



## FIPS 140-2 Non-Proprietary Security Policy

---

### CryptoComply | Java

Software Version 2.2-fips

Document Version 1.0

August 20, 2015



SafeLogic Inc.

530 Lytton Avenue, Ste. 200

Palo Alto, CA 94301

[www.safelogic.com](http://www.safelogic.com)

## Abstract

This document provides a non-proprietary FIPS 140-2 Security Policy for CryptoComply | Java.

## Table of Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>                               | <b>4</b>  |
| 1.1      | About FIPS 140                                    | 4         |
| 1.2      | About this Document                               | 4         |
| 1.3      | External Resources                                | 4         |
| 1.4      | Notices   | 4         |
| 1.5      | Acronyms  | 4         |
| <b>2</b> | <b>CryptoComply   Java</b>                        | <b>6</b>  |
| 2.1      | Cryptographic Module Specification                | 6         |
| 2.1.1    | Validation Level Detail                           | 6         |
| 2.1.2    | Approved Cryptographic Algorithms                 | 7         |
| 2.1.3    | Non-Approved but Allowed Cryptographic Algorithms | 8         |
| 2.1.4    | Non-Approved Cryptographic Algorithms             | 8         |
| 2.2      | Module Interfaces                                 | 11        |
| 2.3      | Roles, Services, and Authentication               | 12        |
| 2.3.1    | Operator Services and Descriptions                | 12        |
| 2.3.2    | Operator Authentication                           | 14        |
| 2.4      | Physical Security                                 | 14        |
| 2.5      | Operational Environment                           | 14        |
| 2.6      | Cryptographic Key Management                      | 15        |
| 2.6.1    | Random Number Generation                          | 17        |
| 2.6.2    | Key/CSP Storage                                   | 17        |
| 2.6.3    | Key/CSP Zeroization                               | 17        |
| 2.7      | Self-Tests  | 17        |
| 2.7.1    | Power-On Self-Tests                               | 17        |
| 2.7.2    | Conditional Self-Tests                            | 18        |
| 2.8      | Mitigation of Other Attacks                       | 18        |
| <b>3</b> | <b>Guidance and Secure Operation</b>              | <b>19</b> |
| 3.1      | Crypto Officer Guidance                           | 19        |
| 3.1.1    | Software Installation                             | 19        |
| 3.1.2    | Additional Rules of Operation                     | 19        |
| 3.2      | User Guidance                                     | 19        |
| 3.2.1    | General Guidance                                  | 19        |
| 3.2.2    | FIPS-Approved Mode of Operation                   | 20        |

## List of Tables

|  |    |
|--|----|
| Table 1 – Acronyms and Terms.....                              | 5  |
| Table 2 – Validation Level by FIPS 140-2 Section.....          | 6  |
| Table 3 – FIPS-Approved Algorithm Certificates.....            | 8  |
| Table 4 - Non Approved Algorithms .....                        | 10 |
| Table 5 – Logical Interface / Physical Interface Mapping ..... | 12 |
| Table 6 – Module Services, Roles, and Descriptions.....        | 13 |
| Table 7 – Module Keys/CSPs.....                                | 16 |
| Table 8 – Power-On Self-Tests .....                            | 18 |
| Table 9 – Conditional Self-Tests.....                          | 18 |

## List of Figures

|   |    |
|---|----|
| Figure 1 – Module Boundary and Interfaces Diagram ..... | 11 |
|---|----|

## 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 Communications Security Establishment (CSE) Cryptographic Module Validation Program (CMVP) run the FIPS 140 program. The NVLAP accredits independent testing labs to perform FIPS 140-2 testing; the CMVP validates modules meeting FIPS 140-2 validation. *Validated* is the term given to a module that is documented and tested against the FIPS 140-2 criteria.

More information is available on the CMVP website at <http://csrc.nist.gov/groups/STM/cmvp/index.html>.

### 1.2 About this Document

This non-proprietary Cryptographic Module Security Policy for the CryptoComply | Java from SafeLogic provides an overview of the product and a high-level description of how it meets the security requirements of FIPS 140-2. This document contains details on the module's cryptographic keys and critical security parameters. This Security Policy concludes with instructions and guidance on running the module in a FIPS 140-2 mode of operation.

CryptoComply | Java may also be referred to as the “module” in this document.

### 1.3 External Resources

The SafeLogic website (<http://www.safelogic.com>) contains information on SafeLogic services and products. The Cryptographic Module Validation Program website contains links to the FIPS 140-2 certificate and SafeLogic contact information.

### 1.4 Notices

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

### 1.5 Acronyms

The following table defines acronyms found in this document:

| Acronym    | Term   |
|------------|--|
| AES        | Advanced Encryption Standard                   |
| ANSI       | American National Standards Institute          |
| API        | Application Programming Interface              |
| CMVP       | Cryptographic Module Validation Program        |
| CO         | Crypto Officer                                 |
| CSE        | Communications Security Establishment          |
| CSP        | Critical Security Parameter                    |
| DES        | Data Encryption Standard                       |
| DH         | Diffie-Hellman                                 |
| DSA        | Digital Signature Algorithm                    |
| EC         | Elliptic Curve                                 |
| EMC        | Electromagnetic Compatibility                  |
| EMI        | Electromagnetic Interference                   |
| FCC        | Federal Communications Commission              |
| FIPS       | Federal Information Processing Standard        |
| GPC        | General Purpose Computer                       |
| GUI        | Graphical User Interface                       |
| HMAC       | (Keyed-) Hash Message Authentication Code      |
| KAT        | Known Answer Test                              |
| MAC        | Message Authentication Code                    |
| NIST       | National Institute of Standards and Technology |
| OS         | Operating System                               |
| PKCS       | Public-Key Cryptography Standards              |
| PRNG       | Pseudo Random Number Generator                 |
| PSS        | Probabilistic Signature Scheme                 |
| RNG        | Random Number Generator                        |
| RSA        | Rivest, Shamir, and Adleman                    |
| SHA        | Secure Hash Algorithm                          |
| SSL        | Secure Sockets Layer                           |
| Triple-DES | Triple Data Encryption Algorithm               |
| TLS        | Transport Layer Security                       |
| USB        | Universal Serial Bus                           |

Table 1 – Acronyms and Terms

## 2 CryptoComply | Java

### 2.1 Cryptographic Module Specification

CryptoComply | Java is a standards-based “Drop-in Compliance” cryptographic engine for native Java environments. The module delivers core cryptographic functions to mobile platforms and features robust algorithm support, including Suite B algorithms. CryptoComply offloads functions for secure key management, data integrity, data at rest encryption, and secure communications to a trusted implementation.

The module's logical cryptographic boundary is the shared library files and their integrity check HMAC files (cryptocomply-2.2-fips.jar and cryptocomply-2.2-fips.hmac). The module is a multi-chip standalone embodiment installed on a General Purpose Device. The module is a software module and relies on the physical characteristics of the host platform. The module's physical cryptographic boundary is defined by the enclosure around the host platform.

All operations of the module occur via calls from host applications and their respective internal daemons/processes.

#### 2.1.1 Validation Level Detail

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

| 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                                  | N/A              |

Table 2 – Validation Level by FIPS 140-2 Section

### 2.1.2 Approved Cryptographic Algorithms

The module’s cryptographic algorithm implementations have received the following certificate numbers from the Cryptographic Algorithm Validation Program:

| Algorithm   | CAVP Certificate |
|---|------------------|
| AES (128-, 192-, 256-bit keys in ECB, CBC, CFB128 and OFB modes)  | 3192             |
| DSA (FIPS 186-4) <ul style="list-style-type: none"> <li>• Signature verification                             <ul style="list-style-type: none"> <li>○ L=1024, N=160, SHA-1 through SHA-512</li> <li>○ L=2048, N=224, 256, SHA-1 through SHA-512</li> <li>○ L=3072, N=256, SHA-1 through SHA-512</li> </ul> </li> <li>• PQG generation (Probable Primes P and Q, Unverifiable and Canonical Generation G)                             <ul style="list-style-type: none"> <li>○ L=2048, N=224, SHA-224 through SHA-512</li> <li>○ L=2048, N=256, SHA-256 through SHA-512</li> <li>○ L=3072, N=256, SHA-256 through SHA-512</li> </ul> </li> <li>• Key Pair Generation                             <ul style="list-style-type: none"> <li>○ L=2048, N=224</li> <li>○ L=2048, N=256</li> <li>○ L=3072, N=256</li> </ul> </li> <li>• Signature Generation                             <ul style="list-style-type: none"> <li>○ L=2048, N=224, SHA-224 through SHA-512</li> <li>○ L=2048, N=256, SHA-256 through SHA-512</li> <li>○ L=3072, N=256, SHA-256 through SHA-512</li> </ul> </li> </ul> | 914              |
| ECDSA (FIPS 186-4) <ul style="list-style-type: none"> <li>• Signature Verification (SHA-1 through SHA-512)                             <ul style="list-style-type: none"> <li>○ P–curves 192, 224, 256, 384, and 521</li> <li>○ K–curves 163, 233, 283, 409, and 571</li> <li>○ B–curves 163, 233, 283, 409, and 571</li> </ul> </li> <li>• Signature Generation (SHA-224 through SHA-512)                             <ul style="list-style-type: none"> <li>○ P–curves 224, 256, 384, and 521</li> <li>○ K–curves 233, 283, 409, and 571</li> <li>○ B–curves 233, 283, 409, and 571</li> </ul> </li> </ul>  | 583              |
| RSA (FIPS 186-4) <ul style="list-style-type: none"> <li>• Key Pair Generation (X9.31)                             <ul style="list-style-type: none"> <li>○ Appendix B.3.3</li> <li>○ Mod 2048, 3072</li> <li>○ Table C.3 Probabilistic Primality Tests (<math>2^{100}</math>)</li> </ul> </li> <li>• Signature Generation (PKCS v1.5, PSS)                             <ul style="list-style-type: none"> <li>○ Mod 2048, SHA-224 through SHA-512</li> <li>○ Mod 3072, SHA-224 through SHA-512</li> </ul> </li> </ul>   | 1622             |

| Algorithm  | CAVP Certificate |
|--|------------------|
| <ul style="list-style-type: none"> <li>• Signature Verification (PKCS v1.5, PSS)                             <ul style="list-style-type: none"> <li>○ Mod 1024, SHA-1 through SHA-512</li> <li>○ Mod 2048, SHA-1 through SHA-512</li> <li>○ Mod 3072, SHA-1 through SHA-512</li> </ul> </li> </ul> |                  |
| HMAC using SHA-1, SHA-224, SHA-256, SHA-384, SHA-512   | 2011             |
| SHA-1, SHA-224, SHA-256, SHA-384, SHA-512  | 2637             |
| SP 800-90A based HMAC-DRBG, no reseed  | 668              |
| Triple-DES (two- and three-key with ECB, CBC, CFB8 and OFB modes) <sup>1</sup>   | 1818             |

**Table 3 – FIPS-Approved Algorithm Certificates**

### 2.1.3 Non-Approved but Allowed Cryptographic Algorithms

The module supports the following non-FIPS 140-2 approved but allowed algorithms:

- Diffie-Hellman (key agreement; key establishment methodology provides between 112 and 219 bits of encryption strength)
- EC Diffie-Hellman (key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength)

### 2.1.4 Non-Approved Cryptographic Algorithms

The module supports the following non-approved algorithms and modes:

| Algorithm          | Modes or Cipher Type  |
|--------------------|---|
| DSA <sup>2</sup>   | PQGen, KeyGen and SigGen; non-compliant less than 112 bits of encryption strength) including FIPS 186-2 signature generation and key generation |
| ECDSA <sup>2</sup> | KeyGen and SigGen; non-compliant less than 112 bits of encryption strength) including FIPS 186-2 signature generation and key generation        |
| RSA <sup>2</sup>   | KeyGen and SigGen; non-compliant less than 112 bits of encryption strength  |
| Diffie-Hellman     | key agreement; key establishment methodology providing between 80 and 112 bits of encryption strength   |
| EC Diffie-Hellman  | key agreement; key establishment methodology provides between 80 and 112 bits of encryption strength  |
| AES <sup>2</sup>   | GCM, CFB8, CTR, CMAC, CCM   |

<sup>1</sup> The use of two-key Triple DES for encryption is restricted: the total number of blocks of data encrypted with the same cryptographic key shall not be greater than 2<sup>20</sup>

<sup>2</sup> Non-compliant



| Algorithm                                   | Modes or Cipher Type   |
|---|--|
| ANSI X9.31 Appendix A.2.4 PRNG <sup>2</sup> | (AES-128)  |
| Blowfish                                    | SymmetricBlockCipher <sup>3</sup>  |
| Camellia                                    | SymmetricBlockCipher <sup>3</sup>  |
| CAST5                                       | SymmetricBlockCipher <sup>3</sup>  |
| CAST6                                       | SymmetricBlockCipher <sup>3</sup>  |
| ChaCha                                      | SymmetricStreamCipher <sup>4</sup>   |
| DES   | SymmetricBlockCipher <sup>3</sup>  |
| TDES Key Wrapping <sup>2</sup>              | SymmetricBlockCipher <sup>3</sup>  |
| ElGamal                                     | AsymmetricBlockCipher <sup>5</sup>   |
| GOST28147                                   | SymmetricBlockCipher <sup>3</sup>  |
| GOST3411                                    | Digest   |
| Grain128                                    | SymmetricStreamCipher <sup>4</sup>   |
| Grainv1                                     | SymmetricStreamCipher <sup>4</sup>   |
| HC128                                       | SymmetricStreamCipher <sup>4</sup>   |
| HC256                                       | SymmetricStreamCipher <sup>4</sup>   |
| IDEA  | SymmetricBlockCipher <sup>3</sup>  |
| IES   | Key Agreement and Stream Cipher based on IEEE P1363a (draft 10)  |
| ISAAC                                       | SymmetricStreamCipher <sup>4</sup>   |
| MD2   | Digest   |
| MD4   | Digest   |
| MD5   | Digest   |
| Naccache Stern                              | AsymmetricBlockCipher <sup>5</sup>   |
| Noekeon                                     | SymmetricBlockCipher <sup>3</sup>  |
| Password-Based-Encryption (PBE)             | <ul style="list-style-type: none"> <li>• PKCS5S1, any Digest, any symmetric Cipher, ASCII</li> <li>• PKCS5S2, SHA1/HMac, any symmetric Cipher, ASCII, UTF8</li> <li>• PKCS12, any Digest, any symmetric Cipher, Unicode</li> </ul> |
| RC2   | SymmetricBlockCipher <sup>3</sup>  |
| RC2 Key Wrapping                            | SymmetricStreamCipher <sup>4</sup>   |
| RC4   | SymmetricStreamCipher <sup>4</sup>   |
| RC532                                       | SymmetricBlockCipher <sup>3</sup>  |
| RC564                                       | SymmetricBlockCipher <sup>3</sup>  |
| RC6   | SymmetricBlockCipher <sup>3</sup>  |
| RFC3211 Wrapping                            | SymmetricBlockCipher <sup>3</sup>  |

<sup>3</sup> Symmetric Block Ciphers can be used with the following modes and padding: ECB, CBC, CFB, CCM, CTS, GCM, GCF, EAX, OCB, OFB, CTR, OpenPGPCFB, GOST OFB, AEAD-CCM, AEAD-EAX, AEAD-GCM, AEAD-OCB, PKCS7Padding, ISO10126d2Padding, ISO7816d4Padding, X932Padding, ISO7816d4Padding, ZeroBytePadding, TBCEPadding

<sup>4</sup> Symmetric Stream Ciphers can only be used with ECB mode.

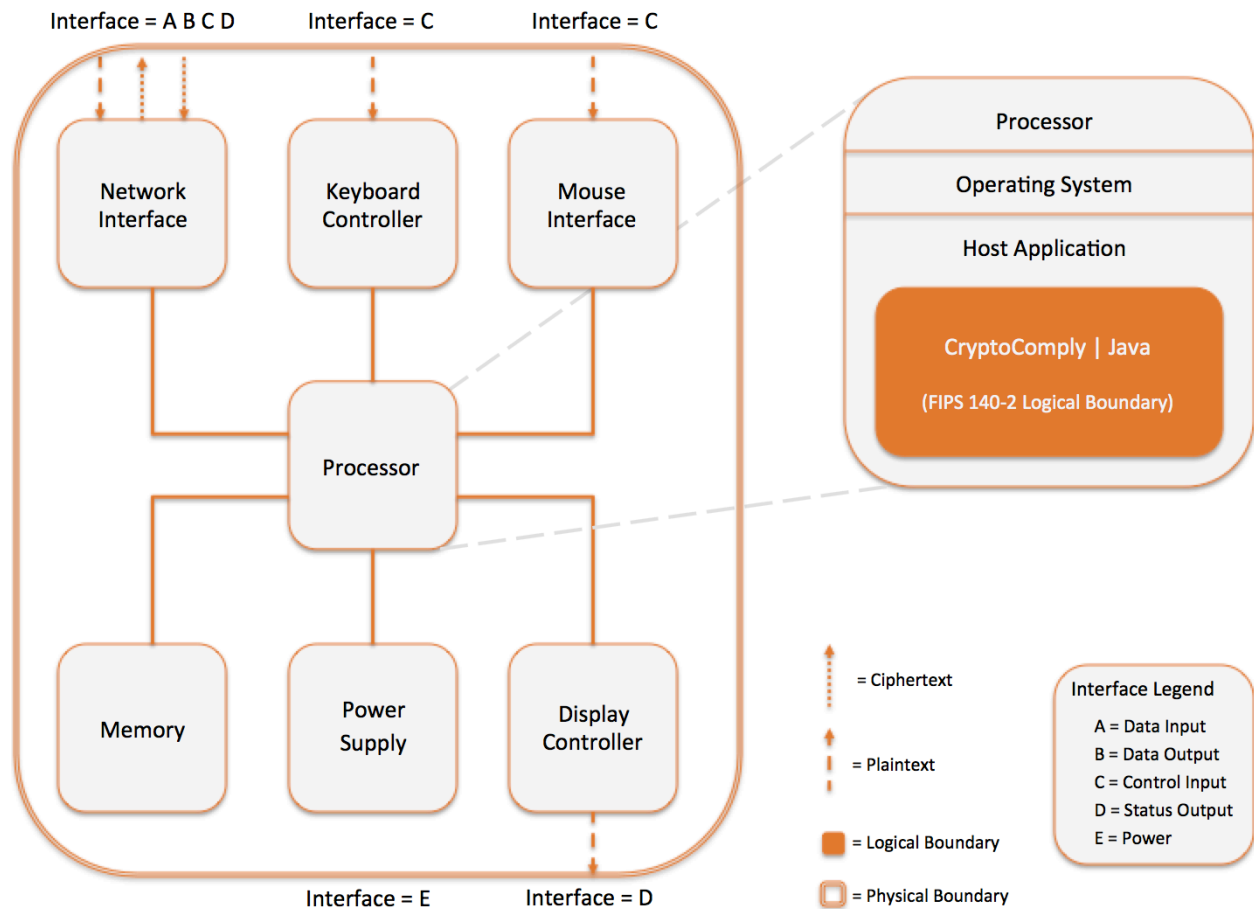
<sup>5</sup> Asymmetric Block Ciphers can be used with ECB mode and the following encodings: OAEP, PKCS1, ISO9796d1

| Algorithm                    | Modes or Cipher Type               |
|------------------------------|------------------------------------|
| RFC3394 Wrapping             | SymmetricBlockCipher <sup>3</sup>  |
| Rijndael                     | SymmetricBlockCipher <sup>3</sup>  |
| Ripe MD128                   | Digest                             |
| Ripe MD160                   | Digest                             |
| Ripe MD256                   | Digest                             |
| Ripe MD320                   | Digest                             |
| RSA Encryption               | AsymmetricBlockCipher <sup>5</sup> |
| Salsa 20                     | SymmetricStreamCipher <sup>4</sup> |
| SEED                         | SymmetricBlockCipher <sup>3</sup>  |
| SEED Wrapping                | SymmetricBlockCipher <sup>3</sup>  |
| Serpent                      | SymmetricBlockCipher <sup>3</sup>  |
| Shacal2                      | SymmetricBlockCipher <sup>3</sup>  |
| SHA-3 <sup>2</sup>           | Digest                             |
| SHA-512/t <sup>2</sup>       | Digest                             |
| Skein-256-*                  | Digest                             |
| Skein-512-*                  | Digest                             |
| Skein-1024-*                 | Digest                             |
| Skipjack <sup>2</sup>        | SymmetricBlockCipher <sup>3</sup>  |
| SP 800-90A DRBG <sup>2</sup> | CTR, Hash                          |
| TEA                          | SymmetricBlockCipher <sup>3</sup>  |
| TDES <sup>2</sup>            | CFB64                              |
| Threefish                    | SymmetricBlockCipher <sup>3</sup>  |
| Tiger                        | Digest                             |
| TLS v1.0 KDF <sup>2</sup>    | Key Derivation Function            |
| Twofish                      | SymmetricBlockCipher <sup>3</sup>  |
| VMPC                         | SymmetricStreamCipher <sup>4</sup> |
| Whirlpool                    | Digest                             |
| XSalsa20                     | SymmetricStreamCipher <sup>4</sup> |
| XTEAEngine                   | SymmetricBlockCipher <sup>3</sup>  |

Table 4 - Non Approved Algorithms

## 2.2 Module Interfaces

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



**Figure 1 – Module Boundary and Interfaces Diagram**

The interfaces (ports) for the physical boundary include the computer keyboard port, mouse port, network port, USB ports, display and power plug. 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 and are provided through the Application Programming Interface (API) that a calling daemon can operate. The logical interfaces expose services that applications directly call, and the API provides functions that may be called by a referencing application (see Section 2.3 – 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: data input, data output, control input, and status output. Each of the FIPS 140- 2 logical interfaces relates to the module’s callable interface, as follows:

| FIPS 140-2 Interface | Logical Interface  | Module Physical Interface           |
|----------------------|--|-------------------------------------|
| Data Input           | Input parameters of API function calls   | Network Interface                   |
| Data Output          | Output parameters of API function calls  | Network Interface                   |
| Control Input        | API function calls   | Keyboard Interface, Mouse Interface |
| Status Output        | Function calls returning status information and return codes provided by API function calls. | Display Controller                  |
| Power                | None   | Power Supply                        |

**Table 5 – Logical Interface / Physical Interface Mapping**

As shown in Figure 1 – Module Boundary and Interfaces Diagram and Table 6 – Module Services, Roles, and Descriptions, the output data path is provided by the data interfaces and is logically disconnected from processes performing key generation or zeroization. No key information will be output through the data output interface when the module zeroizes keys.

## 2.3 Roles, Services, and Authentication

The module supports a Crypto Officer and a User role. The module does not support a Maintenance role. The User and Crypto-Officer roles are implicitly assumed by the entity accessing services implemented by the Module.

### 2.3.1 Operator Services and Descriptions

The module supports services that are available to users in the various roles. All of the services are described in detail in the module’s user documentation. The following table shows the services available to the various roles and the access to cryptographic keys and CSPs resulting from services:

| Service                  | Roles | CSP / Algorithm   | Permission |
|--------------------------|-------|---|------------|
| Initialize module        | CO    | None  | None       |
| Show status              | CO    | None  | None       |
| Run self-tests on demand | CO    | None  | None       |
| Zeroize key              | CO    | AES key<br>DH components<br>DRBG Entropy<br>DRBG Seed<br>DSA private/public key | Write      |

| Service  | Roles | CSP / Algorithm   | Permission     |
|--|-------|---|----------------|
|  |       | ECDH components<br>ECDSA private/public key<br>HMAC key<br>RSA private/public key<br>Triple-DES key |                |
| Generate asymmetric key pair                   | User  | RSA private/public key<br>DSA private/public key  | Write          |
| Generate keyed hash (HMAC)                     | User  | HMAC key  | Read / Execute |
| Generate message digest (SHS <sup>6</sup> )    | User  | None  | None           |
| Generate random number and load entropy (DRBG) | User  | DRBG Seed<br>DRBG Entropy   | Read / Execute |
| Key agreement                                  | User  | DH components<br>ECDH components  | Write          |
| Signature Generation                           | User  | RSA private key<br>DSA private key<br>ECDSA private/public  | Read / Execute |
| Signature Verification                         | User  | RSA public key<br>DSA public key<br>ECDSA private/public  | Read / Execute |
| Symmetric decryption                           | User  | AES key<br>Triple-DES key   | Read / Execute |
| Symmetric encryption                           | User  | AES key<br>Triple-DES key   | Read / Execute |

**Table 6 – Module Services, Roles, and Descriptions**

When in Non-FIPS approved mode of operation, the module allows access to each of the services listed above, with exception of FIPS self-tests. When in non-FIPS-approved mode of operation the module **also** provides a service (API function call) for each non-approved algorithm listed in Section 2.1.4. These function calls are assigned to the User, and have Read/Write/Execute permission to the module's memory while in operation.

---

<sup>6</sup> SHA – Secure Hash Standard

### 2.3.2 Operator Authentication

As required by FIPS 140-2, there are two roles (a Crypto Officer role and User role) in the module that operators may assume. As allowed by Level 1, the module does not support authentication to access services. As such, there are no applicable authentication policies. Access control policies are implicitly defined by the services available to the roles as specified in Table 6 – Module Services, Roles, and Descriptions.

## 2.4 Physical Security

This section of requirements does not apply to this module. The module is a software-only module and does not implement any physical security mechanisms.

## 2.5 Operational Environment

The module operates on a general purpose computer (GPC) running a general purpose operating system (GPOS). For FIPS purposes, the module is running on this operating system in single user mode and does not require any additional configuration to meet the FIPS requirements.

The module was tested on the following platforms:

- OEM PowerEdge R420 running 64-bit Windows Server 2012 with Java Runtime Environment (JRE) v1.7.0\_17.

The module is also supported on the following platform for which operational testing was not performed:

- OEM PowerEdge R420 running 64-bit Windows Server 2012 with Java Runtime Environment (JRE) v1.8.0\_45
- OEM PowerEdge R420 running CentOS 6.7 and CentOS 7
- Nexus 5 running Android 4 and 5

Compliance is maintained for other environment where the module is unchanged. 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 GPC(s) used during testing met Federal Communications Commission (FCC) FCC Electromagnetic Interference (EMI) and Electromagnetic Compatibility (EMC) requirements for business use as defined by 47 Code of Federal Regulations, Part15, Subpart B. FIPS 140-2 validation compliance is maintained when the module is operated on other versions of the GPOS running in single user mode, assuming that the requirements outlined in NIST IG G.5 are met.

## 2.6 Cryptographic Key Management

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

| Keys and CSPs  | Storage Locations | Storage Method | Input Method                      | Output Method                      | Zeroization |
|--|-------------------|----------------|-----------------------------------|------------------------------------|-------------|
| AES key<br>AES128, 192, 256 bit key for encryption, decryption                               | RAM               | Plaintext      | Input electronically in plaintext | Never                              | power cycle |
| Triple-DES key<br>Triple-DES 112, 168 bit key for encryption, decryption                     | RAM               | Plaintext      | Input electronically in plaintext | Never                              | power cycle |
| HMAC key<br>HMAC key for message Authentication with SHS                                     | RAM               | Plaintext      | Input electronically in plaintext | Never                              | power cycle |
| RSA private key<br>RSA 2048, 3072 bit key for signature and key generation                   | RAM               | Plaintext      | Input electronically in plaintext | Never                              | power cycle |
|  | RAM               | Plaintext      | Internally generated              | Output electronically in plaintext | power cycle |
| RSA public key<br>RSA 1024, 2048, 3072 bit key for signature verification and key generation | RAM               | Plaintext      | Input electronically in plaintext | Never                              | power cycle |
|  | RAM               | Plaintext      | Internally generated              | Output electronically in plaintext | power cycle |
| DSA private key<br>DSA 2048, 3072-bit for signature generation                               | RAM               | Plaintext      | Input electronically in plaintext | Never                              | power cycle |
|  | RAM               | Plaintext      | Internally generated              | Output electronically in plaintext | power cycle |
| DSA public key<br>DSA 1024, 2048, 3072-bit key for signature verification                    | Module Binary     | Plaintext      | Input electronically in plaintext | Never                              | power cycle |
|  | RAM               | Plaintext      | Internally generated              | Output electronically in plaintext | power cycle |
| ECDSA private key<br>All NIST defined B, K,  | RAM               | Plaintext      | Input electronically in plaintext | Never exits the module             | power cycle |

| Keys and CSPs  | Storage Locations | Storage Method | Input Method  | Output Method                      | Zeroization |
|--|-------------------|----------------|---|------------------------------------|-------------|
| and P Curves for signature generation  | RAM               | Plaintext      | Internally generated  | Output electronically in plaintext | power cycle |
| ECDSA public key<br>All NIST defined B, K, and P Curves for signature verification                     | RAM               | Plaintext      | Input electronically in plaintext                                     | Never exits the module             | power cycle |
|  | RAM               | Plaintext      | Internally generated  | Output electronically in plaintext | power cycle |
| DH public components<br>Public components of DH protocol   | RAM               | Plaintext      | Internally generated  | Output electronically in plaintext | power cycle |
| DH private component<br>Private exponent of DH protocol  | RAM               | Plaintext      | Internally generated  | Never                              | power cycle |
| EC DH public components<br>Public components of EC DH protocol   | RAM               | Plaintext      | Internally generated  | Output electronically in plaintext | power cycle |
| EC DH private component<br>Private exponent of EC DH protocol  | RAM               | Plaintext      | Internally generated  | Never exits the module             | power cycle |
| DRBG seed<br>Random data 440-bit or 880-bit to generate random number using the DRBG                   | RAM               | Plaintext      | Internally generated using nonce along with DRBG entropy input string | Never exits the module             | power cycle |
| DRBG Entropy Input String<br>512-bit value to generate seed and determine random number using the DRBG | RAM               | Plaintext      | Externally generated; Input electronically in plaintext               | Never exits the module             | power cycle |

R = Read W = Write D = Delete

**Table 7 – Module Keys/CSPs**

The application that uses the module is responsible for appropriate destruction and zeroization of the key material. The module provides functions for key allocation and destruction which overwrite the memory that is occupied by the key information with zeros before it is deallocated.



### **2.6.1 Random Number Generation**

The module uses SP800-90A DRBG for creation of asymmetric and symmetric keys.

The module accepts input from entropy sources external to the cryptographic boundary for use as seed material for the module's Approved DRBG. Therefore, the module generates cryptographic keys whose strengths are modified by available entropy, and no assurance is provided for the strength of the generated keys.

The module performs continual tests on the output of the approved RNG to ensure that consecutive random numbers do not repeat.

### **2.6.2 Key/CSP Storage**

Public and private keys are provided to the module by the calling process and are destroyed when released by the appropriate API function calls or during power cycle. The module does not perform persistent storage of keys.

### **2.6.3 Key/CSP Zeroization**

The application is responsible for calling the appropriate destruction functions from the API. The destruction functions then overwrite the memory occupied by keys with zeros and deallocates the memory. This occurs during process termination / power cycle. Keys are immediately zeroized upon deallocation, which sufficiently protects the CSPs from compromise.

## **2.7 Self-Tests**

FIPS 140-2 requires that the module perform self tests to ensure the integrity of the module and the correctness of the cryptographic functionality at start up. In addition some functions require continuous verification of function, such as the random number generator. All of these tests are listed and described in this section.

If any self-test fails, the module will enter a critical error state, during which cryptographic functionality and all data output is inhibited. To clear the error state, the CO must reboot the host system, reload the module, or restart the calling application. No operator intervention is required to run the self-tests.

The following sections discuss the module's self-tests in more detail.

### **2.7.1 Power-On Self-Tests**

Power-on self-tests are executed automatically when the module is loaded into memory. The module verifies the integrity of the runtime executable using a HMAC-SHA-512 digest computed at build time. If the fingerprints match, the power-up self-tests are then performed. If the power-up self-tests are successful, the module is in FIPS mode.

| TYPE                        | DETAIL   |
|-----------------------------|--|
| Software Integrity Check    | <ul style="list-style-type: none"> <li>• HMAC-SHA512 on all module components</li> </ul>   |
| Known Answer Tests          | <ul style="list-style-type: none"> <li>• AES encrypt and decrypt KATs</li> <li>• Triple-DES encrypt and decrypt KATs</li> <li>• HMAC SHA1 KAT</li> <li>• HMAC SHA-256 KAT</li> <li>• HMAC SHA-512 KAT</li> <li>• RSA sign and verify KATs</li> <li>• SP 800-90A DRBG KAT (HMAC)</li> </ul> |
| Pair-wise Consistency Tests | <ul style="list-style-type: none"> <li>• DSA</li> <li>• ECDSA</li> <li>• Diffie-Hellman</li> <li>• EC Diffie-Hellman</li> </ul>  |

**Table 8 – Power-On Self-Tests**

Input, output, and cryptographic functions cannot be performed while the Module is in a self-test or error state because the module is single-threaded and will not return to the calling application until the power-up self tests are complete. If the power-up self tests fail, subsequent calls to the module will also fail - thus no further cryptographic operations are possible.

### 2.7.2 Conditional Self-Tests

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

| TYPE                        | DETAIL  |
|-----------------------------|---|
| Pair-wise Consistency Tests | <ul style="list-style-type: none"> <li>• DSA</li> <li>• ECDSA</li> <li>• Diffie-Hellman</li> <li>• EC Diffie-Hellman</li> </ul> |
| Continuous RNG Tests        | <ul style="list-style-type: none"> <li>• SP 800-90A DRBG (HMAC)</li> </ul>  |

**Table 9 – Conditional Self-Tests**

## 2.8 Mitigation of Other Attacks

The Module does not contain additional security mechanisms beyond the requirements for FIPS 140-2 Level 1 cryptographic modules.

## 3 Guidance and Secure Operation

### 3.1 Crypto Officer Guidance

#### 3.1.1 Software Installation

The module is provided directly to solution developers and is not available for direct download to the general public. The module and its host application is to be installed on an operating system specified in Section 2.5 or one where portability is maintained.

In order to remain in FIPS-approved mode, the following steps must be taken during the installation process:

1. The Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 7 must be installed in the JRE. Instructions for installation are found in the download file located here: <http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html>
2. The module must be configured as the JRE's default Security Provider by modifying the `jre/lib/security/java.security` file and adding the following line to the list of providers:

```
security.provider.1= com.safelogic.cryptocomply.jce.provider.Provider
```

#### 3.1.2 Additional Rules of Operation

1. The writable memory areas of the module (data and stack segments) are accessible only by the application so that the operating system is in "single user" mode, i.e. only the application has access to that instance of the module.
2. The operating system is responsible for multitasking operations so that other processes cannot access the address space of the process containing the module.

### 3.2 User Guidance

#### 3.2.1 General Guidance

The module is not distributed as a standalone library and is only used in conjunction with the solution.

The end user of the operating system is also responsible for zeroizing CSPs via wipe/secure delete procedures.

If the module power is lost and restored, the calling application can reset the IV to the last value used.

### 3.2.2 FIPS-Approved Mode of Operation

In order to maintain the FIPS-approved mode of operation, the following requirements must be observed:

1. The calling application must instantiate and operate the module through the JCE interface provided by the JDK.
2. The calling application may not share CSPs between non-FIPS-approved-mode and FIPS-approved-mode of operation. The operator must reset the module before switching to FIPS-approved-mode of operation.
3. The calling application must restrict the use of two-key Triple DES encryption: the total number of blocks of data encrypted with the same cryptographic key shall not be greater than  $2^{20}$ .
4. The module requires that a minimum of 256 bits of entropy be provided for each use of the DRBG / load entropy service.