Algorithm |
| TCBC | TDEA Cipher-Block Chaining |
| TCFB | TDEA Cipher Feedback Mode |
| TDEA | Triple Data Encryption Algorithm |
| TDES | Triple Data Encryption Standard |
| TECB | TDEA Electronic Codebook |
| TOFB | TDEA Output Feedback |
| TLS | Transport Layer Security |
| USB | Universal Serial Bus |
| XDH | Edwards Curve Diffie-Hellman using X25519, X448 |
| XOF | Extendable-Output Function |