



## FIPS 140-2 Non-Proprietary Security Policy

---

Symantec Java Cryptographic Module

Software Version 1.3

Document Version 1.0

December 13, 2017

*Prepared For:*



Symantec Corporation

303 2<sup>nd</sup> Street 1000N

San Francisco, CA 94107

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

*Prepared By:*



Apex Assurance Group, LLC

530 Lytton Avenue, Ste. 200

Palo Alto, CA 94301

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

*Latest Update By:*

Symantec DLP Security Engineering

303 2<sup>nd</sup> Street 1000N

San Francisco, CA 94107

## **Abstract**

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

## Table of Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	About FIPS 140	5
1.2	About this Document	5
1.3	External Resources	5
1.4	Notices	5
1.5	Acronyms	5
<b>2</b>	<b>Symantec Java Cryptographic Module</b>	<b>8</b>
2.1	Cryptographic Module Specification	8
2.1.1	Validation Level Detail	8
2.1.2	Approved Cryptographic Algorithms	8
2.1.3	Non-Approved Cryptographic Algorithms	9
2.2	Module Interfaces	10
2.3	Roles, Services, and Authentication	11
2.3.1	Operator Services and Descriptions	11
2.3.2	Operator Authentication	13
2.4	Physical Security	13
2.5	Operational Environment	13
2.6	Cryptographic Key Management	15
2.6.1	Key Generation	16
2.6.2	Key Entry, Output, and Protection	16
2.6.3	Key/CSP Storage and Zeroization	16
2.7	Self-Tests	17
2.7.1	Power-On Self-Tests	17
2.7.2	Conditional Self-Tests	17
2.7.3	Critical Functions Tests	18
2.8	Mitigation of Other Attacks	18
<b>3</b>	<b>Guidance and Secure Operation</b>	<b>19</b>
3.1	Initial Setup	19
3.2	Crypto Officer Guidance	19
3.2.1	Software Packaging and OS Requirements	19
3.2.2	Enabling FIPS Mode	20
3.2.3	Management Procedures	20
3.2.4	Additional Rules of Operation	20
3.3	User Guidance	20
3.3.1	General Guidance	20
3.4	Role Changes	21

## List of Tables

Table 1 – Acronyms and Terms.....	7
Table 2 – Validation Level by DTR Section .....	8
Table 3 – FIPS-Approved Algorithm Certificates.....	9
Table 4 – Logical Interface / Physical Interface Mapping .....	11
Table 5 – Module Services and Descriptions .....	13
Table 6 – Module Keys/CSPs.....	16
Table 7 – Power-On Self-Tests .....	17
Table 8 – Conditional Self-Tests.....	18
Table 9 – Critical Functions Tests.....	18

## List of Figures

Figure 1 – Module Boundary and Interfaces Diagram.....	10
--	----

## 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 current version of the standard is FIPS 140-2. The National Institute of Standards and Technology (NIST) and Communications Security Establishment (CSE) Cryptographic Module Validation Program (CMVP) run the FIPS 140 program. The National Voluntary Laboratory Accreditation Program (NVLAP) accredits independent testing labs to perform FIPS 140 testing; the CMVP also validates test reports for products 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 <http://csrc.nist.gov/groups/STM/cmvp/index.html>.

### 1.2 About this Document

This non-proprietary Cryptographic Module Security Policy for the Java Cryptographic Module from Symantec 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.

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

### 1.3 External Resources

The Symantec website (<http://www.symantec.com>) contains information on Symantec products. The Cryptographic Module Validation Program website contains links to the FIPS 140-2 certificate and Symantec 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
CBC	Cipher Block Chaining
CCM	Counter with CBC-MAC
CFB	Cipher Feedback
CAVP	Cryptographic Algorithm Validation Program
CMVP	Cryptographic Module Validation Program
CO	Crypto Officer
CSE	Communications Security Establishment
CSP	Critical Security Parameter
CTR	Counter
DES	Data Encryption Standard
DESX	Data Encryption Standard XOR'ed
DH	Diffie-Hellman
DMZ	Demilitarized Zone
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
DTR	Derived Test Requirement
EC	Elliptic Curve
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
EC Diffie-Hellman	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption System
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
GPC	General Purpose Computer
HMAC	(Keyed-) Hash Message Authentication Code
HTTP	Hypertext Transfer Protocol
JAR	Java Archive
JRE	Java Runtime Environment
JVM	Java Virtual Machine
KAT	Known Answer Test
MAC	Message Authentication Code
MD	Message Digest

NIST	National Institute of Standards and Technology
NVLAP	National Voluntary Laboratory Accreditation Program
OEAP	Optimal Asymmetric Encryption Padding
OFB	Output Feedback
OS	Operating System
PKCS	Public-Key Cryptography Standards
PRNG	Pseudo Random Number Generator
PSS	Probabilistic Signature Scheme
RC	Rivest Cipher
RACE	Research and Development in Advanced Communications Technologies in Europe
RIPEMD	RACE Integrity Primitives Evaluation Message Digest
RNG	Random Number Generator
RSA	Rivest, Shamir, and Adleman
SEP	Symantec Endpoint Protection
SHA	Secure Hash Algorithm
SP	Special Publication
SSL	Secure Sockets Layer
TDEA	Triple Data Encryption Algorithm
Triple-DES	Triple Data Encryption Algorithm
TLS	Transport Layer Security
USB	Universal Serial Bus

Table 1 – Acronyms and Terms

## 2 Symantec Java Cryptographic Module

### 2.1 Cryptographic Module Specification

The module, Symantec Java Cryptographic Module, is a software shared library that provides cryptographic services required by Symantec’s line of software products. The module is a software-only module installed on a General Purpose Computer running Microsoft Windows 7 (64-bit). The module is comprised of two components:

1. The Symantec cryptographic module wrapper fully initializes and manages FIPS 140-2 mode. This includes performing an integrity check, verifying the provider is configured, performing the provider self tests, and reporting status.
2. An embedded validated module (see Certificate #2057) provides cryptographic functions.

All operations of the module occur via calls from the Symantec applications and their respective internal daemons/processes. There are no untrusted processes calling the services of the module, as APIs’ are not exposed.

#### 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	1

Table 2 – Validation Level by DTR 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 (ECB, CBC, CFB (128), OFB (128), CTR - [128, 192, 256 bit key sizes] CCM, GCM)	2249
DSA	701
ECDSA	357
HMAC DRBG (SP800-90A)	273
HMAC-SHA-1 <sup>1</sup> , HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	1378
RSA X9.31, PKCS#1 v1.5, PKCS#1 v2.1 (SHA256 – PSS)	1154
SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	1938
PBKDF (vendor affirmed)	Vendor Affirmed
Triple-DES (ECB, CBC, CFB-, OFB)	1408

Table 3 – FIPS-Approved Algorithm Certificates<sup>1</sup>

### 2.1.3 Non-Approved Cryptographic Algorithms

The module does not implement any non-approved algorithms in FIPS mode; however, Diffie-Hellman, EC Diffie-Hellman, and RSA Key Wrap (RSA key-pair with a modulus length of 2048 to 4096 bits) are allowed in the FIPS mode of operation. The module utilizes the following non-FIPS-approved algorithm implementations:

- DES
- DESX
- Diffie-Hellman (primitives only, public key size less than 2048 bits, and private key size less than 160 bits)
- EC Diffie-Hellman (primitives only, key size less than 224 bits)
- ECIES
- MD2
- MD4
- MD5
- ECDSA, DSA and RSA signature generation with SHA-1
- FIPS 186-2 PRNG
- Dual\_EC\_DRBG
- RC2 block cipher
- RC4 stream cipher
- RC5 block cipher
- RSA Keypair Generation MultiPrime (two or three primes)
- RIPEMD160
- HMAC-MD5
- 1024-bit DSA PQG, key, and signature generation
- 1024 RSA key and signature generation
- ECDSA Public Key and Signature Generation using curves P-192, B-163, and K-163

<sup>1</sup> Note this implementation has received FIPS 140-2 Level 1 validation 2057:  
<http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm#2057>

Any protocol or associated cryptographic functions have not undergone any testing by the CAVP and are disallowed in an Approved mode.

## 2.2 Module Interfaces

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

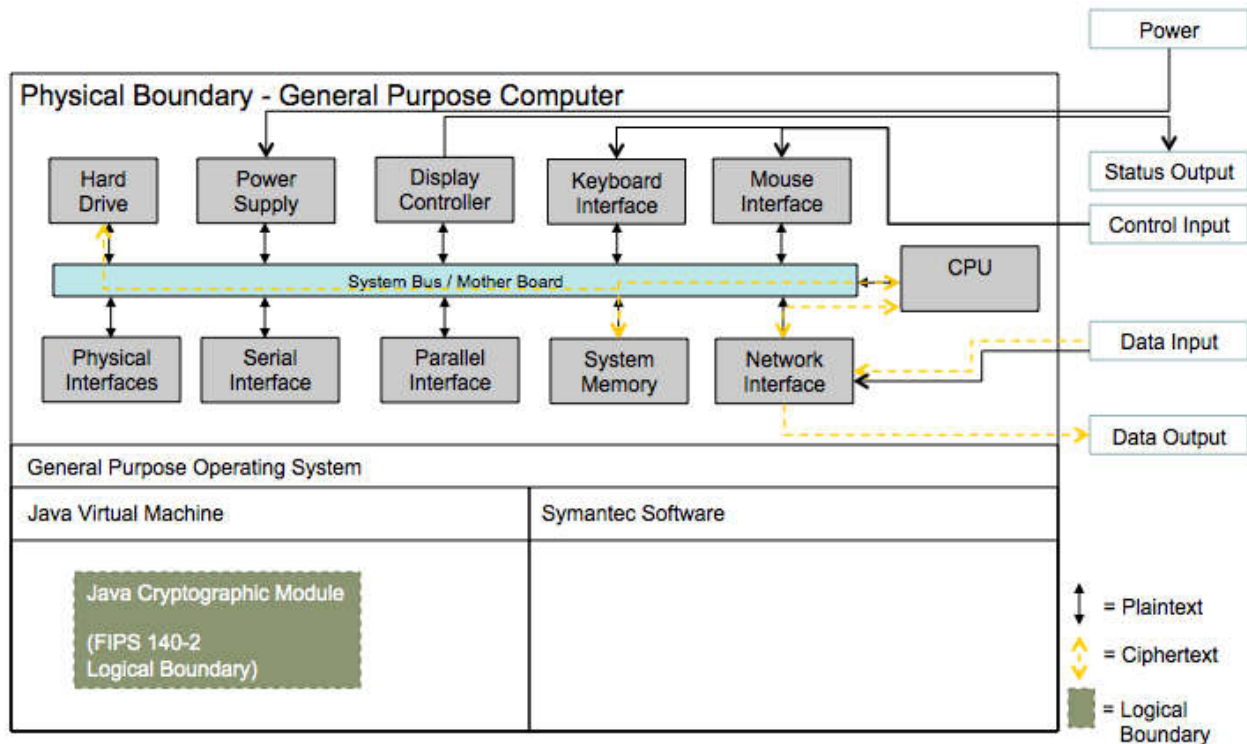


Figure 1 – Module Boundary and Interfaces Diagram

The interfaces (ports) for the physical boundary include the computer keyboard port, CDROM drive, floppy disk, mouse, network port, parallel port, USB ports, monitor port 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	USB ports, network ports, serial ports, SCSI/SATA ports, DVD, audio Ports
Data Output	Output parameters of API function calls	Display (e.g. VGA, HDMI, DVI, etc.), USB ports, network ports, serial ports, SCSI/SATA ports, audio ports, DVD
Control Input	API function calls	USB ports, network ports, serial ports, power switch
Status Output	For FIPS 140-2 mode, function calls returning status information and return codes provided by API function calls.	Display, serial ports, network ports
Power	None	Power supply/connector

**Table 4 – Logical Interface / Physical Interface Mapping**

As shown in Figure 1 – Module Boundary and Interfaces Diagram and Table 5 – Module Services 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. The module does not output key/CSP information while in an error state.

## 2.3 Roles, Services, and Authentication

The module supports a Crypto Officer and a User role. The module does not support a Maintenance role.

### 2.3.1 Operator Services and Descriptions

The services available to the User and Crypto Officer roles in the module are as follows:

Service	Roles	Input	Output	Key/CSP Access
On Demand Self-test	Crypto Officer	None	Status	None
Get FIPS140 Context	User	None	Status	None
Get seeder	User	None	Seed generator	None
Get Default Random Number Generator	User	None	Random Number Generator	None
Check FIPS 140-2 Compliance	User	None	Status	None
Get State	User	None	Status	None
Get Mode	User	None	Status	None
Set Mode	User	API call parameter	Status	None
Get Role	User	None	Status	None
Set Role	User	API call parameter	Status	None
Check Latest Self-Test Results	User	None	Status	None
Check Mode	User	None	Status	None
Disable library	User	API call parameter	None	None
Verify DSA Parameters	User	API call parameter	Status	None
Encryption	User	API call parameters, key, plaintext	Status, ciphertext	AES Key Triple-DES Key
Decryption	User	API call parameters, key, ciphertext	Status, plaintext	AES Key Triple-DES Key
Digital Signature Generation	User	API call parameters, key, message	Status, signature	RSA Private Key RSA Public Key DSA Private Key DSA Public Key
Digital Signature Verification	User	API call parameters, key, signature, message	Status	RSA Private Key RSA Public Key DSA Private Key DSA Public Key
Key Establishment Primitives	User	API call parameters, key	Status, key	RSA Private Key RSA Public Key DH Public Key DH Private Key EC Diffie-Hellman Public Key EC Diffie-Hellman Private Key

Service	Roles	Input	Output	Key/CSP Access
Key Generation	User	API call parameters	Status, key/key pair	AES Key Triple-DES Key DSA Private Key DSA Public Key RSA Private Key RSA Public Key DH Public Key DH Private Key HMAC DRBG Key HMAC with SHA-1 and SHA-2 Keys
MAC	User	API call parameters key, message	Status, hash	HMAC DRBG Key HMAC with SHA-1 and SHA-2 Keys
Hashing	User	API call parameters, message	Status, hash	None
Random Number Generation	User	API call parameters	Status, random bits	HMAC DRBG Entropy HMAC DRBG V Value (Seed Length) HMAC DRBG Key HMAC DRBG init_seed
Zeroization	User	API call parameters	Status	All

**Table 5 – Module Services and Descriptions**

### 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.

## 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 on a modern version of Microsoft Windows and CentOS 6, as a general-purpose operating system (GPOS). For FIPS purposes, the module is running on Microsoft Windows in single user mode and does not require any additional configuration to meet the FIPS requirements.

The module was tested on the following platforms:

- Microsoft Windows 7 (64-Bit) with Sun JRE 8.0 on VmWare ESXi 6.0.0 running on a Dell PowerEdge R430.

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, Part 15, Subpart B. FIPS 140-2 validation compliance is maintained when the module is operated on other versions of the Microsoft Windows GPOS running in single user mode, assuming that the requirements outlined in NIST IG G.5 are met.

Symantec is affirming its compliance as outlined in section G.5 of the Implementation Guidance for FIPS 140-2. Compliance is maintained on platforms for which the binary executable remains unchanged. This includes (but is not limited to):

- Windows XP Professional SP3, x86 (32-bit) with Sun JRE 6.0/7.0/8.0,
- Windows XP Professional SP3, x86\_64 (64-bit) with Sun JRE 6.0/7.0/8.0
- Windows 7, x86 (32-bit) with Sun JRE 6.0/7.0/8.0
- Windows 7, x86\_64 (64-bit) with Sun JRE 6.0/7.0/8.0
- Windows Server 2003 x86 (32-bit) with Sun JRE 6.0/7.0/8.0
- Windows Server 2003 x86\_64 (64-bit) with Sun JRE 6.0/7.0/8.0
- Windows Server 2008 x86 (32-bit) with Sun JRE 6.0/7.0/8.0
- Windows Server 2008 x86\_64 (64-bit), Sun JRE 6.0/7.0/8.0
- Windows Server 2008 R2 x86\_64 (64-bit), Sun JRE 6.0/7.0/8.0
- Windows Server 2012 x86\_64 (64-bit), Sun JRE 6.0/7.0/8.0
- Windows Server 2012 x86\_64 (64-bit), Sun JRE 6.0/7.0/8.0
- CentOS 6.7 x86\_64, Sun JRE 6.0/7.0/8.0
- CentOS 6.8 x86\_64, Sun JRE 6.0/7.0/8.0
- CentOS 7.0 x86\_64, Sun JRE 6.0/7.0/8.0

## 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	Access
AES Key	RAM	Plaintext	Internally generated	API call parameter	SensitiveData.clear() power cycle	CO: RWD  U: RWD
Triple-DES Key	RAM	Plaintext	Internally generated	API call parameter	SensitiveData.clear() power cycle	CO: RWD  U: RWD
HMAC with SHA-1 and SHA-2 Keys	RAM	Plaintext	API call parameter	None	SensitiveData.clear() power cycle	CO: RWD  U: RWD
EC Private Keys	RAM	Plaintext	Internally generated	API call parameter	SensitiveData.clear() power cycle	CO: RWD  U: RWD
EC Public Keys	RAM	Plaintext	Internally generated	API call parameter	SensitiveData.clear() power cycle	CO: RWD  U: RWD
DH Private Key	RAM	Plaintext	Internally generated	API call parameter	SensitiveData.clear() power cycle	CO: RWD  U: RWD
DH Public Key	RAM	Plaintext	Internally generated	API call parameter	SensitiveData.clear() power cycle	CO: RWD  U: RWD
RSA Private Key	RAM	Plaintext	Internally generated	API call parameter	SensitiveData.clear() power cycle	CO: RWD  U: RWD
RSA Public Key	RAM	Plaintext	Internally generated	API call parameter	SensitiveData.clear() power cycle	CO: RWD  U: RWD
DSA Private Key	RAM	Plaintext	Internally generated	API call parameter	SensitiveData.clear() power cycle	CO: RWD  U: RWD
DSA Public Key	RAM	Plaintext	Internally generated	API call parameter	SensitiveData.clear() power cycle	CO: RWD  U: RWD
HMAC DRBG Entropy	RAM	Plaintext	Internally generated	None	SensitiveData.clear() power cycle	CO: RWD  U: RWD

Keys and CSPs	Storage Locations	Storage Method	Input Method	Output Method	Zeroization	Access
HMAC DRBG V Value (Seed Length)	RAM	Plaintext	Internally generated	None	<code>SensitiveData.clear()</code> power cycle	CO: RWD U: RWD
HMAC DRBG Key	RAM	Plaintext	Internally generated	None	<code>SensitiveData.clear()</code> power cycle	CO: RWD U: RWD
HMAC DRBG init_seed	RAM	Plaintext	Internally generated	None	<code>SensitiveData.clear()</code> power cycle	CO: RWD U: RWD
Integrity test key	Hard-coded, RAM	Plaintext	Hard-coded	None	<code>SensitiveData.clear()</code> power cycle	CO: RWD U: RWD
Known Answer Test (KAT) keys	Hard-coded, RAM	Plaintext	Hard-coded	None	<code>SensitiveData.clear()</code> power cycle	CO: RWD U: RWD

R = Read W = Write D = Delete

**Table 6 – Module Keys/CSPs**

### 2.6.1 Key Generation

The module supports the generation of the DSA, RSA, and Diffie-Hellman (DH) and ECC public and Private Keys. In the FIPS-approved mode, RSA keys can only be generated using the approved 186-4 RSA key generation method.

The module employs a FIPS-approved SP 800-90A HMAC DRBG for generating asymmetric and symmetric keys. Entropy for use in key generation is gathered by the embedded validated module; various system parameters are collected to suitably account for the strength of the generated key.

### 2.6.2 Key Entry, Output, and Protection

All keys and CSPs reside on memory internally allocated by the module and can only be output using the exposed APIs. The module does not support key entry or output from the physical boundary. The operating system and the JRE protect the memory and process space from unauthorized access.

### 2.6.3 Key/CSP Storage and Zeroization

The module does not provide long-term cryptographic key storage. Storage of keys is the responsibility of the user of the module. All keys and CSPs are automatically zeroized by the module at the end of their lifetime. The user can ensure destruction of sensitive data by calling `SensitiveData.clear()`. Power cycling the module will also zeroize keys.



## 2.7 Self-Tests

The module performs power-up and conditional self-tests to ensure proper operation. If a power-up self-test fails, the module is disabled and throws a `SecurityException`. The module can only leave the disabled state by restarting the Java Virtual Machine. If a conditional self-test fails, the module throws a `SecurityException` and aborts the operation. A conditional self-test failure does not disable the module.

In event of a self-test failure, the module provides the following message: `Could not initialize class com.rsa.jsafe.provider.JsafeJCE.`

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

### 2.7.1 Power-On Self-Tests

The module implements the following power-on self-tests:

TYPE	DETAIL
Software Integrity Check	RSA Digital Signature Verification
Known Answer Tests	<ul style="list-style-type: none"> <li>• AES (separate encrypt and decrypt) (CBC, CFB, ECB, OFB)</li> <li>• DSA (separate sign and verify)</li> <li>• ECDSA (separate sign and verify)</li> <li>• SP 800-90A HMAC DRBG health tests</li> <li>• HMAC SHA-1</li> <li>• HMAC SHA-224</li> <li>• HMAC SHA-256</li> <li>• HMAC SHA-384</li> <li>• HMAC SHA-512</li> <li>• RSA (separate sign and verify)</li> <li>• SHA-1</li> <li>• SHA-224</li> <li>• SHA-256</li> <li>• SHA-384</li> <li>• SHA-512</li> <li>• Triple-DES (separate encrypt and decrypt)</li> </ul>
Pair-wise Consistency Tests	<ul style="list-style-type: none"> <li>• DSA</li> <li>• ECDSA</li> <li>• RSA</li> </ul>

Table 7 – Power-On Self-Tests

Power-on self-tests are executed automatically when the module is loaded into memory.

### 2.7.2 Conditional Self-Tests

The module implements the following conditional self-tests:

TYPE	DETAIL
Pair-wise Consistency Tests	<ul style="list-style-type: none"> <li>• DSA</li> <li>• ECDSA</li> <li>• RSA</li> </ul>
Continuous RNG Tests	<ul style="list-style-type: none"> <li>• HMAC_DRBG;</li> <li>• DUAL_EC_DRBG (non-approved); and</li> <li>• FIPS 186-2 RNG (non-approved)</li> </ul>

Table 8 – Conditional Self-Tests

### 2.7.3 Critical Functions Tests

The module implements the following critical functions tests:

TYPE	DETAIL
Known Answer Tests	<ul style="list-style-type: none"> <li>• ECIES when operating in FIPS140_ECC_MODE</li> </ul>

Table 9 – Critical Functions Tests

## 2.8 Mitigation of Other Attacks

As a defense against timing attacks, RSA key operations implement blinding by default. By using the blinding method, it is ensured that the decryption time is not correlated to the input ciphertext; as a consequence, attempts of timing attacks are thwarted. Blinding is implemented through blinding modes with the following available options:

- Blinding mode off
- Blinding mode with no update (the blinding value is squared for each operation)
- Blinding mode with full update (a new blinding value is used for each operation).

## 3 Guidance and Secure Operation

This section describes how to configure the module for FIPS-approved mode of operation. Operating the module without maintaining the following settings will remove the module from the FIPS-approved mode of operation.

### 3.1 Initial Setup

The Symantec cryptographic module wrapper fully initializes and manages FIPS mode. This includes performing an integrity check, verifying the provider is configured, performing the provider self tests, and reporting status.

When the module is loaded by the host application, the `FIPSMoDeManager.startFIPSMoDe()` function is called to initialize the module in a FIPS-approved mode of operation. This function checks the integrity of the module, runs all power-up self-tests, and, if successful, sets the module in the `FIPS140_SSL_MODE` by default. The initialization function records the following message to a log file:

```
System running in FIPS 140 mode.
```

The module uses JAR-signing to check the integrity of the module (the consuming application provides the signing certificate for the JARs of the module). Upon failure of either the software integrity test or any of the self-tests, the function throws an exception as status output and disables the library. Additionally, the module logs the following message:

```
FIPS initialization failed, FIPS cryptographic services disabled
```

### 3.2 Crypto Officer Guidance

#### 3.2.1 Software Packaging and OS Requirements

The module must be installed on a General Purpose Operating System running in single user mode. To configure single-user mode, the following must be disabled:

- Remote registry and remote desktop services
- Remote assistance
- Guest accounts
- Server and terminal services

Contact Microsoft support for configuration details; specific configuration steps are beyond the scope of this document.

### 3.2.2 Enabling FIPS Mode

No specific configuration is required to enforce FIPS mode beyond the `FIPSMoDeManager.startFIPSMoDe()` function. Status can be verified by calling the `FIPSMoDeManager.isInFIPS140MoDe()` function, which returns true if the module is in a FIPS-Approved mode and false if in a non-Approved mode.

### 3.2.3 Management Procedures

The Crypto Officer can run the self-tests at any time by calling the `runSelfTests()` function.

### 3.2.4 Additional Rules of Operation

1. All host system components that can contain sensitive cryptographic data (main memory, system bus, disk storage) must be located in a secure environment.
2. The writable memory areas of the Module (data and stack segments) are accessible only by the calling application so that the Module is in "single user" mode, i.e. only the calling application has access to that instance of the Module.
3. Imported keys should be generated via FIPS-approved manner.
4. The operating system is responsible for multitasking operations so that other processes cannot access the address space of the process containing the Module.
5. The operator must invoke the `SensitiveData.clear()` method before changing the module mode in order to ensure all keys and CSPs are zeroized.

## 3.3 User Guidance

### 3.3.1 General Guidance

In order to use the module in FIPS 140-2 mode of operation, the User must only use the approved algorithms as listed in Table 3 – FIPS-Approved Algorithm Certificates. The requirements for using the approved algorithms in a FIPS 140-2 mode of operation are as follows:

- The bit-length for a RSA and DSA key pair must be at least 2048 bits.
- SP 800-90A HMAC-DRBG random data requests must be less than or equal to  $2^{19}$  bits in length.
- Bit lengths for an HMAC key must be one half of the block size.
- RSA keys used for signing shall not be used for any other purpose other than digital signatures.

- For RSA digital signature generation the HMAC DRBG must be used
- EC key pairs must have domain parameters from the set of NIST-recommended named curves (P224, P256, P384, P521, B233, B283, B409, B571, K233, K283, K409, and K571). The domain parameters can be specified by name or can be explicitly defined.
- The module implements both Diffie-Hellman and EC Diffie-Hellman key agreement schemes.
- EC Diffie-Hellman primitives must use curve domain parameters from the set of NIST-recommended named curves listed above. The domain parameters can be specified by name, or can be explicitly defined. When using the NIST-recommended curves, the computed Diffie-Hellman shared secret provides between 112 bits and 256 bits of encryption strength.
- When using the Approved SP 800-90A DRBG to generate keys, the DRBGs requested security strength must be at least as great as the security strength of the key being generated.
- Only FIPS 140 approved random number generators may be used for generation of keys (asymmetric and symmetric)
- If the module power is lost and restored, the calling application can reset the AES GCM IV to the last value used.

Additionally, operators should take care to zeroize CSPs' when they are no longer needed.

### 3.4 Role Changes

If the operator needs to operate the module in different roles, then the operator must ensure that all instantiated cryptographic objects are destroyed before changing from the Crypto User role to the Crypto Officer role, or unexpected results could occur.