Security Builder® FIPS Java Module

Version 2.8

FIPS 140-2 Non-Proprietary
Security Policy

Certicom Corp.

October 7, 2011

certicom™

Contents

1 Introduction

1.1 Overview

This is a non-proprietary Federal Information Processing Standard (FIPS) 140-2 Security Policy for Certicom's **Security Builder® FIPS Java Module Version 2.8** (SB FIPS Java Module). SB FIPS Java Module is a cryptographic toolkit for Java language users, providing services of various cryptographic algorithms such as hash algorithms, encryption schemes, message authentication, and public key cryptography. This Security Policy specifies the rules under which SB FIPS Java Module must operate. These security rules are derived from the requirements of FIPS 140-2 [1], and related documents [6, 7, 8].

1.2 Purpose

This Security Policy is created for the following purposes:

1. It is required for FIPS 140-2 validation.
2. To outline SB FIPS Java Module's conformance to FIPS 140-2 Level 1 Security Requirements.
3. To provide users with how to configure and operate the cryptographic module in order to comply with FIPS 140-2.

1.3 References

References

[1] NIST *Security Requirements For Cryptographic Modules, FIPS PUB 140-2*, December 3, 2002.

[2] NIST *Security Requirements For Cryptographic Modules, Annex A: Approved Security Functions for FIPS PUB 140-2*, January 27, 2010.

[3] NIST *Security Requirements For Cryptographic Modules, Annex B: Approved Protection Profiles for FIPS PUB 140-2*, June 14, 2007.

[4] NIST *Security Requirements For Cryptographic Modules, Annex C: Approved Random Number Generators for FIPS PUB 140-2*, July 21, 2009.

[5] NIST *Security Requirements For Cryptographic Modules, Annex D: Approved Key Establishment Techniques for FIPS PUB 140-2*, October 8, 2009.

[6] NIST *Derived Test Requirements for FIPS 140-2*, Draft, March 24, 2004.

[7] NIST *Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program*, June 15, 2010.

[8] NIST *Frequently Asked Questions for the Cryptographic Module Validation Program*, December 4, 2007.

1.4 Change Notes

Change history is manually recorded in Table 1.

Table 1: Change History

Date	Author	Description
2010/08/03	A.Y.	Moved to Subversion Algorithm certificate numbers are added.
2010/08/13	A.Y.	Updated IP notices
2011/01/25	R.W.	Editorial corrections
2011/01/25	R.W.	Updated copyright year
2011/08/15	R.W.	Editorial modifications
2011/09/07	A.Y.	Respond to 2nd round comments

The following were placed here by RCS upon check-in.

```
Revision 1.50  2010/04/21 13:10:23  ayamada
Editorial corrections.

Revision 1.49  2010/04/20 15:45:06  ayamada
Updated with the latest information.

Revision 1.48  2010/01/08 17:53:52  ayamada
Added key agreement algorithms for SB FIPS Module Version 2.8.

Revision 1.47  2010/01/07 20:04:25  rwilliam
Merged 2.2.1 details from development branch.

Revision 1.46  2008/09/15 14:00:42  ayamada
Revised based on the comments from CMVP.

Revision 1.45  2008/06/16 18:00:39  ayamada
1. Algorithm Certificate numbers are provided.
2. Updated references.
3. Editorial correction.

Revision 1.44  2008/05/09 19:45:52  rwilliam
Clarified runSelfTests

Revision 1.43  2008/05/09 15:29:34  ayamada
Correction on the date.

Revision 1.42  2008/05/09 15:29:05  ayamada
Fixed the description to correctly represent the behavior in Java.

Revision 1.41  2008/04/18 17:04:55  ayamada
Corrections on the supported algorithms.
Editorial corrections.

Revision 1.40  2008/04/18 13:28:40  ayamada
For SB FIPS Jave Module 2.2.
1. Updated the list of tested platforms.
2. The tested JRE version is now 1.6.
3. Added new algorithms: AES CMAC, AES GCM, DRBG.
4. Clarification on the operational environment.

Revision 1.39  2008/04/15 12:35:28  ayamada
```

```
Fixed a typo.

Revision 1.38  2008/04/11 22:09:50  ayamada
For Version 2.2.
New platform list and JRE list.
Includes new algorithms such as NIST SP 800-90 RNG.

Revision 1.37  2008/03/14 17:50:42  ayamada
The first draft for SB FIPS Java Module 2.2.

Revision 1.36  2008/01/24 15:25:36  ayamada
Updated from the 2.1 branch.

Revision 1.35.6.1  2007/08/29 19:04:22  ayamada
Updated to add Solaris 10 32-bit.

Revision 1.35  2006/12/08 14:17:26  ayamada
Date update.

Revision 1.34  2006/12/08 14:15:55  ayamada
Correction on the DSS certificate number.

Revision 1.33  2006/12/07 19:46:09  ayamada
Updated the date on the document.

Revision 1.32  2006/12/07 19:44:24  ayamada
Added FIPS Algorithm certificate numbers.

Revision 1.31  2006/11/03 17:45:58  ayamada
Added extra space on page 2.

Revision 1.30  2006/11/03 17:39:02  ayamada
Added patent statement.

Revision 1.29  2006/11/03 15:07:49  ayamada
Addition of some clarifications and reference update.

Revision 1.28  2006/11/02 19:58:52  ayamada
Simplified operational environment description.

Revision 1.27  2006/10/13 14:22:49  ayamada
Improved to have clearer statements.

Revision 1.26  2006/09/14 14:04:22  ayamada
Revised to add 2 platforms.
Windows OS is updated to XP.

Revision 1.25  2006/06/28 14:09:02  ayamada
A bit more editorial improvements.

Revision 1.24  2006/06/28 14:01:32  ayamada
Some editorial fixes.

Revision 1.23  2006/06/27 20:15:40  mmezheri
updated supported hardware, software diagram and algorithms table

Revision 1.22  2006/06/26 15:20:06  mmezheri
updated regarding gse-j 2.1

Revision 1.21  2006/06/05 21:52:41  mmezheri
new algorithms updating

Revision 1.20  2005/12/14 17:02:48  zlieber
Merged sbgsej_2_0 branch: -j root-of-sbgsej_2_0-branch -j sbgsej_2_0_9

Revision 1.11.2.2  2005/09/27 18:10:34  ayamada
Further revised the notes on the security of key establishment techniques.
```

```
Revision 1.19  2005/09/27 13:42:29  ayamada
Added notes on security levels of Diffie-Hellman, Elliptic Curve Diffie-Hellman, and ECMQV.

Revision 1.18  2005/09/21 20:01:27  ayamada
Added clarification on key size for the RSA key wrapping techniques.

Revision 1.17  2005/09/14 18:47:47  ayamada
Further clarified the status of DES in Table 3.

Revision 1.16  2005/09/14 17:20:11  ayamada
1. Fix on Table 3 to clarify the legacy status of DES.
2. FIPS Approved is corrected to FIPS allowed for key establishment techniques.

Revision 1.15  2005/04/22 13:19:47  ayamada
Further clarifications.

Revision 1.14  2005/03/31 17:46:50  ayamada
Editorial corrections.

Revision 1.13  2005/03/28 15:26:50  ayamada
Correction on the date.

Revision 1.12  2005/03/28 15:21:17  ayamada
The algorithm certificate numbers are obtained.
Some minor editorial corrections.

Revision 1.11  2005/02/18 18:53:36  ayamada
A few minor corrections.

Revision 1.10  2005/02/18 16:04:20  ayamada
More clarifications and editorial corrections.

Revision 1.9  2005/02/15 17:10:11  efung
Typo

Revision 1.8  2005/02/11 21:16:22  efung
Superscript the (R)

Revision 1.7  2005/02/10 23:00:19  efung
supertab isn’t actually being used (package seems to be superseded by
supertabular)

Revision 1.6  2005/02/10 19:08:35  ayamada
Further corrections.

Revision 1.5  2005/02/10 18:34:20  ayamada
Editorial corrections.

Revision 1.4  2005/02/04 18:35:07  ayamada
Completed the appendix.

Revision 1.3  2005/02/01 15:56:28  ayamada
Editorial corrections.

Revision 1.2  2005/01/28 14:54:00  ayamada
Revised for differences of Java from C.

Revision 1.1  2005/01/28 13:48:21  ayamada
Initial revision: copied from GSE-C 2.0.
```

2 Cryptographic Module Specification

SB FIPS Java Module is a multiple-chip standalone cryptographic module that operates with the following components:

- A commercially available general-purpose computer hardware.
- A commercially available Operating System (OS) that runs on the computer hardware.
- A commercially available Java Virtual Machine (JVM) that runs on the computer hardware and OS.

2.1 Physical Specifications

The general-computer hardware component consists of the following devices:

1. CPU (Microprocessor)
2. Memory
   (a) Working memory is located on the RAM containing the following spaces:
      i. Input/output buffer
      ii. Plaintext/ciphertext buffer
      iii. Control buffer

      Key storage is not deployed in this module.

   (b) Program memory is also located on RAM.
3. Hard Disk (or disks)
4. Display Controller
5. Keyboard Interface
6. Mouse Interface
7. Network Interface
8. Serial Port
9. Parallel Port
10. Power Supply

The configuration of this component is illustrated in Figure 1.

2.2 Computer Hardware, OS, and JVM

SB FIPS Java Module is tested on the following representative combinations of computer hardware and OS, running the Java Runtime Environment (JRE) 1.5.0 and 1.6.0 by Sun Microsystems:

1. Solaris 10, 32-bit SPARC (Binary compatible to Solaris 9)
2. Solaris 10, 64-bit SPARC (Binary compatible to Solaris 9)
3. Red Hat Linux AS 5.5, 32-bit x86 (Binary compatible to AS 2.1/3.0/4.0/5.0)
4. Red Hat Linux AS 5.5, 64-bit x86 (Binary compatible to AS 4.0/5.0)
5. Windows Vista, 32-bit x86 (Binary compatible to Windows 98/2000/2003/XP)
6. Windows Vista, 64-bit x86 (Binary compatible to Windows 64-bit XP)

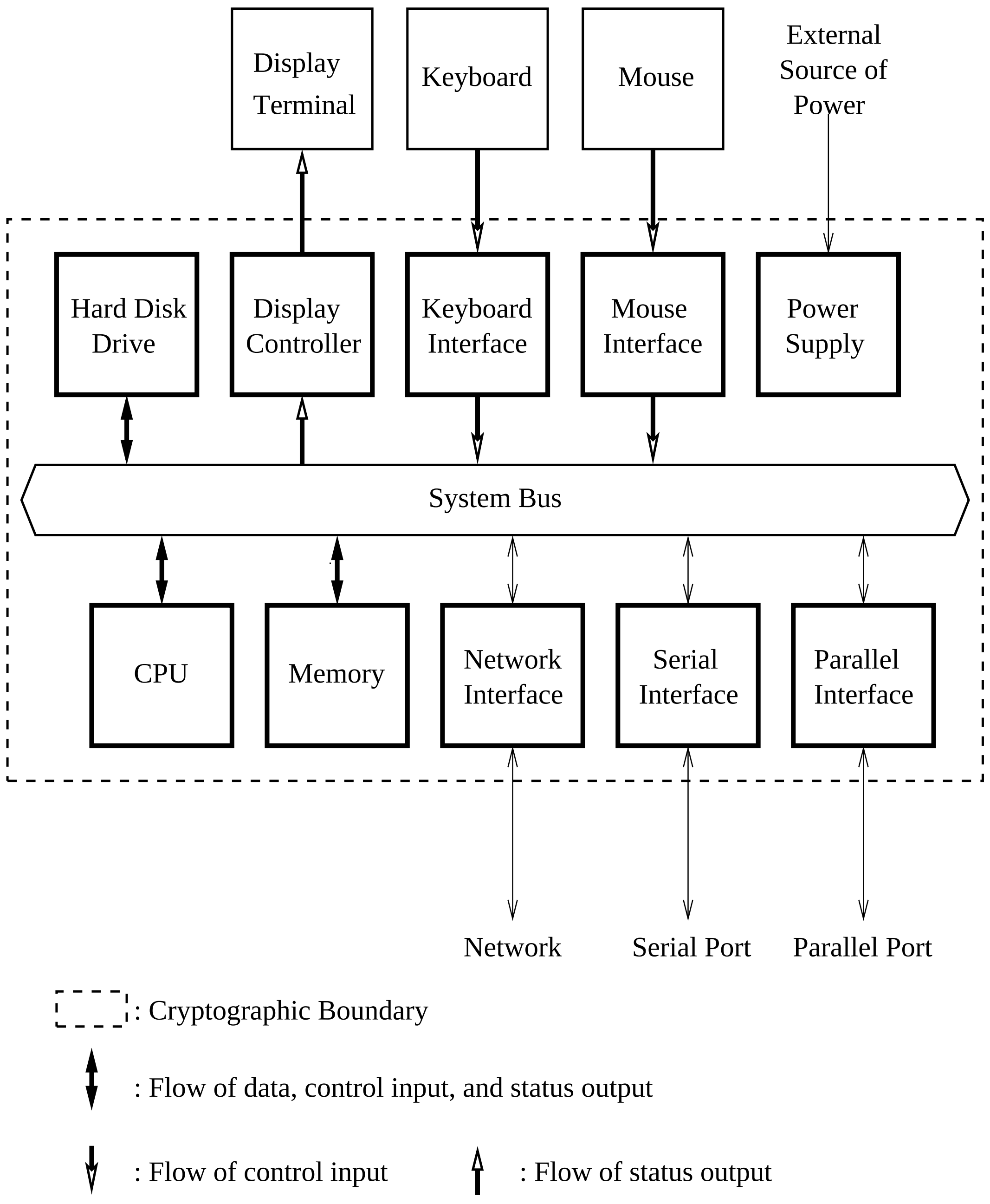

Figure 1: Cryptographic Module Hardware Block Diagram

7. Windows 2008 Server, 64-bit x86

The module will run on the JREs 1.3.1, and 1.4.2, and on various hardware and OS such as,

1. Any other Solaris Platforms,
2. Any other Linux Platforms,
3. Any other Windows Platforms,
4. AIX Platforms, and
5. HP-UX Platforms,

while maintaining its compliance to the FIPS 140-2 Level 1 requirements. Thus, this validation is applicable to these JREs and platforms as well.

2.3 Software Specifications

SB FIPS Java Module software is manufactured by Certicom Corp., providing services to the Java computer language users in the form of a Java archive (JAR). The same binary is used for all identified computer hardware and OS because the JVM underneath SB FIPS Java Module will absorb the differences of the computer hardware and OS.

The interface into SB FIPS Java Module is via Application Programmer's Interface (API) method calls. These method calls provide the interface to the cryptographic services, for which the parameters and return codes provide the control input and status output (see Figure 2).

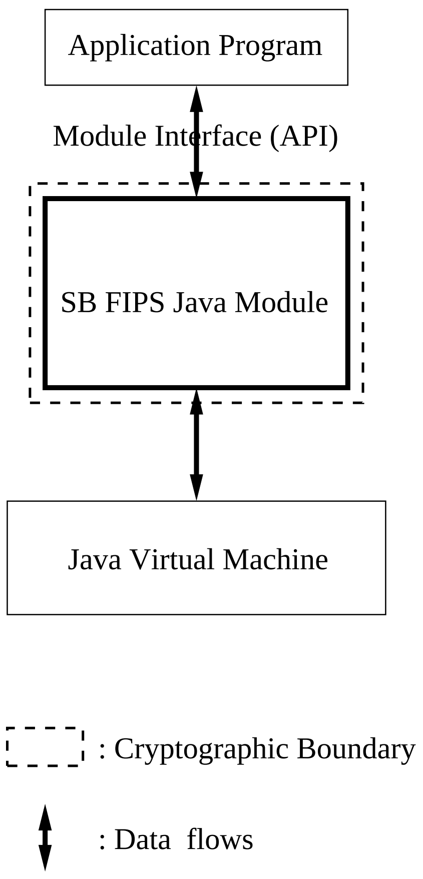

Figure 2: Cryptographic Module Software Block Diagram

3 Cryptographic Module Ports and Interfaces

The physical and logical interfaces are summarized in Table 2.

Table 2: Logical and Physical Interfaces

I/O	**Logical Interface**	**Physical Interface**
Data Input	API	Ethernet port
Data Output	API	Ethernet port
Control Input	API	Keyboard and Mouse
Status Output	Return Code	Display
Power Input	Initialization Function	The Power Supply is the power interface.
Maintenance	Not Supported	Not Supported

4 Roles, Services, and Authentication

4.1 Roles

SB FIPS Java Module supports Crypto Officer and User Roles (see Table 3). These roles are enforced by this Security Policy.

Table 3: Roles and Services

Service	**Crypto Officer**	**User**
Initialization, etc.		
Initialization	×	×
Deinitialization	×	×
Self-tests	×	×
Show status	×	×
Symmetric Ciphers (AES and Triple-DES)		
Key generation (Triple-DES only)	×	×
Encrypt	×	×
Decrypt	×	×
Key zeroization	×	×
Hash Algorithms and Message Authentication (SHS, HMAC)		
Hashing	×	×
Message Authentication	×	×
Random Number Generation (pRNG)		
Instantiation	×	×
Seeding	×	×
Request	×	×
CSP/key zeroization	×	×
Digital Signature (DSA, ECDSA, RSA)		
Key pair generation	×	×
Sign	×	×
Verify	×	×
Key zeroization	×	×
Key Agreement (Diffie-Hellman, Elliptic Curve Diffie-Hellman, ECMQV)		
Key pair generation	×	×
Shared secret generation	×	×
Key zeroization	×	×
Key Wrapping (RSA)		
Key pair generation	×	×
Wrap	×	×
Unwrap	×	×
Key zeroization	×	×

In order to operate the module securely, it is the Crypto Officer and User's responsibility to confine calls to those methods that have been FIPS 140-2 Approved. Thus, in the approved mode of operation, all Roles shall confine themselves to calling FIPS Approved algorithms, as marked in Table 4.

4.2 Services

SB FIPS Java Module supports many cryptographic algorithms. The set of cryptographic algorithms supported by SB FIPS Java Module are given in Table 4.

Table 4: Supported Algorithms and Standards

	Algorithm	**FIPS Approved or Allowed**	**Cert Number**
Block Ciphers	DES (ECB, CBC, CFB64, OFB64)		
	Triple-DES (ECB, CBC, CFB64, OFB64) [FIPS 46-3]	×	#964
	DESX (ECB, CBC, CFB64, OFB64)		
	AES (ECB, CBC, CFB128, OFB128, CTR, CCM, CMAC, GCM) [FIPS 197]	×	#1411
	ARC2 (ECB, CBC, CFB64, OFB64) [RFC 2268]		
Stream Cipher	ARC4		
Hash Functions	SHA-1 [FIPS 180-2]	×	#1281
	SHA-224 [FIPS 180-2]	×	#1281
	SHA-256 [FIPS 180-2]	×	#1281
	SHA-384 [FIPS 180-2]	×	#1281
	SHA-512 [FIPS 180-2]	×	#1281
	MD5 [RFC 1321]		
	MD2 [RFC 1115]		
Message Authentication	HMAC-SHA-1 [FIPS 198]	×	#832
	HMAC-SHA-224 [FIPS 198]	×	#832
	HMAC-SHA-256 [FIPS 198]	×	#832
	HMAC-SHA-384 [FIPS 198]	×	#832
	HMAC-SHA-512 [FIPS 198]	×	#832
	HMAC-MD5 [RFC 2104]		
pRNG	ANSI X9.62 RNG [ANSI X9.62]	×	#773
	DRBG [NIST SP 800-90]	×	#52
Digital Signature	DSA [FIPS 186-2]	×	#455
	ECDSA [FIPS 186-2, ANSI X9.62]	×	#179
	RSA PKCS #1 v1.5 Signature [PKCS #1 v2.1]	×	#687
	RSA PSS [PKCS #1 v2.1]	×	#687
	ECQV		
Key Agreement	Diffie-Hellman [NIST SP 800-56A]	×	#8
	Elliptic Curve Diffie-Hellman [NIST SP 800-56A]	×	#8
	ECMQV [NIST SP 800-56A]	×	#8
Key Wrapping	RSA PKCS #1 v1.5 Encryption [PKCS #1 v2.1]		
	RSA OAEP [NIST SP 800-56B]	×	
	ECIES [ANSI X9.63]		

The Triple-DES, AES (ECB, CBC, CFB128, OFB128, CTR, CCM, GCM, and CMAC modes), SHS (SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512), HMAC-SHS (HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA256, HMAC-SHA-384, and HMAC-SHA-512), pRNG (ANSI X9.62, NIST SP 800-90), DSA, ECDSA, RSA PKCS #1 v1.5 Signature, and RSA PSS algorithms, and NIST SP 800-56A Key Establishment techniques (key agreement) Diffie-Hellman, Elliptic Curve Diffie-Hellman, and ECMQV have been independently tested. SB FIPS Java Module also supports a NIST SP 800-56B Key Establishment technique (key wrapping), RSA OAEP. In order to operate the module in a FIPS Approved mode of operation only these FIPS Approved or allowed algorithms may be used.

DES, DESX, AES CCM* mode, ARC2, ARC4, MD5, MD4, MD2, and HMAC-MD5, ECQV, ECIES and RSA PKCS #1 v1.5 Encryption algorithm are supported as non FIPS Approved algorithms. In order to operate the module in a FIPS Approved mode of operation these algorithms must not be used.

Table 5 summarizes the keys and CSPs used in the FIPS mode.

Table 5: Key and CSP, Key Size, Security Strength, and Access in FIPS mode

Algorithm	**Key and CSP**	Key Size	Strength	**Access**
AES	key	128-256 bits	128-256 bits	Use
Triple-DES	key	112 bits	112 bits	Create, Read, Use
HMAC	key	160-512 bits	80-256 bits	Use
pRNG	seed key, seed	160 bits	80 bits	Use
DSA	key pair	1024-15360 bits	80-256 bits	Create, Read, Use
ECDSA	key pair	160-571 bits	80-256 bits	Create, Read, Use
RSA Signature	key pair	1024-15360 bits	80-256 bits	Create, Read, Use
Diffie-Hellman	static/ephemeral key pair	1024-15360 bits	80-256 bits	Create, Read, Use
Elliptic Curve Diffie-Hellman	static/ephemeral key pair	160-571 bits	80-256 bits	Create, Read, Use
ECMQV	static/ephemeral key pair	160-571 bits	80-256 bits	Create, Read, Use
RSA Key wrapping	key pair	1024-15360 bits	80-256 bits	Create, Read, Use

4.3 Operator Authentication

SB FIPS Java Module does not deploy authentication mechanism. The roles of Crypto Officer and User are implicitly selected by the operator.

5 Finite State Model

The Finite State model contains the following states:

- Installed/Uninitialized
- Initialized
- Self-Test
- Idle
- Crypto Officer/User
- Error

The following is the important features of the state transition:

1. When the module is installed by the Crypto Officer, the module is in the Installed/Uninitialized state.
2. When the initialization command is applied to the module, i.e., the module is loaded on the memory, turning to the Initialized state. Then, it transits to the Self-Test state automatically, running the Power-up Tests. While in the Self-Test state, the module prohibits all data output via the data output interface. On success the module enters Idle; on failure the module enters Error and the module is disabled. From the Error state the Crypto Officer may need to re-install to attempt correction.
3. From the Idle state (which is only entered if Self-Tests have succeeded), the module can transit to the Crypto Officer/User state when an API method is called.
4. When the API method has completed successfully, the state transits back to Idle.
5. If the Conditional Test (Continuous RNG Test or Pair-wise Consistency Test) fails, the state transits to Error and the module is disabled.
6. When On-demand Self-Test is executed, the module enters the Self-Test state. On success the module enters Idle; on failure the module enters Error and the module is disabled.
7. When the de-initialization command is executed, the module goes back to the Installed/Uninitialized state.

6 Physical Security

Physical security is not applicable to this software module at Level 1 Security.

7 Operational Environment

This module is to be run in single user operational environment, where each user application runs in virtually separated independent space. Note that modern Operating Systems such as UNIX, Linux, and Windows provide such operational environment.

8 Cryptographic Key Management

SB FIPS Java Module provides the underlying method to support FIPS 140-2 Level 1 key management. The user will select FIPS Approved algorithms and will handle keys with appropriate care to build up a system that complies with FIPS 140-2. It is the Crypto Officer and User's responsibility to select FIPS 140-2 validated algorithms (see Table 4).

8.1 Key Generation

SB FIPS Module provides FIPS 140-2 compliant key generation. The underlying random number generation uses a FIPS Approved method, ANSI X9.62 RNG or DRBG.

8.2 Key Establishment

SB FIPS Java Module provides the following FIPS allowed key establishment techniques [5]:

1. Diffie-Hellman
2. Elliptic Curve Diffie-Hellman
3. ECMQV
4. RSA OAEP

The Elliptic Curve Diffie-Hellman and ECMQV key agreement technique implementations support elliptic curve sizes from 160 bits to 571 bits that provide between 80 and 256 bits of security strength. The Diffie-Hellman key agreement technique implementation supports modulus sizes from 512 bits to 15360 bits that provide between 56 and 256 bits of security strength, where 1024 bits and above must be used to provide minimum of 80 bits of security in the FIPS Mode. The RSA OAEP key wrapping implementation supports modulus sizes from 512 bits to 15360 bits that provide between 56 and 256 bits of security strength, where 1024 bits and above must be used to provide minimum of 80 bits of security in the FIPS Mode.

It is the users responsibility to ensure that the appropriate key establishment techniques are applied to the appropriate keys.

8.3 Key Entry and Output

Secret (security sensitive) keys must be imported into or exported from the SB FIPS Java Module in encrypted form using a FIPS Approved algorithm when crossing the module's physical boundary.

8.4 Key Storage

SB FIPS Java Module is a low-level cryptographic toolkit, and as such does not provide key storage.

8.5 Zeroization of Keys

SB FIPS Java Module provides zeroizable interfaces which implement zeroization methods. Zeroization of all keys and CSPs are performed in the finalizing methods of the objects; JVM executes the finalizing methods every time it operates garbage collection.

9 Self-Tests

9.1 Power-up Tests

9.1.1 Tests upon Power-up

Self-Tests are initiated automatically by the module at start-up. The following tests are applied:

1. **Known Answer Tests (KATs):**
   KATs are performed on Triple-DES, AES, SHA (via HMAC-SHS), HMAC-SHS, RNG, RSA Signature Algorithm, Diffie-Hellman, Elliptic Curve Diffie-Hellman, ECMQV, and KDF (via key agreement). For DSA and ECDSA, Pair-wise Consistency Test is used.
2. **Software Integrity Test:**
   The software integrity test deploys ECDSA signature validation to verify the integrity of the module.

9.1.2 On-Demand Self-Tests

On-demand self tests may be invoked by the Crypto Officer or User by invoking a method, which is described in the Crypto Officer And User Guide in Appendix A.

9.2 Conditional Tests

The Continuous RNG Test is executed on all RNG generated data, examining the first 160 bits of each requested random generation for repetition. This ensures that the RNG is not stuck at any constant value.

Also, upon each generation of a RSA, DSA, or ECDSA key pair, the generated key pair is tested of their correctness by generating a signature and verifying the signature on a given message as a Pair-wise Consistency Test.

Upon generation or reception of Diffie-Hellman, Elliptic Curve Diffie-Hellman, or ECMQV key pair, the key pair is tested of their correctness by checking shared secret matching of two key agreement parties as a Pair-wise Consistency Test.

9.3 Failure of Self-Tests

Failure of the Self-Tests places the cryptographic module in the Error state, wherein no cryptographic operations can be performed. The module is disabled. Additionally, the cryptographic module will throw a Java exception to the caller.

10 Design Assurance

10.1 Configuration Management

A configuration management system for the cryptographic module is employed and has been described in a document to the certifying lab. It uses the Concurrent Versioning System (CVS) or Subversion (SVN) to track the configurations.

10.2 Delivery and Operation

Please refer to Section A.1 of Crypto Officer And User Guide in Appendix A to review the steps necessary for the secure installation and initialization of the cryptographic module.

10.3 Development

Detailed design information and procedures have been described in documentation submitted to the testing laboratory. The source code is fully annotated with comments, and is also submitted to the testing laboratory.

10.4 Guidance Documents

Crypto Officer Guide and User Guide are provided in Appendix A. This appendix outlines the operations for Crypto Officer and User to ensure the security of the module.

11 Mitigation of Other Attacks

SB FIPS Java Module implements mitigation of the following attacks:

1. Timing attack on RSA
2. Attack on biased private key of DSA

11.1 Timing Attack on RSA

When employing Montgomery computations, timing effects allow an attacker to tell when the base of exponentiation is near the secret modulus. This leaks information concerning the secret modulus.

In order to mitigate this attack, the following is executed: The bases of exponentiation are randomized by a novel technique that requires no inversion to remove.

Note that Remote Timing Attacks are Practical:
`http://crypto.stanford.edu/~dabo/papers/ssl-timing.pdf`

11.2 Attack on Biased Private Key of DSA

The standard for choosing ephemeral values in DSA signature introduce a slight bias. Means to exploit these biases were presented to ANSI by D. Bleichenbacher.

In order to mitigate this attack, the following is executed: The bias in the RNG is reduced to levels which are far below the Bleichenbacher attack threshold.

Change Notice 1 of FIPS 186-2 is published to mitigate this attack:
`http://csrc.nist.gov/CryptoToolkit/tkdigsigs.html`

A Crypto Officer And User Guide

A.1 Installation

In order to carry out a secure installation of SB FIPS Java Module, the Crypto Officer must follow the procedure described in this section.

A.1.1 Installing

The Crypto Officer is responsible for the installation of SB FIPS Java Module. Only the Crypto Officer is allowed to install the product.

Place the cryptographic module, `EccpressoFIPS.jar`, in `CLASSPATH` or as an installed extension.

A.1.2 Uninstalling

Remove the jar file, `EccpressoFIPS.jar`, from the computer hardware.

A.2 Commands

A.2.1 Initialization

`FIPSManager.getInstance().activateFIPSMode()`

This method runs a series of Self-Tests on the module. These tests examine the integrity of the shared object, and the correct operation of the cryptographic algorithms. If these tests are successful, the module will be enabled.

A.2.2 Deinitialization

`FIPSManager.getInstance().deactivateFIPSMode()`

This method de-initializes the module.

A.2.3 Self-Tests

`FIPSManager.getInstance().runSelfTests()`

This method runs a series of Self-Tests, and returns if the tests are successful, otherwise, an exception is thrown. These tests examine the integrity of the shared object, and the correct operation of the cryptographic algorithms. If these tests fail, the module will be disabled. Section A.3 of this document describes how to recover from the disabled state.

A.2.4 Show Status

Status can be found by calling `FIPSManager.getInstance().isFIPSMode()` and `FIPSManager.getInstance().requestCryptoOperation()`. If both methods return true, the module is in the Idle state.

A.3 When Module is Disabled

When SB FIPS Java Module becomes disabled, attempt to bring the module back to the Installed state by calling the deinitialization method, and then to initialize the module using the initialization method. If the initialization is successful, the module is recovered. If this attempt fails, uninstall the module and re-install it. If the module is initialized successfully by this re-installation, the recovery is successful. If this recovery attempt fails, it indicates a fatal error. Please contact Certicom Support immediately.