Security Builder® FIPS Java Module

Version 2.0

FIPS 140-2 Non-Proprietary
Security Policy

Certicom Corp.

September 27, 2005

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 2.0** (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*, December 3, 2002.

[2] NIST *Security Requirements For Cryptographic Modules, Annex A: Approved Security Functions for FIPS PUB 140-2*, March 11, 2004.

[3] NIST *Security Requirements For Cryptographic Modules, Annex B: Approved Protection Profiles for FIPS PUB 140-2*, July 2, 2003.

[4] NIST *Security Requirements For Cryptographic Modules, Annex C: Approved Random Number Generators for FIPS PUB 140-2*, March 17, 2003.

[5] NIST *Security Requirements For Cryptographic Modules, Annex D: Approved Key Establishment Techniques for FIPS PUB 140-2,* February 23, 2004.

[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,* July 26, 2004.

[8] NIST *Frequently Asked Questions for the Cryptographic Module Validation Program*, December 18, 2003.

1.4 Change Notes

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

```
$Log: SBFIPSJModule2SecurityPolicy.tex,v $
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 DH, ECDH, 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, consisting of 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.
- SB FIPS Java Module software that runs on the computer hardware, OS, and JVM.

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:

1. Windows 2003, x86 (Binary compatible to Windows 98/2000/XP)
2. Red Hat Linux Application Server 3.0, x86 (Binary compatible to AS 2.1)
3. Solaris 2.9, SPARC 32-bit SPARC
4. Solaris 2.9, SPARC 64-bit SPARC

The Java Runtime Environment (JRE) 1.3.1 and 1.4.2 by Sun Microsystems are used to test SB FIPS Java Module on the platforms above. The module will run on the equivalent JREs on various hardware and OS, while maintaining its compliance to the FIPS 140-2 Level 1 requirements. Such hardware and OS include the following:

5. Any other Windows Platforms,
6. Any other Linux Platforms,
7. AIX Platforms, and
8. HP-UX Platforms.

Thus, this validation is applicable to these platforms as well.

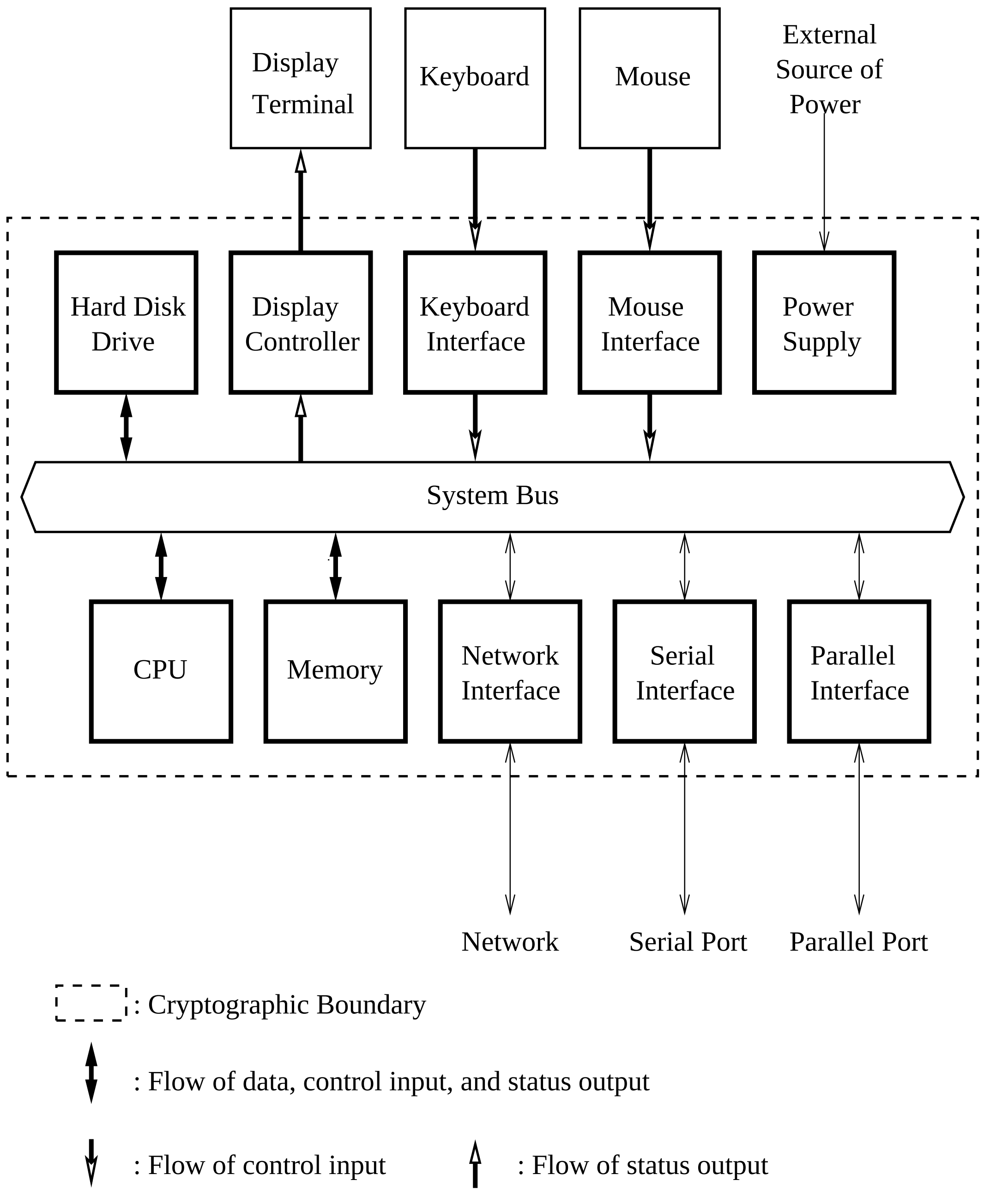

Figure 1: Cryptographic Module Hardware Block Diagram

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).

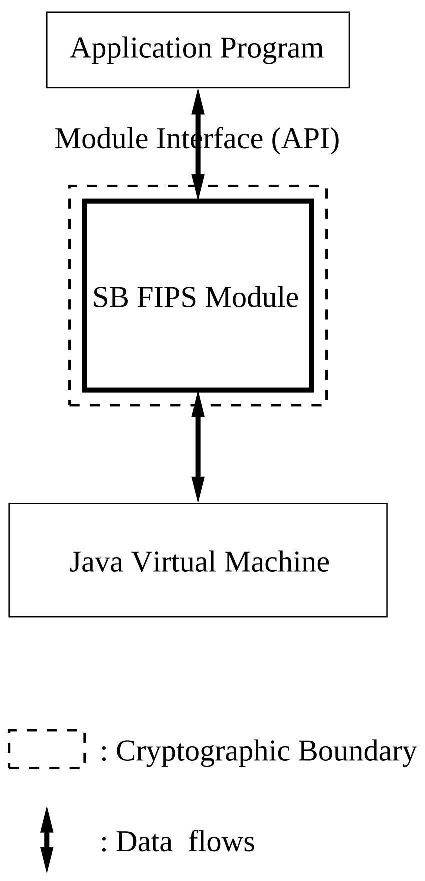

Figure 2: Cryptographic Module Software Block Diagram

3 Cryptographic Module Ports and Interfaces

The physical and logical interfaces are summarized in Table 1.

Table 1: 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 `FIPSManager.activateFIPSMode()`	The Power Supply is the power interface.

4 Roles, Services, and Authentication

4.1 Roles

SB FIPS Java Module supports Crypto Officer and User Roles, meeting FIPS 140-2 Level 1 requirements. These roles are enforced by this Security Policy. The Crypto Officer has the responsibility for installing SB FIPS Java Module (see Table 2).

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 3.

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 3.

The DES (DES is provided for legacy systems), TDES, AES, 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), RNG, DSA, RSA PKCS #1, and ECDSA algorithms have been validated to comply with FIPS. SB FIPS Java Module also supports FIPS Approved key establishment techniques (key agreement and key wrapping), DH, ECDH, ECMQV, and RSA PKCS #1. In order to operate the module in compliance with FIPS, only these FIPS Approved algorithms should be used.

DESX, ARC2, ARC4, MD5, MD2, and HMAC-MD5 are supported as non FIPS Approved algorithms. In order to operate the module in compliance with FIPS, these algorithms should not be used.

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.

Table 2: Roles, Services and Access

Service	Crypto Officer	User	Keys and CSPs	Access
Installation, etc.				
Installation	×			
Uninstallation	×			
Self-Tests	×	×	ECDSA public key	Use
Show status	×	×		
Symmetric Ciphers				
Key generation	×	×	AES, TDES, DES keys (DES for legacy systems)	Create, Read
Encrypt	×	×	AES, TDES, DES keys (DES for legacy systems)	Use
Decrypt	×	×	AES, TDES, DES keys (DES for legacy systems)	Use
Hash Algorithms and Message Authentication				
Hashing	×	×		
Msg. Authentication	×	×	HMAC keys	Use
Random Number Generation				
Seeding	×	×	Seed	Use
Request	×	×		
Digital Signature				
Key pair generation	×	×	RSA, DSA, ECDSA key pairs	Create, Read
Sign	×	×	RSA, DSA, ECDSA private keys	Use
Verify	×	×	RSA, DSA, ECDSA public keys	Use
Key Agreement				
Key pair generation	×	×	ECDH, ECMQV, DH key pairs	Create, Read
Shared secret generation	×	×	ECDH, ECMQV, DH key pairs	Use
Key Wrapping				
Key pair generation	×	×	RSA key pairs	Create, Read
Wrap	×	×	RSA public keys	Use
Unwrap	×	×	RSA private keys	Use

Table 3: Supported Algorithms and Standards

	Algorithm	**FIPS Approved**	**Cert Number**
Block Ciphers	DES (transitional phase only, valid until May 19, 2007) (ECB, CBC, CFB64, OFB64) [FIPS 46-3]	×	#298
	TDES (ECB, CBC, CFB64, OFB64) [FIPS 46-3]	×	#318
	DESX (ECB, CBC, CFB64, OFB64)		
	AES (ECB, CBC, CFB128, OFB128) [FIPS 197]	×	#227
	ARC2 (ECB, CBC, CFB64, OFB64) [RFC 2268]		
Stream Cipher	ARC4		
Hash Functions	SHA-1 [FIPS 180-2]	×	#307
	SHA-224 [FIPS 180-2]	×	#307
	SHA-256 [FIPS 180-2]	×	#307
	SHA-384 [FIPS 180-2]	×	#307
	SHA-512 [FIPS 180-2]	×	#307
	MD5 [RFC 1321]		
	MD2 [RFC 1115]		
Message Authentication	HMAC-SHA-1 [FIPS 198]	×	#37
	HMAC-SHA-224 [FIPS 198]	×	#37
	HMAC-SHA-256 [FIPS 198]	×	#37
	HMAC-SHA-384 [FIPS 198]	×	#37
	HMAC-SHA-512 [FIPS 198]	×	#37
	HMAC-MD5 [RFC 2104]		
RNG	ANSI X9.62 RNG [ANSI X9.62]	×	#68
Digital Signature	DSS [FIPS 186-2]	×	#128
	ECDSA [FIPS 186-2, ANSI X9.62]	×	#6
	RSA PKCS1-v1_5 [PKCS #1 v2.1]	×	#54
Key Agreement	DH [ANSI X9.42]	×	
	ECDH [ANSI X9.63]	×	
	ECMQV [ANSI X9.63]	×	
Key Wrapping	RSA PKCS1-v1_5 [PKCS #1 v2.1]	×	
	RSA OAEP [PKCS #1 v2.1]	×	

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 Initialization state. Then, it transits to the Self-Test state automatically, running the Power-up Tests. While in the Self-Test state, all data output via the data output interface is prohibited. 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. While in the Self-Test state, all data output via the data output interface is prohibited. 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 mode on the target operational environment such as UNIX, Linux, or Windows. Therefore the module is restricted to single operator mode operation by the operating system. This section describes the settings for single user mode to be followed.

7.1 UNIX

On UNIX systems, the following operations must be applied [8]:

- Remove all login accounts except "root" (the super user).
- Disable NIS and other name services for users and groups.
- Turn off all remote login, remote command execution, and file transfer daemons.

For example, on HP-UX and Solaris, the following must be applied:

1. Login as the "root" user.
2. Edit the system file `/etc/passwd` and remove all the users except "root" and the pseudo-users. Make sure the password fields for the pseudo-users are a star ($*$). This prevents login as the pseudo-users.
3. Edit the system file `/etc/nsswitch.conf`. Make sure that "files" is the only option for "passwd" and "group". This disables NIS and other name services for users and groups.
4. Edit the system file `/etc/inetd.conf`. Remove or comment out the lines for remote login, remote command execution, and file transfer daemons such as `telnetd`, `rlogind`, `remshd`, `rexecd`, `ftpd`, and `tftpd`.
5. Reboot the system for the changes to take effect.

For IBM AIX, the following:

1. Login is as the "root" user.

2. Edit the system file `/etc/passwd` and remove all the users except “root” and the pseudo-users. Make sure that for the pseudo-uses, either the password files are a star ($*$) or the login shell fields are empty. This prevents login as the pseudo-users.

3. Remove all lines that begin with a plus sign ($+$) or minus sign ($-$) from `/etc/passwd` and `/etc/group`. This disables NIS and other name services for users and groups.

4. Edit the system file `/etc/inetd.conf`. Remove or comment out the lines for remote login, remote command execution, and file transfer daemons such as `telnetd`, `rlogind`, `remshd`, `rexecd`, `ftpd`, and `tftpd`.

5. Reboot the system for the changes to take effect.

7.2 Linux

On Linux systems, the following operations must be applied:

- Remove all login accounts except “root” (the super user).
- Disable NIS and other name services for users and groups.
- Turn off all remote login, remote command execution, and file transfer daemons.

For example, on RedHat Linux, the following must be applied:

1. Login as the “root” user.

2. Edit the system file `/etc/passwd` and `/etc/shadow` and remove all the users except “root” and the pseudo-users. Make sure the password fields in `/etc/shadow` for the pseudo-users are either a star ($*$) or double exclamation mark (!!). This prevents login as the pseudo-users.

3. Edit the system file `/etc/nsswitch.conf` and make “files” the only option for “passwd”, “group”, and “shadow”. This disables NIS and other name services for users and groups.

4. In the `/etc/inetd.d` directory, edit the files “rexec”, “rlogin”, “rsh”, “rsync”, “telnet”, and “wu-ftpd”, and set the value of “disable” to “yes”.

5. Reboot the system for the changes to take effect.

7.3 Windows

In order to ensure single user mode, the following rules:

- All remote guest account must be disabled in order to ensure that up to one human operator can operate SB FIPS Java Module at a time.
- The use of SB FIPS Java Module is restricted to a single instance of loading the module to the memory space at a time.
- The actual memory space of the virtual memory space must be on the local machine.

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 3).

8.1 Key Generation

SB FIPS Module provides FIPS 140-2 compliant key generation. The underlying random number generation uses a FIPS Approved method, the ANSI X9.62 RNG [4].

8.2 Key Establishment

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

1. Diffie-Hellman (DH)
2. EC Diffie-Hellman (ECDH)
3. ECMQV
4. RSA PKCS1-v1_5
5. RSA OAEP

The RSA key wrapping techniques above are based on the PKCS #1 v2.1 standard, and are used to transport keys.

The DH key agreement technique implementation supports modulus sizes up to 15360 bits that provide 256 bits of security. The ECDH and ECMQV key agreement technique implementations support elliptic curve sizes above 512 bits that provide 256 bits of security. The RSA implementation supports modulus sizes up to 15360 bits that provide 256 bits of security.

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

8.3 Key Entry and Output

It is not allowed to import or export keys, including a seed key, and other security sensitive information outside of physical boundary in plaintext format. For key entry and export, users must ensure to deploy appropriate encryption method using FIPS Approved encryption algorithms in SB FIPS Java Module or any other FIPS validated cryptographic module.

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 a zeroizable interface which implements a zeroization method. Zeroization of all keys and CSPs are performed in the finalizing method of the objects; JVM executes the finalizing methods every time it operates garbage collection. Also, a zeroization class is provided, where it will invoke the zeroize method when given an object that implements the zeroizable interface.

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 DES (DES is provided for legacy systems), TDES, AES, SHS (via HMAC-SHS), HMAC-SHS, RNG, and RSA PKCS #1 v1.5 Signature Algorithm. 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 Cryptographic 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.

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. If any Self-Test fails, the cryptographic module will output error code.

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) 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 certifying laboratory.

This toolkit is designed and developed using high level language Java for Java users.

Development for the cryptographic module is carried out in a multi-platform environment. Certicom is using the CVS revision control system to control revisions. Development of new versions and major features are performed on a branch of the software, and these branches merged back into the trunk after testing and review, but the branch is maintained to perform post-release maintenance.

Releases are tested first in engineering (via an automated daily procedure, the daily build). They are then committed to the product candidate repository. These releases are then sent to the QA department which must run its own tests before they move them into the product branch. Only QA can move candidates into products.

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 (unlike other blinding methods e.g. BSAFE Crypto-C User Manual v 4.2).

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 standards for choosing ephemeral values in El-Gamal type signatures 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. The Crypto Officer must have administrative privileges on the computer.

Place `EccpressoFIPS.jar` file in `CLASSPATH` or as an installed extension.

A.1.2 Uninstalling

Remove the jar files from the computer hardware.

A.2 Commands

A.2.1 Initialization

`FIPSManager.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.deactivateFIPSMode()`

This method de-initializes the module.

A.2.3 Self-Tests

`FIPSManager.runSelfTest()`

This method runs a series of Self-Tests, and returns true if the tests are successful. 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.isFIPSMode()` and `FIPSManager.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.