VMware, Inc.
3401 Hillview Ave
Palo Alto, CA 94304, USA
Tel: 877-486-9273
Email: info@vmware.com
http://www.vmware.com

VMware's BC-FJA (Bouncy Castle FIPS Java API)

Software Version: 1.0.2

FIPS 140-2 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 0.4

TABLE OF CONTENTS

1 INTRODUCTION 4
1.1 Purpose 4
1.2 Reference 4

2 VMware's BC-FJA (Bouncy Castle FIPS Java API) Module 5
2.1 Introduction 5
2.1.1 VMware's BC-FJA (Bouncy Castle FIPS Java API) 5
2.2 Module Specification 5
2.2.1 Physical Cryptographic Boundary 6
2.2.2 Logical Cryptographic Boundary 7
2.2.3 Modes of Operation 7
2.2.4 Module Configuration 8
2.3 Roles, Authentication and Services 9
2.3.1 Assumption of Roles 9
2.3.2 Services 9
2.4 Physical Security 12
2.5 Operational Environment 12
2.5.1 Use of External RNG 14
2.6 Cryptographic Key Management 14
2.6.1 Critical Security Parameters 20
2.6.2 Public Keys 22
2.7 Self-Tests 22
2.8 Mitigation of Other Attacks Policy 23

3 Secure Operation 25
3.1 Basic Enforcement 25
3.2 Additional Enforcement with a Java SecurityManager 25
3.3 Basic Guidance 25
3.4 Enforcement and Guidance for GCM IVs 26
3.5 Enforcement and Guidance for use of the Approved PBKDF 26
3.6 Rules for setting the N and the S String in cSHAKE 26
3.7 Guidance for the use of DRBGs and Configuring the JVM's Entropy Source 27

4 References and Acronyms 28

LIST OF FIGURES

Figure 1 – Hardware Block Diagram 6
Figure 2 – Module's Logical Cryptographic Boundary 7

LIST OF TABLES

Table 1 – Security Level Per FIPS 140-2 Section 5
Table 2 – FIPS 140-2 Logical Interfaces 7
Table 3 – Available Java Permissions 8
Table 4 – Roles Description 9
Table 5 – Services 9
Table 6 – CSP Access Rights within Services 11
Table 7 – Tested Configuration 12
Table 8 – Approved and CAVP Validated Cryptographic Functions 15
Table 9 – Approved Cryptographic Functions Tested with Vendor Affirmation 17
Table 10 – Non-Approved but Allowed Cryptographic Functions 19
Table 11 – Non-Approved Cryptographic Functions for use in non-FIPS mode only 19
Table 12 – Critical Security Parameters (CSPs) 20
Table 13 – Public Keys 22
Table 14 – Power Up Self-tests 22
Table 15 – Conditional Self-tests 23
Table 16 – References 28
Table 17 – Acronyms 29

1 INTRODUCTION

1.1 Purpose

This is a non-proprietary Cryptographic Module Security Policy for the VMware's BC-FJA (Bouncy Castle FIPS Java API) Module from VMware, Inc. This Security Policy describes how the VMware's BC-FJA (Bouncy Castle FIPS Java API) Module meets the security requirements of Federal Information Processing Standards (FIPS) Publication 140-2, which details the U.S. and Canadian Government requirements for cryptographic modules. More information about the FIPS 140-2 standard and validation program is available on the National Institute of Standards and Technology (NIST) and the Canadian Centre for Cyber Security (CCCS), a branch of the Communications Security Establishment (CSE), Cryptographic Module Validation Program (CMVP) website at https://csrc.nist.gov/projects/cryptographic-module-validation-program.

This document also describes how to run the module in a secure FIPS-Approved mode of operation. The VMware's BC-FJA (Bouncy Castle FIPS Java API) Module is also referred to in this document as “BC-FJA Module”, or “the module”.

1.2 Reference

This document deals only with operations and capabilities of the composite module in the technical terms of a FIPS 140-2 cryptographic module security policy. More information is available on the module from the following sources:

- The VMware website (http://www.vmware.com) contains information on the full line of products from VMware.
- The CMVP website (https://csrc.nist.gov/Projects/Cryptographic-Module-Validation-Program/Validated-Modules/Search) contains options to get contact information for individuals to answer technical or sales-related questions for the module.

2 VMWARE'S BC-FJA (BOUNCY CASTLE FIPS JAVA API) MODULE

2.1 Introduction

VMware, Inc., a global leader in virtualization, cloud infrastructure, and business mobility, delivers customer-proven solutions that accelerate Information Technology (IT) by reducing complexity and enabling more flexible, agile service delivery. With VMware solutions, organizations are creating exceptional experiences by mobilizing everything, responding faster to opportunities with modern data and apps hosted across hybrid clouds, and safeguarding customer trust with a defense-in-depth approach to cybersecurity. VMware enables enterprises to adopt an IT model that addresses their unique business challenges. VMware's approach accelerates the transition to solutional-computing while preserving existing investments and improving security and control.

2.1.1 VMware's BC-FJA (Bouncy Castle FIPS Java API)

The VMware's BC-FJA (Bouncy Castle FIPS Java API) is a software cryptographic module based on the Legion of the Bouncy Castle Inc. FIPS Java API (BC-FJA) Module (SW Version 1.0.2). The module is a software library that provides cryptographic functions to various VMware applications via a well-defined Java-language application program interface (API). The module only performs communications with the calling application (the process that invokes the module services).

The VMware's BC-FJA (Bouncy Castle FIPS Java API) is validated at the FIPS 140-2 Section levels shown in Table 1:

Table 1 – Security Level Per FIPS 140-2 Section

Section	Section Title	Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services, and Authentication	1
4	Finite State Model	1
5	Physical Security	N/A[1]
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC[2]	1
9	Self-tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	1

2.2 Module Specification

The VMware's BC-FJA (Bouncy Castle FIPS Java API) is a software cryptographic module with a multiple-chip standalone embodiment. The overall security level of the module is 1. The software version of the module is 1.0.2, using the 1.0.2 SW version of the Legion of the Bouncy Castle Inc. BC-FJA (Bouncy Castle FIPS Java API) Module.

[1] N/A – Not Applicable

[2] EMI/EMC – Electromagnetic Interference/Electromagnetic Compatibility

2.2.1 Physical Cryptographic Boundary

As a software module, there are no physical protection mechanisms implemented. Therefore, the module must rely on the physical characteristics of the host system. The BC-FJA Module runs on a General-Purpose Computer (GPC) and the physical boundary of the cryptographic module is defined by the hard enclosure around the host system on which it runs. The module supports the physical interfaces of the GPC. See Figure 1 below for a block diagram of the typical GPC and the ports or interfaces across the physical cryptographic boundary marked with red dotted line.

Abbreviations introduced in Figure 1 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.

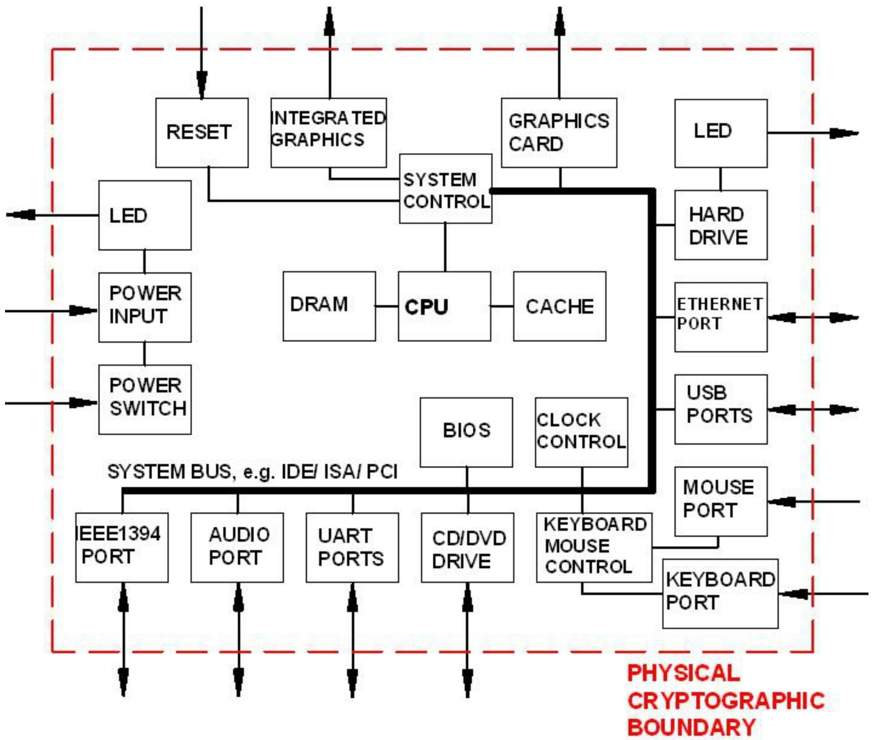

Figure 1 – Hardware Block Diagram

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 2.

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 provide 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.

Table 2 – 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	Startup/Shutdown of a process containing the module.

2.2.2 Logical Cryptographic Boundary

The executable for VMware's BC-FJA (Bouncy Castle FIPS Java API) Module is a Java Archive (JAR) file which is*: bc-fips-1.0.2.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 2 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.

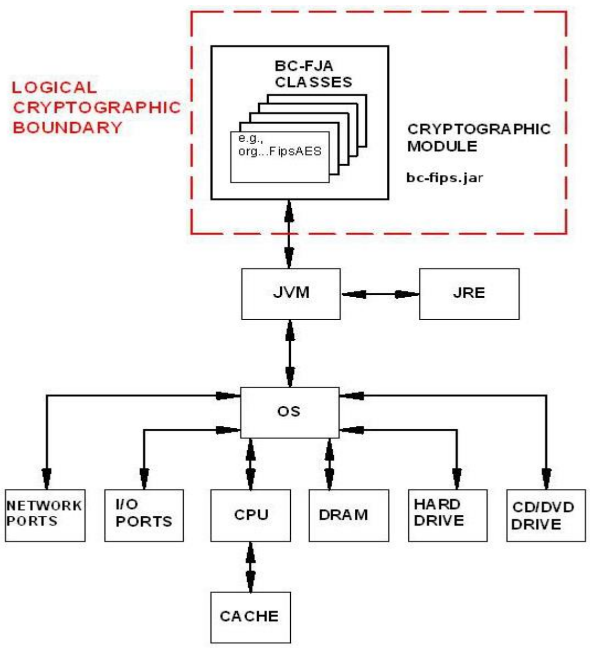

Figure 2 – Module's Logical Cryptographic Boundary

2.2.3 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 transition method is called. To verify that the module is in the Approved Mode of operation, the user can call a FIPS-approved mode status method (*CryptoServicesRegistrar.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.

2.2.4 Module Configuration

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

If the module detects that the system property *org.bouncycastle.fips.approved_only* is set to *true* the module will start in approved mode and non-approved mode functionality will not be available.

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 3.

Table 3 – 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.

2.3 Roles, Authentication and Services

2.3.1 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 4 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 4 – 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

2.3.2 Services

All services implemented by the Module are listed in Table 5 below, and Table 6 describes all usage of CSPs by the service.

Table 5 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 5 – 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 800-56C Concatenation Derivation Function (KDM)	(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 6 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 6 – CSP Access Rights within Services

Service	CSPs																										
	AES Encryption Key	AES Decryption Key	AES Authentication Key	AES Wrapping Key	DH Agreement Key	DRBG (CTR AES)	DRBG (CTR Triple-DES)	DRBG (Hash)	DRBG (HMAC)	DSA Signing Key	EC Agreement Key	EC Signing Key	HMAC Authentication Key	IKEv2 DF Secret	PBKDF Secret	RSA Signing Key	RSA Key Transport Key	SP 800-56A Concat. DF Secret	SP 800-108 KDF Secret	SRTP DF Secret	SSH DF Secret	TLS KDF Secret	Triple-DES Authentication Key	Triple-DES Encryption Key	Triple-DES Decryption Key	Triple-DES Wrapping Key	X9.63 KDF Secret
Initialize Module and Run Self-Tests on Demand																											
Show Status																											
Zeroize / Power-off	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Data Encryption	R																							R			
Data Decryption		R																							R		
MAC Calculation			R										R													R	
Signature Generation										R		R				R											
Signature Verification										R		R				R											
DRBG (SP800-90A) output	G	G	G	G	G	G R	G R	G R	G R	G	G	G	G			G	G						G	G	G	G	
Message Hashing																											
Keyed Message Hashing													R														
TLS Key Derivation Function																						R					
SP 800-108 KBKDF																			R								
SSH Derivation Function																					R						
X9.63 Derivation Function					G						G					G											R
SP 800-56C Concatenation Derivation Function (KDM)					G						G					G		R									
IKEv2 Derivation Function														R													
SRTP Derivation Function																				R							
PBKDF													G R		R												
Key Agreement Schemes	G	G	G	G	R						R		G				R						G	G	G	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	G	G	G																		
Utility																											

2.4 Physical Security

The VMware's BC-FJA (Bouncy Castle FIPS Java API) is a software module, which FIPS defines as a multi-chip standalone cryptographic module. As such, it does not include physical security mechanisms. Thus, the FIPS 140-2 requirements for physical security are not applicable.

2.5 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.

The cryptographic module was tested on the following operational environment on the general-purpose computer (GPC) platforms detailed in Table 7 below:

Table 7 – Tested Configuration

#	GPC Platforms	Operational Environment (on ESXi 7.0)	Java SE Runtime Environment	Processor Family
1	Dell PowerEdge R740 with Intel Xeon Gold Processor	Photon OS 2.0	Java SE Runtime Environment v8 (1.8.0), single-user mode	Intel Xeon Gold 6126 Series
2		Photon OS 2.0	Java SE Runtime Environment v11 (1.11.0), single-user mode	Intel Xeon Gold 6126 Series
3		Photon OS 3.0	Java SE Runtime Environment v8 (1.8.0), single-user mode	Intel Xeon Gold 6126 Series
4		Photon OS 3.0	Java SE Runtime Environment v11 (1.11.0), single-user mode	Intel Xeon Gold 6126 Series
5		Ubuntu OS 16.04	Java SE Runtime Environment v8 (1.8.0), single-user mode	Intel Xeon Gold 6126 Series
6		Ubuntu OS 16.04	Java SE Runtime Environment v11 (1.11.0), single-user mode	Intel Xeon Gold 6126 Series
7		Ubuntu 18.04	Java SE Runtime Environment v8 (1.8.0), single-user mode	Intel Xeon Gold 6126 Series

#	GPC Platforms	Operational Environment	Java SE Runtime Environment	Processor Family
8		Ubuntu 18.04	Java SE Runtime Environment v11 (1.11.0), single-user mode	Intel Xeon Gold 6126 Series
9		Windows Server 2019	Java SE Runtime Environment v11 (1.11.0), single-user mode	Intel Xeon Gold 6126 Series
10		Windows Server 2016	Java SE Runtime Environment v8 (1.8.0), single-user mode	Intel Xeon Gold 6126 Series
11		Windows Server 2016	Java SE Runtime Environment v11 (1.11.0), single-user mode	Intel Xeon Gold 6126 Series
12		CentOS 8	Java SE Runtime Environment v11 (1.11.0), single-user mode	Intel Xeon Gold 6126 Series
13		SUSE Linux Enterprise Server 15	Java SE Runtime Environment v11 (1.11.0), single-user mode	Intel Xeon Gold 6126 Series
#	**GPC Platforms**	**Operational Environment**	**Java SE Runtime Environment**	**Processor Family**
14	Dell Latitude E7470 with Intel Core i5	Ubuntu 18.04	Java SE Runtime Environment v11 (1.11.0), single-user mode	Core i5 Processor
15		Windows 10	Java SE Runtime Environment v11 (1.11.0), single-user mode	Core i5 Processor

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
   - VMware ESXi
   - VMware Photon OS

BLUX

Further, VMware, Inc. affirms that the VMware's BC-FJA (Bouncy Castle FIPS Java API) Module runs in its configured, Approved mode of operation on the following binary compatible platforms executing VMware ESXi 6.0, ESXi 6.5, ESXi 6.7, or ESXi 7.0 and Java SE Runtime Environment 7, 8, or 11 with any of the above listed OS:

- Dell PowerEdge R740 with Intel Xeon Processor
- Dell PowerEdge T430 with Intel Xeon Processor
- Dell PowerEdge T320 with Intel Xeon Processor
- Dell PowerEdge R340 with Intel Xeon Processor
- Dell PowerEdge R530 with Intel Xeon Processor
- Dell PowerEdge R730 with Intel Xeon Processor
- Dell PowerEdge R830 with Intel Xeon Processor
- Dell PowerEdge FC640 with Intel Xeon Processor
- HPE ProLiant DL380 Gen9 with Intel Xeon Processor
- HPE ProLiant DL38P Gen8 with AMD Opteron Processor
- Cisco UCS – B22 M Series Blade Servers with Intel Processor
- Cisco UCS – C24 M3 Series Rackmount with Intel Xeon Processor
- A general-purpose computer platform with an Intel Core i/Intel Xeon/AMD Opteron Processor executing VMware ESXi and any OS (including Apple OS (OS X, macOS, iOS), Android OS, and others) with single user mode.
- A general-purpose computer platform with Intel Core i/Intel Xeon/AMD Opteron Processor executing any OS (including Apple OS (OS X, macOS, iOS), Android OS, and others) in single user mode with JDK 7, 8, or 11.
- A cloud computing environment composed of a general-purpose computing platform executing VMware ESXi or a VMware cloud solution that is executing VMware ESXi.

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-1.0.2.jar*.

2.5.1 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 and for the default DRBG will request a minimum of 256 bits of entropy. In approved mode the minimum amount of entropy that can be requested by a DRBG is 112 bits. The module will wait until the *SecureRandom.generateSeed()* returns the requested amount of entropy, blocking if necessary.

2.6 Cryptographic Key Management

The Module implements the FIPS Approved and Non-Approved but Allowed cryptographic functions listed in Table 8 to Table 10 below.

Table 8 – 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	C1742
CCM	[SP 800-38C] Functions: Generation, Authentication Key sizes: 128, 192, 256 bits	C1742
CMAC	[SP 800-38B] Functions: Generation, Authentication Key sizes: AES with 128, 192, 256 bits and Triple-DES with 2-key[3,4], 3-key	C1742
GCM/GMAC[5]	[SP 800-38D] Functions: Generation, Authentication Key sizes: 128, 192, 256 bits	C1742
DRBG	[SP 800-90A] Functions: Hash DRBG, HMAC DRBG, AES-CTR DRBG, Triple-DES-CTR DRBG. Security Strengths: 112, 128, 192, and 256 bits	C1742
DSA[6]	[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)	C1742
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	C1742

[3] 2^20 block limit is enforced by module

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

[5] GCM encryption with an internally generated IV, see section 3.4 concerning external IVs. IV generation is compliant with IG A.5.

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

Algorithm	Description	Cert #
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, SHA3-224, SHA3-256, SHA3-384, SHA3-512	C1742
KAS[7]	[SP 800-56A-rev1] Parameter sets/Key sizes: FB, FC, EB, EC, ED, EE	C1742
KAS Component	[SP 800-56A-rev1] Function: ECC-CDH Primitive Parameter sets/Key sizes: EB, EC, ED, EE	C1742
KDF, Existing Application-Specific[8]	[SP 800-135] Functions: TLS v1.0/1.1 KDF, TLS 1.2 KDF, SSH KDF, X9.63 KDF, IKEv2 KDF, SRTP KDF.	C1742
KBKDF, using Pseudorandom Functions[9]	[SP 800-108] Modes: Counter Mode, Feedback Mode, Double-Pipeline Iteration Mode Functions: CMAC-based KBKDF with AES, 3-key Triple-DES or HMAC-based KBKDF with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	C1742
Key Wrapping Using Block Ciphers[10]	[SP 800-38F] Modes: AES KW, KWP Key sizes: 128, 192, 256 bits (provides between 128 and 256 bits of strength)	C1742
	[SP 800-38F] Mode: Triple-DES TKW Key size: 3-key (provides 112 bits of strength)	C1742
RSA	[FIPS 186-4, FIPS 186-2, ANSI X9.31-1998 and PKCS #1 v2.1 (PSS and PKCS1.5)] Functions: Key Pair Generation (2048 and 3072 bits) Signature Generation, Signature Verification, Component Test Key sizes: 2048, 3072 bits (1024, 1536, 4096 only for SigVer) SP 800-56B Section 7.1.2 RSA Decryption Primitive	C1742

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

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

[9] 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.

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

Algorithm	Description	Cert #
SHS	[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	C1742
SHA-3, SHAKE	[FIPS 202] SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256	C1742
Triple-DES (Triple-DES)	[SP 800-67] Functions: Encryption, Decryption Modes: TECB, TCBC, TCFB64, TCFB8, TOFB, CTR Key sizes: 2-key (Decryption only)[11], 3-key[12]	C1742

Table 9 – 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.12
CKG using output from DRBG13	[SP 800-133] Section 6.1 (Asymmetric from DRBG) Section 7.1 (Symmetric from DRBG) Using C1742 (DRBG)	Vendor Affirmed IG D.12
cSHAKE128, cSHAKE256	[SP 800-185] Section 3, cSHAKE Using C1742 (SHA3, SHAKE)	Vendor Affirmed IG A.15

[11] 2^20 block limit is enforced by the module, 2-key encryption is disabled.

[12] 3-key Triple-DES encryption must not be used for more than 2^20 blocks for any given key.

[13] The resulting key or a generated seed is an unmodified output from a DRBG

Algorithm	Description	IG Ref.
KAS[14]	[SP 800-56A-rev2] Section 5.6.2.3.1 (Finite Field Cryptography (FFC) Full Public Key Validation Routine) Section 5.6.2.3.2 (Elliptic Curve Cryptography (ECC) Full Public Key Validation Routine) Section 5.7 (DLC Primitive) Section 5.8 (Key Derivation Functions for Key Agreement Schemes) Section 5.9 (Key Confirmation) Section 6 (Key Agreement) Parameter sets/Key sizes: FB, FC, EB, EC, ED, EE Using C1742 (CVL)	Vendor Affirmed IG D.1 rev 2
KAS using SHA-512/224 or SHA-512/256	[SP 800-56A-rev2] Parameter sets/Key sizes: FB, FC, EB, EC, ED, EE[15] Using C1742 (CVL)	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 Using C1742 (HMAC)	Vendor Affirmed IG D.6
Key Wrapping[16] Using RSA	[SP 800-56B, Section 7.2.3] RSA-KEMS-KWS with, and without, key confirmation. Key sizes: 2048, 3072 bits	Vendor Affirmed IG D.4
Key Transport[16] Using RSA	[SP 800-56B, Section 7.2.2] RSA-OAEP with, and without, key confirmation. Key sizes: 2048, 3072 bits	Vendor Affirmed IG D.4
RSA	[SP 800-131 rev2] Section 3 Key sizes: 4096 up to 16384 bits Using mechanism tested in C1742	Vendor Affirmed IG A.14

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

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

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

Table 10 – Non-Approved but Allowed Cryptographic Functions

Algorithm	Description
NDRNG	[IG 7.15] Non-deterministic random number generator.
Non-SP 800-56A-rev2 Compliant DH	[IG D.8] Diffie-Hellman 2048 and 3072 bits, bit key agreement primitive for use with system-level key establishment; not used by the module to establish keys within the module (Cert C1742 (CVL), key agreement; key establishment methodology provides 112 bits or 128 bits of encryption strength)
Non-SP 800-56A-rev2 Compliant EC DH	[IG D.8] EC Diffie-Hellman key agreement primitive for use with system-level key establishment; not used by the module to establish keys within the module (Cert C1742 (CVL), key agreement; key establishment methodology provides between 112 bits and 256 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: 4096 up to 16384 bits.
MD5 within TLS	[IG D.2]

Table 11 – Non-Approved Cryptographic Functions for use in non-FIPS mode only

AES (non-compliant[17]) ARC4 (RC4) Blowfish Camellia CAST5 DES Diffie-Hellman KAS (non-compliant[18]) DSA (non-compliant[19]) DSTU4145 ECDSA (non-compliant[20]) EdDSA ElGamal GOST28147	KAS[21] using SHA-512/224 or SHA-512/256 KBKDF using SHA-512/224 or SHA-512/256 (non-compliant) MD5 OpenSSL PBKDF (non-compliant) PKCS#12 PBKDF (non-compliant) PKCS#5 Scheme 1 PBKDF (non-compliant) PRNG X9.31 RC2 RIPEMD128 RIPEMD160 RIPEMD256 RIPEMD320

[17] Support for additional modes of operation.

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

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

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

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

GOST3410-1994	RSA (non-compliant[22])
GOST3410-2001	RSA KTS (non-compliant[23])
GOST3411	SCrypt
HMAC-GOST3411	SEED
HMAC-MD5	Serpent
HMAC-RIPEMD128	SipHash
HMAC-RIPEMD160	SHACAL-2
HMAC-RIPEMD256	TIGER
HMAC-RIPEMD320	Triple-DES (non-compliant[24])
HMAC-TIGER	Twofish
HMAC-WHIRLPOOL	WHIRLPOOL
IDEA	XDH

2.6.1 Critical Security Parameters

All CSPs used by the Module are described in this section in Table 12**.** All usage of these CSPs by the Module (including all CSP lifecycle states) is described in the services detailed in section 2.3.2.

Table 12 – Critical Security Parameters (CSPs)

CSP	Description / Usage
AES Encryption Key	[FIPS-197, SP 800-38A, SP 800-38C, SP 800-38D, Addendum to SP 800-38A] AES (128/192/256) encrypt key[25]
AES Decryption Key	[FIPS-197, SP 800-38A, SP 800-38C, 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 (160 - 512 bits) private key agreement key
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)

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

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

[24] Support for additional modes of operation.

[25] 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(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 (P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409 and B-571) private key agreement key
EC Signing Key	[FIPS 186-4] ECDSA (P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409 and B-571) 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 up to 16384 bits) signature generation key
RSA Key Transport Key	[SP 800-56B] RSA (2048 up to 16384 bits) key transport (decryption) key
SP 800-56C Concatenation Derivation Function Secret Value	[SP 800-56C] 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
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.

2.6.2 Public Keys

Table 13 – Public Keys

CSP	Description / Usage
DH Agreement Key	[SP 800-56A-rev2] Diffie-Hellman (2048 and 3072) 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 (P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409 and B-571) public key agreement key
EC Verification Key	[FIPS 186-4] ECDSA (P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409 and B-571) signature verification key
RSA Key Transport Key	[SP 800-56B] RSA (2048-16384) key transport (encryption) key.
RSA Verification Key	[FIPS 186-4] RSA (1024-16384) signature verification key

2.7 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 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-256, SHA-512, SHA3-256

Test Target	Description
ECC KAS	KATs: Per IG 9.6 – Primitive "Z" Computation Parameter Sets/Key sizes: EC
SP 800-108 KBKDF	KATs: Per IG 9.4 – Output Verification Modes: Counter, Feedback, Double Pipeline PRFs: AES-CMAC, Triple-DES-CMAC, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256
RSA	KATs: Signature Generation, Signature Verification Key sizes: 2048 bits
SHS	KATs: Output Verification SHA sizes: SHA-1, SHA-256, SHA-512
Triple-DES	KATs: Encryption, Decryption Mode: TECB Key sizes: 3-Key
Triple-DES-CMAC	KATs: Generation, Verification Key sizes: 3-Key
Extendable-Output functions (XOF)	KATs: Output Verification XOFs: SHAKE256
Key Wrapping 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.
ECDH/ECCDH	EC DH Pairwise Consistency Test performed on every DH key pair generation.
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.
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.

2.8 Mitigation of Other Attacks Policy

The Module implements basic protections to mitigate against timing-based attacks against its internal implementations. There are two countermeasures used.

The first is Constant Time Comparisons, which protect the digest and integrity algorithms by strictly avoiding "fast fail" comparison of MACs, signatures, and digests so the time taken to compare a MAC, signature, or digest is constant regardless of whether the comparison passes or fails.

The second is made up of Numeric Blinding and decryption/signing verification which both protect the RSA algorithm.

Numeric Blinding prevents timing attacks against RSA decryption and signing by providing a random input into the operation which is subsequently eliminated when the result is produced. The random input makes it impossible for a third party observing the private key operation to attempt a timing attack on the operation as they do not have knowledge of the random input and consequently the time taken for the operation tells them nothing about the private value of the RSA key.

Decryption/signing verification is carried out by calculating a primitive encryption or signature verification operation after a corresponding decryption or signing operation before the result of the decryption or signing operation is returned. The purpose of this is to protect against Lenstra's CRT attack by verifying the correctness the private key calculations involved. Lenstra's CRT attack takes advantage of undetected errors in the use of RSA private keys with CRT values and, if exploitable, can be used to discover the private value of the RSA key.

3 Secure Operation

The VMware's BC-FJA (Bouncy Castle FIPS Java API) Module meets Level 1 requirements for FIPS 140-2. The sections below describe how to install, use, and keep the module in FIPS-Approved mode of operation.

3.1 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.

3.2 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 2.2.4 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.

3.3 Basic Guidance

The jar file representing the module needs to be installed in a JVM's class path in a manner appropriate to its use in applications running on the JVM.

Functionality in the module is provided in two ways. At the lowest level there are distinct classes that provide access to the FIPS approved and non-FIPS approved services provided by the module. A more abstract level of access can also be gained through the use of strings providing operation names passed into the module's Java cryptography provider through the APIs described in the Java Cryptography Architecture (JCA) and the Java Cryptography Extension (JCE).

When the module is being used in FIPS approved-only mode, classes providing implementations of algorithms which are not FIPS approved, or allowed, are explicitly disabled.

3.4 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 DRBG in line with Section 8.2.2 of SP 800-38D.

In approved mode, when a GCM IV is generated using the FipsNonceGenerator a counter is used as the basis for the nonce. Rollover of the counter in the FipsNonceGenerator will result in an IllegalStateException indicating the FipsNonceGenerator is exhausted and, as per IG A.5, where used for TLS, rollover will terminate any TLS session in process using the current key and the exception can only be recovered from by using a new handshake and creating a new FipsNonceGenerator.

In approved mode, importing a GCM IV for encryption that originates from outside the module is non-conformant.

Per IG A.5, section 2.6.1 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.

3.5 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.

For users interested in introducing memory hardness as a layer on top of the PBKDF the scrypt augmentation to PBDKF based on HMAC SHA-256 (as described in RFC 7914) is also available.

3.6 Rules for setting the N and the S String in cSHAKE

The cSHAKE algorithm offers to input string for customizing the output of the cSHAKE function, the Function-Name input (N) and the Customization String (S).

The Function-Name input (N) is reserved for values specified by NIST and should only be set to the appropriate NIST specified value. Any other use of N is non-conformant.

The Customization String (S) is available to allow users to customize the cSHAKE function as they wish. The length of S is limited to the available size of a byte array in the JVM running the module.

3.7 Guidance for the use of DRBGs and Configuring the JVM's Entropy Source

A user can instantiate the default Approved DRBG for the module explicitly by using SecureRandom.getInstance("DEFAULT", "BCFIPS"), or by using a BouncyCastleFipsProvider object instead of the provider name as appropriate. This will seed the Approved DRBG from the live entropy source of the JVM, for example /dev/random on the tested Linux operational environments, with a number of bits of entropy appropriate to the security level of the default Approved DRBG configured for the module.

An additional option is available using the Approved Hash_DRBG and the process outlined in SP-800 90A, Section 8.6.5. This can be turned on by following the instructions in Section 2.3 of the User Guide. The two DRBGs are instantiated in a chain as a "Source DRBG" to seed the "Target DRBG" in accordance with Section 7 of Draft NIST SP 800-90C, where the Target DRBG is the default Approved DRBG used by the module.

The initial seed and the subsequent reseeds for the DRBG chain come from the live entropy source configured for the JVM. The DRBG chain will reseed automatically by pausing for 20 requests (which will usually equate to 5120 bytes). An entropy gathering thread reseeds the DRBG chain when it has gathered sufficient entropy (currently 256 bits) from the live entropy source. Once reseeded, the request counter is reset and the reseed process begins again.

The “Source DRBG” in the chain is internal to the module and inaccessible to the user to ensure it is only used for generating seeds for the default Approved DRBG of the module.

The user shall ensure that the Approved entropy source is configured per Section 2.5.1 of this Security Policy and will block, or fail, if it is unable to provide the amount of entropy requested.

4 REFERENCES AND ACRONYMS

The standards in Table 16 are referred to in this Security Policy. Table 17 provides definitions for the acronyms used in this document.

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 Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher
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 800-67	Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher
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
SP 800-132	Recommendation for Password-Based Key Derivation
SP 800-133	Recommendation for Cryptographic Key Generation
SP 800-135	Recommendation for Existing Application –Specific Key Derivation Functions

Table 17 – Acronyms

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
EdDSA	Edwards Curve DSA using Ed25519, Ed448
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FIPS	Federal Information Processing Standards
GCM	Galois/Counter Mode
GMAC	Galois Message Authentication Code
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
JCA	Java Cryptography Architecture
JCE	Java Cryptography Extension
JDK	Java Development Kit
JRE	Java Runtime Environment

Acronym	Definition
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	Not 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, and 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
XDH	Edwards Curve Diffie-Hellman using X25519, X448
XOF	Extendable-Output Function

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com