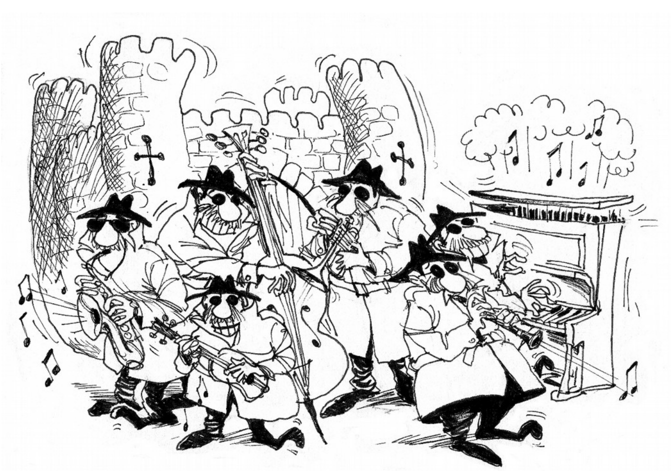
Legion of the Bouncy Castle Inc.
BC-FJA (Bouncy Castle FIPS Java API)

Non-Proprietary FIPS 140-2 Cryptographic Module Security Policy

Version: 0.5 Date: 08/03/16



Legion of the Bouncy Castle Inc.

(ABN 84 166 338 567)

https://www.bouncycastle.org

Table of Contents

List of Tables

List of Figures

1 Introduction

This document defines the Security Policy for the Legion of the Bouncy Castle Inc. FIPS Java API (BC-FJA) Module, hereafter denoted the Module. The Module is a cryptographic library. The Module meets FIPS 140-2 overall Level 1 requirements. The SW version is 1.0.0 .

The cryptographic module was tested on the following operational environment on the general purpose computer (GPC) platforms detailed in Table 1.

Operational Environments			
GPC Platform	**CPU Family**	**OS**	**Java SE Runtime Environment**
Cisco UCSB-B200-M4 Blade	Intel Xeon Processor E5 v3	Solaris 11 on vSphere 6	Java SE Runtime Environment v7 (1.7.0), single-user mode
		Centos 6.4 on vSphere 6	Java SE Runtime Environment v8 (1.8.0), single-user mode

Table 1 – Cryptographic Module Tested Environments

As per FIPS 140-2 Implementation Guidance G.5, the cryptographic module will remain compliant with the FIPS 140-2 validation when operating on any general purpose computer (GPC) provided that:

1) No source code has been modified.
2) The GPC uses the specified single-user platform, or another compatible single-user platform such as one of the Java SE Runtime Environments listed on any of the following:

    HP-UX
    Linux Centos
    Linux Debian
    Linux Oracle RHC
    Linux Oracle UEK
    Linux SUSE
    Linux Ubuntu
    Mac OS X
    Microsoft Windows
    Microsoft Windows Server
    Microsoft Windows XP
    RedHat Enterprise Linux

For the avoidance of doubt, it is hereby stated that the CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

The Module is intended for use by US Federal agencies and other markets that require a FIPS 140-2 validated Cryptographic Library. The Module is a software-only embodiment; the cryptographic boundary is the Java Archive (JAR) file, *bc-fips-1.0.0.jar*.

The FIPS 140-2 security levels for the Module are given in Table 2 as follows:

Table 2 – Security Level of Security Requirements

Security Requirement	Security 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
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	1

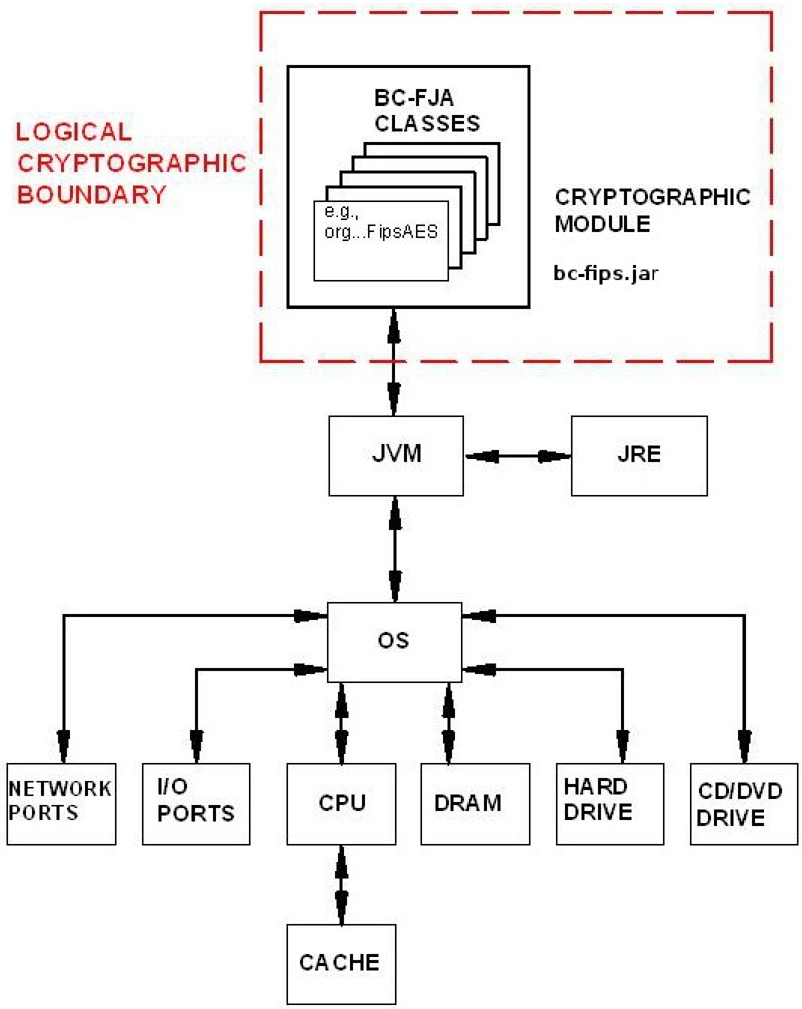
1.1 Logical and Physical Cryptographic Boundaries

1.1.1 Logical Cryptographic Boundary

The executable for the BC-FJA Module is*: bc-fips-1.0.0.jar*. This module is the only software component within the Logical Cryptographic Boundary and the only software component that carries out cryptographic functions covered by FIPS 140-2. Figure 1 shows the logical relationship of the cryptographic module to the other software and hardware components of the computer. The BC classes are executed on the Java Virtual Machine (JVM) using the classes of the Java Runtime Environment (JRE). The JVM is the interface to the computer's Operating System (OS) that is the interface to the various physical components of the computer.

The physical components of the computer are discussed further in Section 6. Abbreviations introduced in Figure 1 that describe physical components are: Central Processing Unit (CPU), Dynamic Random Access Memory (DRAM) and Input Output (I/O).

Figure 1 – Block Diagram of the Software for the BC-FJA Module.

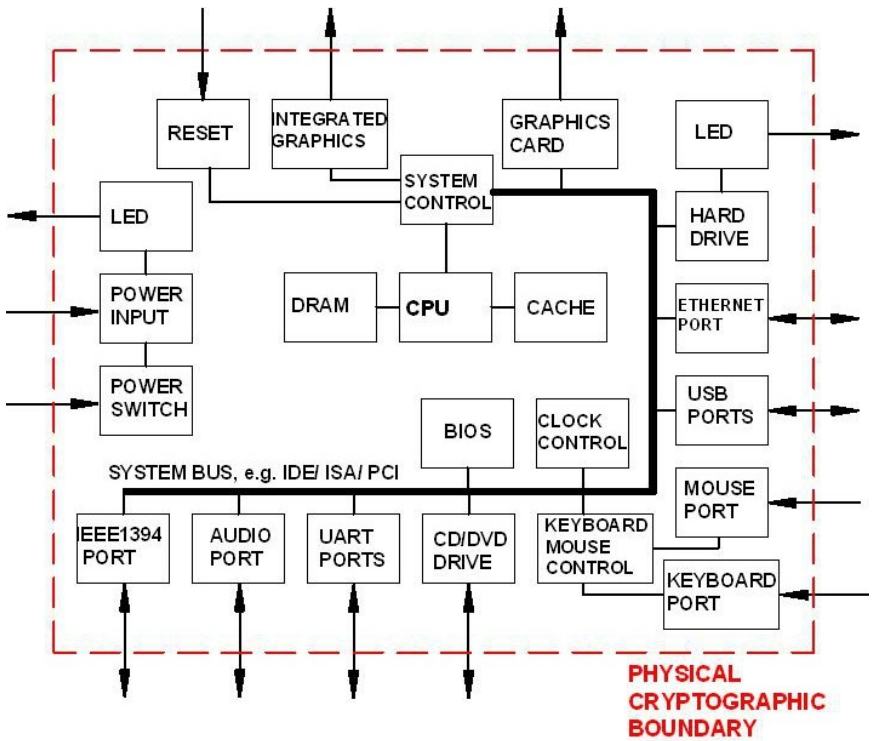


1.1.2 Physical Boundary

The BC-FJA Module runs on a General Purpose Computer (GPC). The Physical Cryptographic Boundary for the module is the case of that computer. Figure 2 shows a block diagram of the physical components of a typical GPC and the ports or interfaces across the Physical Cryptographic Boundary. All the physical components are standard electronic components; there are not any custom integrated circuits or components dedicated to FIPS 140-2 related functions.

Abbreviations introduced in Figure 2 are: Basic I/O System (BIOS), Integrated Device Electronics (IDE), Institute of Electrical and Electronic Engineers (IEEE), Instruction Set Architecture (ISA), Peripheral Component Interconnect (PCI), Universal Asynchronous Receiver/Transmitter (UART) and Universal Serial Bus (USB). Input or output ports are designated by arrows with single heads, while I/O ports are indicated by bidirectional arrows.

Figure 2 – Block Diagram of the Physical Components of a typical GPC.



For FIPS 140-2 purposes, the BC-FJA Module is defined as a "multi-chip standalone module", therefore, the module's physical ports or interfaces are defined as those for the hardware of the GPC. These physical ports are separated into the logical interfaces defined by FIPS 140-2, as shown in Table 3.

The BC-FJA Module is a software module only, and, therefore, control of the physical ports is outside of the module's scope. The module does provides a set of logical interfaces which are mapped to the following FIPS 140-2 defined logical interfaces: data input, data output, control input, status output, and power. When the module performs self-tests, is in an error state, is generating keys, or performing zeroization, the module prevents all output on the logical data output interface as only the thread performing the operation has access to the data. The module is single-threaded, and in an error state, the module does not return any output data, only an error value.

The mapping of the FIPS 140-2 logical interfaces to the module is described in table 3.

Table 3 – FIPS 140-2 Logical Interfaces

Interface	Module Equivalent
Data Input	API input parameters – plaintext and/or ciphertext data.
Data Output	API output parameters and return values – plaintext and/or ciphertext data.
Control Input	API method calls – method calls, or input parameters, that specify commands and/or control data used to control the operation of the module.
Status Output	API output parameters and return/error codes that provide status information used to indicate the state of the module.
Power	Start up/Shutdown of a process containing the module.

1.2 Modes of Operation

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

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

1.3 Module Configuration

In default operation the module will start with both approved and non-approved mode enabled.

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

Use of the module with a Java Security manager requires the setting of some basic permissions to allow the module HMAC-SHA-256 software integrity test to take place as well as to allow the module itself to examine secret and private keys. The basic permissions required for the module to operate correctly with a Java Security manager are indicated by a Y in the **Req** column of Table 4.

Table 4 – Available Java Permissions

Permission	Settings	Req	Usage
RuntimePermission	`“getProtectionDomain”`	Y	Allows checksum to be carried out on jar.
RuntimePermission	`“accessDeclaredMembers”`	Y	Allows use of reflection API within the provider.

PropertyPermission	`“java.runtime.name”, “read”`	N	Only if configuration properties are used.
SecurityPermission	`"putProviderProperty.BCFIPS"`	N	Only if provider installed during execution.
CryptoServicesPermission	`“unapprovedModeEnabled”`	N	Only if unapproved mode algorithms required.
CryptoServicesPermission	`“changeToApprovedModeEnabled”`	N	Only if threads allowed to change modes.
CryptoServicesPermission	`“exportSecretKey”`	N	To allow export of secret keys only.
CryptoServicesPermission	`“exportPrivateKey”`	N	To allow export of private keys only.
CryptoServicesPermission	`“exportKeys”`	Y	Required to be applied for the module itself. Optional for any other codebase.
CryptoServicesPermission	`“tlsNullDigestEnabled”`	N	Only required for TLS digest calculations.
CryptoServicesPermission	`“tlsPKCS15KeyWrapEnabled”`	N	Only required if TLS is used with RSA encryption.
CryptoServicesPermission	`“tlsAlgorithmsEnabled”`	N	Enables both NullDigest and PKCS15KeyWrap.
CryptoServicesPermission	`“defaultRandomConfig”`	N	Allows setting of default SecureRandom.
CryptoServicesPermission	`“threadLocalConfig”`	N	Required to set a thread local property in the CryptoServicesRegistrar
CryptoServicesPermission	`“globalConfig”`	N	Required to set a global property in the CryptoServicesRegistrar.

2 Cryptographic Functionality

The Module implements the FIPS Approved and Non-Approved but Allowed cryptographic functions listed in Table 5 to Table 7, below.

Table 5 – Approved and CAVP Validated Cryptographic Functions

Algorithm	Description	Cert #
AES	[FIPS 197, SP 800-38A] Functions: Encryption, Decryption Modes: ECB, CBC, OFB, CFB8, CFB128, CTR Key sizes: 128, 192, 256 bits	3756
CCM	[SP 800-38C] Functions: Generation, Authentication Key sizes: 128, 192, 256 bits	3756
CMAC	[SP 800-38B] Functions: Generation, Authentication Key sizes: AES with 128, 192, 256 bits and Triple-DES with 2-key[1,2], 3-key	3756 (AES), 2090 (Triple-DES)
GCM/GMAC[3]	[SP 800-38D] Functions: Generation, Authentication Key sizes: 128, 192, 256 bits	3756
DRBG	[SP 800-90A] Functions: Hash DRBG, HMAC DRBG, CTR DRBG Security Strengths: 112, 128, 192, and 256 bits	1031
DSA[4]	[FIPS 186-4] Functions: PQG Generation, PQG Verification, Key Pair Generation, Signature Generation, Signature Verification Key sizes: 1024, 2048, 3072 bits (1024 only for SigVer)	1043

[1] 2^20 block limit is enforced by module

[2] In approved mode of operation, the use of 2-key Triple-DES to generate MACs for anything other than verification purposes is non-compliant.

[3] GCM with an internally generated IV, see section 8.3 concerning external IVs. IV generation is compliant with IG A.5.

[4] DSA signature generation with SHA-1 is only for use with protocols.

Algorithm	Description	Cert #
ECDSA	[FIPS 186-4] Functions: Signature Generation Component, Public Key Generation, Signature Generation, Signature Verification, Public Key Validation Curves/Key sizes: P-192*, P-224, P-256, P-384, P-521, K-163*, K-233, K-283, K-409, K-571, B-163*, B-233, B-283, B-409, B-571 * Curves only used for Signature Verification and Public Key Validation	804, 705 (CVL)
HMAC	[FIPS 198-1] Functions: Generation, Authentication SHA sizes: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256	2458
KAS[5]	[SP 800-56A-rev2] Parameter sets/Key sizes: FB, FC, EB, EC, ED, EE	73
KDF, Existing Application-Specific[6]	[SP 800-135] Functions: TLS v1.0/1.1 KDF, TLS 1.2 KDF, SSH KDF, X9.63 KDF, IKEv2 KDF, SRTP KDF.	704 (CVL)
KDF, using Pseudorandom Functions[7]	[SP 800-108] Modes: Counter Mode, Feedback Mode, Double-Pipeline Iteration Mode Functions: CMAC-based KDF with AES, 2-key Triple-DES, 3-key Triple-DES or HMAC-based KDF with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	78
Key Wrapping Using Block Ciphers[8]	[SP 800-38F] Modes: KW, KWP, TKW	3756 (AES), 2090 (Triple-DES)
RSA	[FIPS 186-4, FIPS 186-2, ANSI X9.31-1998 and PKCS #1 v2.1 (PSS and PKCS1.5)] Functions: Key Pair Generation, Signature Generation, Signature Verification, Component Test Key sizes: 2048, 3072 bits (1024, 1536, 4096 only for SigVer)	1932, 706 (CVL)

[5] Keys are not established directly into the module using the key agreement algorithms.

[6] These protocols have not been reviewed or tested by the CAVP and CMVP.

[7] Note: CAVP testing is not provided for use of the PRFs SHA-512/224 and SHA-512/256. These must not be used in approved mode.

[8] Keys are not established directly into the module using key unwrapping.

Algorithm	Description	Cert #
SHA	[FIPS 180-4], Functions: Digital Signature Generation, Digital Signature Verification, non-Digital Signature Applications SHA sizes: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256	3126
SHA-3, SHAKE	[FIPS 202] SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256	3
Triple-DES (Triple-DES)	[SP 800-20] Functions: Encryption, Decryption Modes: TECB, TCBC, TCFB64, TCFB8, TOFB, CTR Key sizes: 2-key[9], 3-key	2090

Table 6 – Approved Cryptographic Functions Tested with Vendor Affirmation

Algorithm	Description	IG Ref.
AES-CBC Ciphertext Stealing (CS)	[Addendum to SP 800-38A, Oct 2010] Functions: Encryption, Decryption Modes: CBC-CS1, CBC-CS2, CBC-CS3 Key sizes: 128, 192, 256 bits	Vendor Affirmed IG A.3
KAS[10] using SHA-512/224 or SHA-512/256	[SP 800-56A-rev2] Parameter sets/Key sizes: FB, FC, EB, EC, ED, EE[11]	Vendor Affirmed IG A.3
KDF, Password-Based	[SP 800-132] Options: PBKDF with Option 1a Functions: HMAC-based KDF using SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	Vendor Affirmed IG D.6
Key Agreement[10] Using RSA	[SP 800-56B] RSA-KEMS-KWS with, and without, key confirmation. Key sizes: 2048, 3072 bits	Vendor Affirmed IG D.4

[9] 2^20 block limit is enforced by the module, encryption is disabled.

[10] Keys are not directly established into the module using key agreement or transport techniques.

[11] Note: HMAC SHA-512/224 must not be used with EE.

Algorithm	Description	IG Ref.
Key Transport[10] Using RSA	[SP 800-56B] RSA-OAEP with, and without, key confirmation. Key sizes: 2048, 3072 bits	Vendor Affirmed IG D.4

Table 7 – Non-Approved but Allowed Cryptographic Functions

Algorithm	Description
Non-SP 800-56A-rev2 Compliant DH	[IG D.8] Diffie-Hellman 2048-bit key agreement primitive for use with system-level key establishment; not used by the module to establish keys within the module (key agreement; key establishment methodology provides 112 bits of encryption strength)
Non-SP 800-56B compliant RSA Key Transport	[IG D.9] RSA May be used by a calling application as part of a key encapsulation scheme. Key sizes: 2048 and 3072 bits
MD5 within TLS	[IG D.2]

Table 8 – Non-Approved Cryptographic Functions for use in non-FIPS mode only.

AES (non-compliant[12])	RIPEMD128
ARC4 (RC4)	HMAC-RIPEMD128
Blowfish	RIPEMD-160
Camellia	HMAC-RIPEMD160
CAST5	RIPEMD256
DES	HMAC-RIPEMD256
DSA (non-compliant[13])	RIPEMD320
DSTU4145	HMAC-RIPEMD320
ECDSA (non-compliant[14])	X9.31 PRNG
ElGamal	RSA (non-compliant[16])
GOST28147	RSA KAS (non-compliant[15])

[12] Support for additional modes of operation.

[13] Deterministic signature calculation, support for additional digests, and key sizes.

[14] Deterministic signature calculation, support for additional digests, and key sizes.

GOST3410-1994	SCrypt
GOST3410-2001	SEED
GOST3411	Serpent
HMAC-GOST3411	SipHash
IDEA	SHACAL-2
KAS (non-compliant[15])	TIGER
MD5	HMAC-TIGER
HMAC-MD5	Triple-DES (non-compliant[17])
OpenSSL PBKDF	Twofish
PKCS#12 PBKDF	WHIRLPOOL
PKCS#5 Scheme 1 PBKDF	HMAC-WHIRLPOOL
RC2	

2.1 Critical Security Parameters

All CSPs used by the Module are described in this section in Table 9. All usage of these CSPs by the Module (including all CSP lifecycle states) is described in the services detailed in Section 3.2.

Table 9 – Critical Security Parameters (CSPs)

CSP	Description / Usage
AES Encryption Key	[FIPS-197, SP 800-56C, SP 800-38D, Addendum to SP 800-38A] AES (128/192/256) encrypt key[18]
AES Decryption Key	[FIPS-197, SP 800-56C, SP 800-38D, Addendum to SP 800-38A] AES (128/192/256) decrypt key
AES Authentication Key	[FIPS-197] AES (128/192/256) CMAC/GMAC key
AES Wrapping Key	[SP 800-38F] AES (128/192/256) key wrapping key
DH Agreement key	[SP 800-56A-rev2] Diffie-Hellman (>= 2048) private key agreement key

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

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

[17] Support for additional modes of operation.

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

CSP	Description / Usage
DRBG(CTR AES)	V (128 bits) and AES key (128/192/256), entropy input (length dependent on security strength)
DRBG(CTR Triple-DES)	V (64 bits) and Triple-DES key (192), entropy input (length dependent on security strength)
DRBG(Hash)	V (440/888 bits) and C (440/888 bits), entropy input (length dependent on security strength)
DRBG(HMAC)	V (160/224/256/384/512 bits) and Key (160/224/256/384/512 bits), entropy input (length dependent on security strength)
DSA Signing Key	[FIPS 186-4] DSA (2048/3072) signature generation key
EC Agreement Key	[SP 800-56A-rev2] EC (All NIST defined B, K, and P curves >= 224 bits) private key agreement key
EC Signing Key	[FIPS 186-4] ECDSA (All NIST defined B, K, and P curves >= 224 bits) signature generation key.
HMAC Authentication Key	[FIPS 198-1] Keyed-Hash key (SHA-1, SHA-2). Key size determined by security strength required (>= 112 bits)
IKEv2 Derivation Function Secret Value	[SP 800-135] Secret value used in construction of key for the specified IKEv2 PRF.
PBKDF Secret Value	[SP 800-132] Secret value used in construction of Keyed-Hash key for the specified PRF.
RSA Signing Key	[FIPS 186-4] RSA (>= 2048) signature generation key
RSA Key Transport Key	[SP 800-56B] RSA (>=2048) key transport (decryption) key
SP 800-56A-rev2 Concatenation Derivation Function	[SP 800-56A-rev2] Secret value used in construction of key for underlying PRF.
SP 800-108 KDF Secret Value	[SP 800-108] Secret value used in construction of key for the specified PRF.
SRTP Derivation Function Secret Value	[SP 800-135] Secret value used in construction of key for the specified SRTP PRF.
SSH Derivation Function Secret Value	[SP 800-135] Secret value used in construction of key for the specified SSH PRF.
TLS KDF Secret Value	[SP 800-135] Secret value used in construction of Keyed-Hash key for the specified TLS PRF.
Triple-DES Authentication Key	[SP 800-67] Triple-DES (128/192) CMAC key
Triple-DES Encryption Key	[SP 800-67] Triple-DES (192) encryption key

CSP	Description / Usage
Triple-DES Decryption Key	[SP 800-67] Triple-DES (128/192) decryption key
Triple-DES Wrapping Key	[SP 800-38F] Triple-DES (192 bits) key wrapping/unwrapping key, (128 unwrapping only).
X9.63 KDF Secret Value	[SP 800-135] Secret value used in construction of Keyed-Hash key for the specified X9.63 PRF.

2.2 Public Keys

Table 10 – Public Keys

CSP	Description / Usage
DH Agreement Key	[SP 800-56A-rev2] Diffie-Hellman (>= 2048) public key agreement key
DSA Verification Key	[FIPS 186-4] DSA (1024/2048/3072) signature verification key
EC Agreement Key	[SP 800-56A-rev2] EC (All NIST defined B, K, and P curves) public key agreement key
EC Verification Key	[FIPS 186-4] ECDSA (All NIST defined B, K, and P curves) signature verification key
RSA Key Transport Key	[SP 800-56B] RSA (>=2048) key transport (encryption) key.
RSA Verification Key	[FIPS 186-4] RSA (>= 1024) signature verification key

3 Roles, Authentication and Services

3.1 Assumption of Roles

The module supports two distinct operator roles, User and Cryptographic Officer (CO). The cryptographic module implicitly maps the two roles to the services. A user is considered the owner of the thread that instantiates the module and, therefore, only one concurrent user is allowed.

Table 11 lists all operator roles supported by the module. The module does not support a maintenance role and/or bypass capability. The module does not support authentication.

Table 11 – Roles Description

Role ID	Role Description	Authentication Type
CO	Cryptographic Officer – Powers on and off the module.	N/A – Authentication not required for Level 1
User	User – The user of the complete API.	N/A – Authentication not required for Level 1

3.2 Services

All services implemented by the Module are listed in Table 12 below and Table 13 describes all usage of CSPs by the service.

Table 12 lists the services. The second column provides a description of each service and availability to the Cryptographic Officer and User, in columns 3 and 4, respectively.

Table 12 – Services

Service	Description	CO	U
Initialize Module and Run Self-Tests on Demand	The JRE will call the static constructor for self-tests on module initialization.	X	
Show Status	A user can call *FipsStatus.IsReady()* at any time to determine if the module is ready. CryptoServicesRegistrar.*IsInApprovedOnlyMode()* can be called to determine the FIPS mode of operation.		X
Zeroize / Power-off	The module uses the JVM garbage collector on thread termination.		X
Data Encryption	Used to encrypt data.		X
Data Decryption	Used to decrypt data.		X
MAC Calculation	Used to calculate data integrity codes with CMAC.		X
Signature Authentication	Used to generate signatures (DSA, ECDSA, RSA).		X
Signature Verification	Used to verify digital signatures.		X
DRBG (SP800-90A) output	Used for random number, IV and key generation.		X
Message Hashing	Used to generate a SHA-1, SHA-2, or SHA-3 message digest, SHAKE output.		X
Keyed Message Hashing	Used to calculate data integrity codes with HMAC.		X

Service	Description	CO	U
TLS Key Derivation Function	(secret input) (outputs secret) Used to calculate a value suitable to be used for a master secret in TLS from a pre-master secret and additional input.		X
SP 800-108 KDF	(secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input.		X
SSH Derivation Function	(secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input.		X
X9.63 Derivation Function	(secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input.		X
SP 800-56A-rev2 Concatenation Derivation Function	(secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input.		X
IKEv2 Derivation Function	(secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input.		X
SRTP Derivation Function	(secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input.		X
PBKDF	(secret input) (outputs secret) Used to generate a key using an encoding of a password and an additional function such as a message hash.		X
Key Agreement Schemes	Used to calculate key agreement values (SP 800-56A, Diffie-Hellman).		X
Key Wrapping	Used to encrypt a key value. (RSA, AES, Triple-DES)		X
Key Unwrapping	Used to decrypt a key value. (RSA, AES, Triple-DES)		X
NDRNG Callback	Gathers entropy in a passive manner from a user-provided function		X
Utility	Miscellaneous utility functions, does not access CSPs		X

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

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

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

Table 13 defines the relationship between access to CSPs and the different module services. The modes of access shown in the table are defined as:

- G = Generate: The module generates the CSP.
- R = Read: The module reads the CSP. The read access is typically performed before the module uses the CSP.
- E = Execute: The module executes using the CSP.
- W = Write: The module writes the CSP. The write access is typically performed after a CSP is imported into the module, when the module generates a CSP, or when the module overwrites an existing CSP.
- Z = Zeroize: The module zeroizes the CSP.

Table 13 – CSP Access Rights within Services

Service	CSPs									
	AES Keys	DH Keys	DRBG Keys	DSA Keys	EC Agreement Key	ECDSA Key	HMAC Keys	KDF Secret Values	RSA Keys	Triple-DES Keys
Initialize Module and Run Self-Tests on Demand										
Show Status										
Zeroize / Power-off	Z	Z	Z	Z	Z	Z	Z		Z	Z
Data Encryption	R									R
Data Decryption	R									R
MAC Calculation	R						R			R
Signature Authentication				R		R			R	
Signature Verification				R		R			R	
DRBG (SP800-90A) output	G	G	G,R	G	G	G	G		G	G
Message Hashing										
Keyed Message Hashing							R			
TLS Key Derivation Function								R		
SP 800-108 KDF								R		
SSH Derivation Function								R		

Service	CSPs									
	AES Keys	DH Keys	DRBG Keys	DSA Keys	EC Agreement Key	ECDSA Key	HMAC Keys	KDF Secret Values	RSA Keys	Triple-DES Keys
X9.63 Derivation Function								R		
SP 800-56A-rev2 Concatenation Derivation Function								R		
IKEv2 Derivation Function								R		
SRTP Derivation Function								R		
PBKDF							G,R			
Key Agreement Schemes	G	R			R		R		R	G
Key Wrapping/Transport (RSA, AES, Triple-DES)	R						R		R	R
Key Unwrapping (RSA, AES, Triple-DES)	R						R		R	R
NDRNG Callback			G							
Utility										

4 Self-tests

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

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

Table 14 – Power Up Self-tests

Test Target	Description
Software Integrity	HMAC-SHA256
AES	KATs: Encryption, Decryption Modes: ECB Key sizes: 128 bits
CCM	KATs: Generation, Verification Key sizes: 128 bits
AES-CMAC	KATs: Generation, Verification

Test Target	Description
	Key sizes: AES with 128 bits
FFC KAS	KATs: Per IG 9.6 – Primitive “Z” Computation Parameter Sets/Key sizes: FB
DRBG	KATs: HASH_DRBG, HMAC_DRBG, CTR_DRBG Security Strengths: 256 bits
DSA	KAT: Signature Generation, Signature Verification Key sizes: 2048 bits
ECDSA	KAT: Signature Generation, Signature Verification Curves/Key sizes: P-256
GCM/GMAC	KATs: Generation, Verification Key sizes: 128 bits
HMAC	KATs: Generation, Verification SHA sizes: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256
ECC KAS	KATs: Per IG 9.6 – Primitive “Z” Computation Parameter Sets/Key sizes: FB
RSA	KATs: Signature Generation, Signature Verification Key sizes: 2048 bits
SHS	KATs: Output Verification SHA sizes: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512
Triple-DES	KATs: Encryption, Decryption Modes: TECB, Key sizes: 3-Key
Triple-DES-CMAC	KATs: Generation, Verification Key sizes: 3-Key
Extendable-Output functions (XOF)	KATs: Output Verification XOFs:SHAKE128, SHAKE256
Key Agreement Using RSA	KATs: SP 800-56B specific KATs per IG D.4 Key sizes: 2048 bits
Key Transport Using RSA	KATs: SP 800-56B specific KATs per IG D.4 Key sizes: 2048 bits

Table 15 – Conditional Self-tests

Test Target	Description
NDRNG	NDRNG Continuous Test performed when a random value is requested from the NDRNG.
DH	DH Pairwise Consistency Test performed on every DH key pair generation.
DRBG	DRBG Continuous Test performed when a random value is requested from the DRBG.
DSA	DSA Pairwise Consistency Test performed on every DSA key pair generation.

Test Target	Description
ECDSA	ECDSA Pairwise Consistency Test performed on every EC key pair generation.
RSA	RSA Pairwise Consistency Test performed on every RSA key pair generation.
DRBG Health Checks	Performed conditionally on DRBG, per SP 800-90A Section 11.3. Required per IG C.1.
SP 800-56A Assurances	Performed conditionally per SP 800-56A Sections 5.5.2, 5.6.2, and/or 5.6.3. Required per IG 9.6.

5 Physical Security Policy

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

6 Operational Environment

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

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

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

6.1 Use of External RNG

The module makes use of the JVM's configured SecureRandom entropy source to provide entropy when required. The module will request entropy as appropriate to the security strength and seeding configuration for the DRBG that is using it. In approved mode the minimum amount of entropy that would be requested is 112 bits with a larger minimum being set if the security strength of the operation requires it.. The module will wait until the *SecureRandom.generateSeed()* returns the requested amount of entropy, blocking if necessary.

7 Mitigation of Other Attacks Policy

The Module implements basic protections to mitigate against timing based attacks against its internal implementations. The two counter-measures used are: Constant Time Comparisons, which protect the digest and integrity algorithms, and Numeric Blinding and decryption/signing verification which both protect the RSA algorithm.

The module protects RSA private keys against Lenstra's CRT attack by verifying the correctness of any signature generation performed before returning the signature value.

8 Security Rules and Guidance

8.1 Basic Enforcement

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

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

8.2 Additional Enforcement with a Java SecurityManager

In the presence of a Java SecurityManager approved mode services specific to a context, such as DSA and ECDSA for use in TLS, require specific policy permissions to be configured in the JVM configuration by the Cryptographic Officer or User. The SecurityManager can also be used to restrict the ability of particular code bases to examine CSPs. See section 1.3 Module Configuration for further advice on this.

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

8.3 Enforcement and Guidance for GCM IVs

IVs for GCM can be generated randomly, where an IV is not generated randomly the module supports the importing of GCM IVs.

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

In approved mode, importing a GCM IV is non-conformant unless the source of the IV is also FIPS approved for GCM IV generation.

Per IG A.5, section 2.1 of this security policy also states that in the event module power is lost and restored the consuming application must ensure that any of its AES-GCM keys used for encryption or decryption are re-distributed.

8.4 Enforcement and Guidance for use of the Approved PBKDF

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

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

As the module is a general purpose software module, it is not possible to anticipate all the levels of use for the PBKDF, however a user of the module should also note that a password should at least contain enough entropy to be unguessable and also contain enough entropy to reflect the security strength required for the key being generated. In the event a password encoding is simply based on ASCII a 14 byte password is unlikely to contain sufficient entropy for most purposes. Users are referred to Appendix A, “Security Considerations” of SP 800-132 for further information on password, salt, and iteration count selection.

9 References and Definitions

The following standards are referred to in this Security Policy.

Table 16 – References

Abbreviation	Full Specification Name
ANSI X9.31	*X9.31-1998, Digital Signatures using Reversible Public Key Cryptography for the Financial Services Industry (rDSA), September 9, 1998*
FIPS 140-2	*Security Requirements for Cryptographic modules, May 25, 2001*
FIPS 180-4	*Secure Hash Standard (SHS)*
FIPS 186-3	*Digital Signature Standard (DSS)*
FIPS 186-4	*Digital Signature Standard (DSS)*
FIPS 197	*Advanced Encryption Standard*
FIPS 198-1	*The Keyed-Hash Message Authentication Code (HMAC)*
FIPS 202	*SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*
IG	*Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program*
PKCS#1 v2.1	*RSA Cryptography Standard*
PKCS#5	*Password-Based Cryptography Standard*
PKCS#12	*Personal Information Exchange Syntax Standard*
SP 800-20	*Modes of Operation Validation System for Triple Data Encryption Algorithm (TMOVS)*
SP 800-38A	*Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*
SP 800-38B	*Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*
SP 800-38C	*Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*
SP 800-38D	*Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*
SP 800-38F	*Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping*
SP 800-56A	*Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*
SP 800-56B	*Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography*
SP 800-56C	*Recommendation for Key Derivation through Extraction-then-Expansion*
SP 800-89	*Recommendation for Obtaining Assurances for Digital Signature Applications*
SP 800-90A	*Recommendation for Random Number Generation Using Deterministic Random Bit Generators*
SP 800-108	*Recommendation for Key Derivation Using Pseudorandom Functions*

Abbreviation	Full Specification Name
SP 800-132	*Recommendation for Password-Based Key Derivation*
SP 800-135	*Recommendation for Existing Application –Specific Key Derivation Functions*

Table 17 – Acronyms and Definitions

Acronym	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
BC	Bouncy Castle
BC-FJA	Bouncy Castle FIPS Java API
CBC	Cipher-Block Chaining
CCM	Counter with CBC-MAC
CDH	Computational Diffie-Hellman
CFB	Cipher Feedback Mode
CMAC	Cipher-based Message Authentication Code
CMVP	Crypto Module Validation Program
CO	Cryptographic Officer
CPU	Central Processing Unit
CS	Ciphertext Stealing
CSP	Critical Security Parameter
CTR	Counter-mode
CVL	Component Validation List
DES	Data Encryption Standard
DH	Diffie-Hellman
DRAM	Dynamic Random Access Memory
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Authority
DSTU4145	Ukrainian DSTU-4145-2002 Elliptic Curve Scheme
EC	Elliptic Curve
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Authority
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FIPS	Federal Information Processing Standards
GCM	Galois/Counter Mode
GMAC	Galois Message Authentication Code

Acronym	Definition
GOST	Gosudarstvennyi Standard Soyuza SSR/Government Standard of the Union of Soviet Socialist Republics
GPC	General Purpose Computer
HMAC	key-Hashed Message Authentication Code
IG	See References
JAR	Java ARchive
JDK	Java Development Kit
JRE	Java Runtime Environment
JVM	Java Virtual Machine
IV	Initialization Vector
KAS	Key Agreement Scheme
KAT	Known Answer Test
KDF	Key Derivation Function
KW	Key Wrap
KWP	Key Wrap with Padding
MAC	Message Authentication Code
MD5	Message Digest algorithm MD5
N/A	Non Applicable
NDRNG	Non Deterministic Random Number Generator
OCB	Offset Codebook Mode
OFB	Output Feedback
OS	Operating System
PBKDF	Password-Based Key Derivation Function
PKCS	Public Key Cryptography Standards
PQG	Diffie-Hellman Parameters P, Q and G
RC	Rivest Cipher, Ron’s Code
RIPEMD	RACE Integrity Primitives Evaluation Message Digest
RSA	Rivest Shamir Adleman
SHA	Secure Hash Algorithm
TCBC	TDEA Cipher-Block Chaining
TCFB	TDEA Cipher Feedback Mode
TDEA	Triple Data Encryption Algorithm
TDES	Triple Data Encryption Standard
TECB	TDEA Electronic Codebook
TOFB	TDEA Output Feedback
TLS	Transport Layer Security
USB	Universal Serial Bus
XOF	Extendable-Output Function