FIPS 140-2 Non-Proprietary Security Policy

CTERA Crypto Module™ (Java)

Software Version 3.0.1

Document Version 1.1

November 16, 2021

Prepared For:

CTERA Networks Ltd.
CTERA Networks NA HQ
205 E. 42nd Street,
New York, NY, 10017
www.ctera.com

Prepared By:

SafeLogic Inc.
530 Lytton Ave, Suite 200
Palo Alto, CA 94301
www.safelogic.com

Overview

This document provides a non-proprietary FIPS 140-2 Security Policy for the CTERA Crypto Module™ (Java).

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 About FIPS 140

Federal Information Processing Standards Publication 140-2 — Security Requirements for Cryptographic Modules specifies requirements for cryptographic modules to be deployed in a Sensitive but Unclassified environment. The National Institute of Standards and Technology (NIST) and Canadian Centre for Cyber Security (CCCS) Cryptographic Module Validation Program (CMVP) run the FIPS 140 program. The NVLAP accredits independent testing labs to perform FIPS 140 testing; the CMVP validates modules meeting FIPS 140 validation. *Validated* is the term given to a module that is documented and tested against the FIPS 140 criteria.

More information is available on the CMVP website at http://csrc.nist.gov/groups/STM/cmvp/index.html.

1.2 About this Document

This non-proprietary Cryptographic Module Security Policy for CTERA Crypto Module™ (Java) from CTERA Networks Ltd. (“CTERA”) provides an overview of the product and a high-level description of how it meets the overall Level 1 security requirements of FIPS 140-2.

The CTERA Crypto Module™ (Java) may also be referred to as the “module” in this document.

1.3 External Resources

The CTERA website (https://www.ctera.com/) contains information on CTERA services and products. The Cryptographic Module Validation Program website contains links to the FIPS 140-2 certificate and CTERA contact information.

1.4 Notices

This document may be freely reproduced and distributed in its entirety without modification.

2 CTERA Crypto Module™ (Java)

2.1 Cryptographic Module Specification

The CTERA Crypto Module™ (Java) is the FIPS validated cryptographic provider for the CTERA Portal.

The module's logical cryptographic boundary is the Java Archive (JAR) file (ccj-3.0.1.jar). The module is a multi-chip standalone embodiment installed on a General Purpose Device. The module is a software module and relies on the physical characteristics of the host platform. The module's physical cryptographic boundary is defined by the enclosure of the host platform.

All operations of the module occur via calls from host applications and their respective internal daemons/processes. As such there are no untrusted services calling the services of the module.

2.1.1 Validation Level Detail

The following table lists the module's level of validation for each area in FIPS 140-2:

FIPS 140-2 Section Title	Validation Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
Electromagnetic Interference / Electromagnetic Compatibility	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	1

Table 1 – Validation Level by FIPS 140-2 Section

2.1.2 Modes of Operation

The module supports 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 a 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.

2.1.3 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 *com.safelogic.cryptocomply.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 2 – 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

Permission	Settings	Req	Usage
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

Table 2 – Available Java Permissions

2.1.4 Approved Cryptographic Algorithms

The module's cryptographic algorithm implementations have received the following certificate numbers from the Cryptographic Algorithm Validation Program.

CAVP Cert.	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli	Use
4702	**AES**	FIPS 197 SP 800-38A	ECB, CBC, OFB, CFB8, CFB128, CTR	128, 192, 256	Encryption, Decryption
Based on 4702	**AES-CBC Ciphertext Stealing (CS)**	Addendum to SP 800-38A, Oct 2010	CBC-CS1, CBC-CS2, CBC-CS3	128, 192, 256	Encryption, Decryption
4702	**CCM**	SP 800-38C	AES	128, 192, 256	Generation, Authentication
4702 (AES) 2494 (Triple-DES)	**CMAC**	SP 800-38B	AES, Triple-DES	AES with 128, 192, 256 Triple-DES with 2-key[1], 3-key	Generation, Authentication
4702	**GCM/GMAC[2]**	SP 800-38D	AES	128, 192, 256	Generation, Authentication
1600	**DRBG**	SP 800-90A	Hash DRBG, HMAC DRBG, CTR DRBG	112, 128, 192, 256	Random Bit Generation

[1] 2^{20} block limit is enforced by module for legacy operations only
[2] GCM with an internally generated IV, see section 3.1.3 concerning external IVs. IV generation is compliant with IG A.5.

CAVP Cert.	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli	Use
1244	**DSA**[3]	FIPS 186-4	PQG Generation, PQG Verification, Key Pair Generation, Signature Generation, Signature Verification	1024, 2048, 3072 bits (1024 only for SigVer)	Digital Signature Services
1160 1343 (CVL)	**ECDSA**	FIPS 186-4	Signature Generation Component, Key Pair Generation, Signature Generation, Signature Verification, Public Key Validation	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	Digital Signature Services
3114	**HMAC**	FIPS 198-1	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256	Various (KS<BS, KS=BS, KS>BS)	Generation, Authentication

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

CAVP Cert.	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli	Use
130	**KAS**[4]	SP 800-56A	KAS-FFC, KAS-ECC	FB (L=2048, N=224), FC (L=2048, N=256), EB (P-224), EC (P-256), ED (P-384), EE (P-521)	Key Agreement
1344 (CVL)	**KAS Component**	SP 800-56A	ECC-CDH Primitive	P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571	Key Agreement Primitive
1342 (CVL)	**KDF, Existing Application-Specific**[5]	SP 800-135	TLS v1.0/1.1 KDF, TLS 1.2 KDF, SSH KDF, X9.63 KDF, IKEv2 KDF, SRTP KDF	Various (See CVL #1342 for details)	KDF Services
145	**KBKDF, using Pseudorandom Functions**[6]	SP 800-108	Counter Mode, Feedback Mode, Double-Pipeline Iteration Mode	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	KDF Services

[4] Keys are not established directly into the module using the key agreement algorithms
[5] These protocols have not been reviewed or tested by the CAVP and CMVP.
[6] 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.

CAVP Cert.	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli	Use
4702 (AES) 2494 (Triple-DES)	**Key Wrapping Using Block Ciphers (KTS)**[7]	SP 800-38F	KW, KWP, TKW	AES-128, AES-192, AES-256, 3-key Triple-DES	Key Transport For AES, the key establishment methodology provides between 128 and 256 bits of encryption strength For Triple-DES, key establishment methodology provides 112 bits of encryption strength
2562 1345 (CVL)	**RSA**	FIPS 186-4 FIPS 186-2	Padding from: ANSI X9.31-1998 PKCS #1 v2.1 (PSS and PKCS1.5)	1024, 1536, 2048, 3072, 4096 bits (1024, 1536, 4096 only for SigVer)	Key Pair Generation, Signature Generation, Signature Verification, Component Test
3849	**SHS**	FIPS 180-4	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256	N/A	Digital Signature Generation, Digital Signature Verification, non-Digital Signature Applications

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

CAVP Cert.	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli	Use
24	**SHA-3, SHAKE**	FIPS 202	SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256	N/A	Digital Signature Generation, Digital Signature Verification, non-Digital Signature Applications
2494	**Triple-DES**	SP 800-67	TECB, TCBC, TCFB64, TCFB8, TOFB, CTR	2-key[8], 3-key[9]	Encryption, Decryption

Table 3 – FIPS-Approved Algorithm Certificates

[8] 2^{20} block limit is enforced by the module, encryption is disabled.
[9] 3-key Triple-DES encryption must not be used for more than 2^{32} blocks for any given key.

The following Approved cryptographic algorithms were implemented with vendor affirmation.

Algorithm	IG Reference	Use
CKG using output from DRBG	Vendor Affirmed IG D.12	[SP 800-133] Section 6.1 (Asymmetric from DRBG) Section 7.1 (Symmetric from DRBG) Using DRBG #1600
KAS[10] using SHA-512/224 or SHA-512/256	Vendor Affirmed IG A.3, D.1-rev2	[SP 800-56A-rev2] Parameter sets/Key sizes: FB, FC, EB, EC, ED, EE[11] Using CVL #1344
KDF, Password-Based	Vendor Affirmed IG D.6	[SP 800-132] Options: PBKDF with Option 1a Functions: HMAC-based KDF using SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 Using HMAC #3114
Key Wrapping Using RSA	Vendor Affirmed IG D.4	[SP 800-56B] RSA-KEMS-KWS with, and without, key confirmation Key sizes: 2048, 3072 bits
Key Transport Using RSA	Vendor Affirmed IG D.4	[SP 800-56B] RSA-OAEP with, and without, key confirmation Key sizes: 2048, 3072 bits

Table 4 – Approved Cryptographic Functions Tested with Vendor Affirmation

[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.

2.1.5 Non-Approved but Allowed Cryptographic Algorithms

The module supports the following non-FIPS 140-2 approved but allowed algorithms that may be used in the Approved mode of operation.

Algorithm	Use
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 (key wrapping; key establishment methodology provides 112 or 128 bits of encryption strength)
MD5 within TLS	[IG D.2, IG 1.23 example 2a]

Table 5 – Non-Approved but Allowed Cryptographic Algorithms

2.1.6 Non-Approved Mode of Operation

The module supports a non-approved mode of operation. The algorithms listed in this section are not to be used by the operator in the FIPS Approved mode of operation.

Algorithm	Use
AES (non-compliant[12])	Encryption, Decryption
ARC4 (RC4)	Encryption, Decryption
Blowfish	Encryption, Decryption
Camellia	Encryption, Decryption
CAST5	Encryption, Decryption
DES	Encryption, Decryption
DSA (non-compliant[13])	Public Key Cryptography
DSTU4145	Public Key Cryptography
ECDSA (non-compliant[14])	Public Key Cryptography

[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.

Algorithm	Use
ElGamal	Public Key Cryptography
GOST28147	Encryption, Decryption
GOST3410-1994	Hashing
GOST3410-2001	Hashing
GOST3411	Hashing
HMAC-GOST3411	Hashing
IDEA	Encryption, Decryption
KAS (non-compliant[15])	Key Agreement
KBKDF using SHA-512/224 or SHA-512/256 (non-compliant)	KDF
MD5	Hashing
HMAC-MD5	Hashing
OpenSSL PBKDF	KDF
PKCS#12 PBKDF	KDF
PKCS#5 Scheme 1 PBKDF	KDF
RC2	Encryption, Decryption
RIPEMD128	Hashing
HMAC-RIPEMD128	Hashing
RIPEMD-160	Hashing
HMAC-RIPEMD160	Hashing
RIPEMD256	Hashing
HMAC-RIPEMD256	Hashing
RIPEMD320	Hashing
HMAC-RIPEMD320	Hashing
X9.31 PRNG	Random Number Generation
RSA (non-compliant[16])	Public Key Cryptography
RSA KTS (non-compliant[17])	Public Key Cryptography
SCrypt	KDF
SEED	Encryption, Decryption
Serpent	Encryption, Decryption
SipHash	Hashing
SHACAL-2	Encryption, Decryption
TIGER	Hashing
HMAC-TIGER	Hashing
TripleDES (non-compliant[18])	Encryption, Decryption

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

Algorithm	Use
Twofish	Encryption, Decryption
WHIRLPOOL	Hashing
HMAC-WHIRLPOOL	Hashing

Table 6 – Non-Approved Cryptographic Functions for use in non-FIPS mode only

2.2 Critical Security Parameters and Public Keys

2.2.1 Critical Security Parameters

The table below provides a complete list of Critical Security Parameters used within the module:

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[19]
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
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

[19] The AES-GCM key and IV are 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
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 (112/192) 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

Table 7 – Critical Security Parameters

2.2.2 Public Keys

The table below provides a complete list of the public keys used within the module:

Public Key	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

Table 8 – Public Keys

2.3 Module Interfaces

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

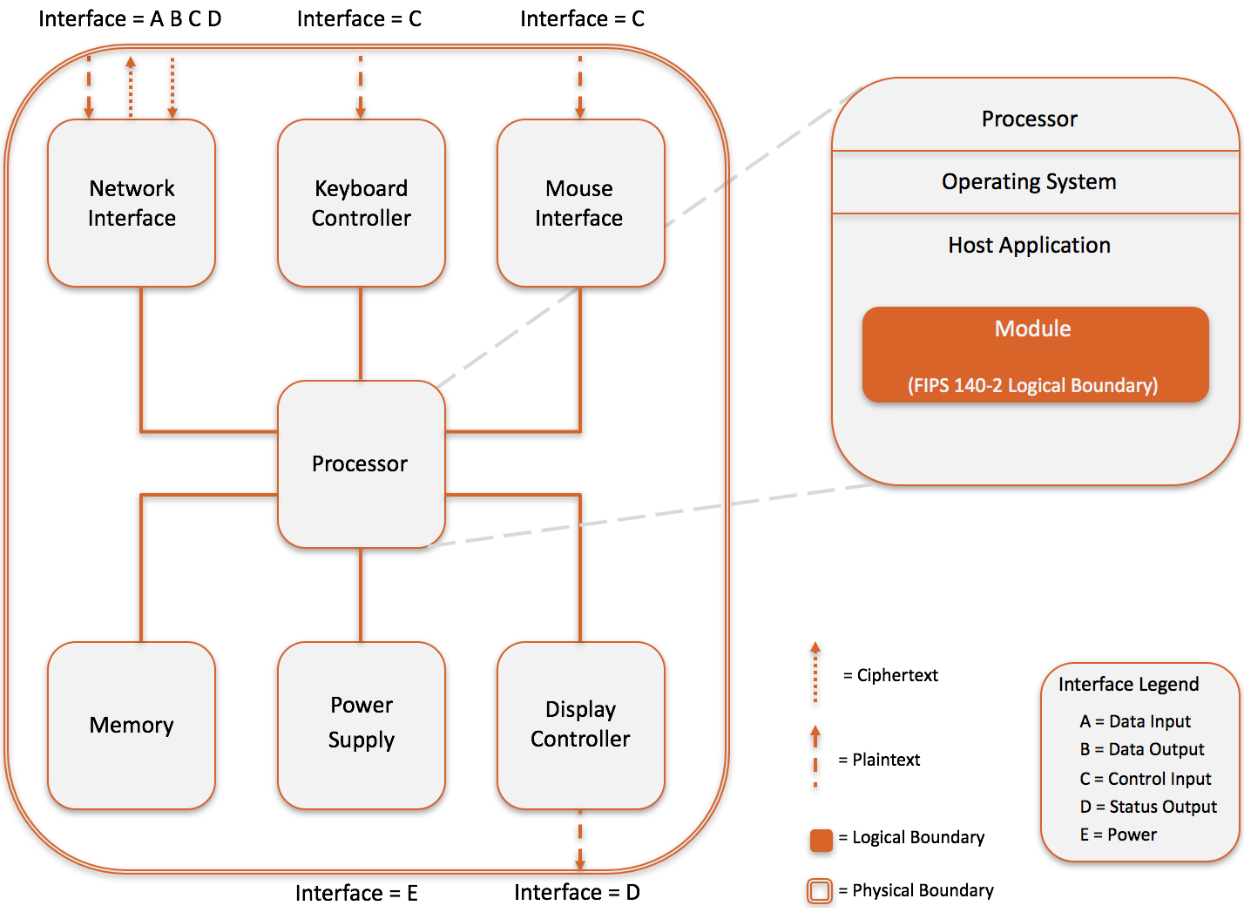

Figure 1 – Module Boundary and Interfaces Diagram

The interfaces (ports) for the physical boundary are shown above. When operational, the module does not transmit any information across these physical ports because it is a software cryptographic module. Therefore, the module's interfaces are purely logical and are provided through the Application Programming Interface (API) that a calling daemon can operate. The logical interfaces expose services that applications directly call, and the API provides functions that may be called by a referencing application (see Section 2.4 – Roles, Services, and Authentication for the list of available functions). The module distinguishes between logical interfaces by logically separating the information according to the defined API.

The API provided by the module is mapped onto the FIPS 140- 2 logical interfaces: data input, data output, control input, and status output. Each of the FIPS 140- 2 logical interfaces relates to the module's callable interface, as follows:

FIPS 140-2 Interface	Logical Interface	Module Physical Interface
Data Input	API input parameters – plaintext and/or ciphertext data	Network Interface
Data Output	API output parameters and return values – plaintext and/or ciphertext data	Network Interface
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	Keyboard Interface, Mouse Interface
Status Output	API output parameters and return/error codes that provide status information used to indicate the state of the module	Display Controller, Network Interface
Power	None	Power Supply

Table 9 – Logical Interface / Physical Interface Mapping

As shown in Figure 1 – Module Boundary and Interfaces Diagram and Table 11 – Module Services, Roles, and Descriptions, the output data path is provided by the data interfaces and is logically disconnected from processes performing key generation or zeroization. No key information will be output through the data output interface when the module zeroizes keys.

2.4 Roles, Services, and Authentication

2.4.1 Assumption of Roles

The module supports two distinct operator roles, which are the User and Crypto 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.

The module does not support a Maintenance role or bypass capability. The module does not support authentication.

Role	Role Description	Authentication Type
CO	Crypto Officer – Powers on and off the module	N/A – Authentication is not a requirement for Level 1
User	User – The user of the complete API	N/A – Authentication is not a requirement for Level 1

Table 10 – Description of Roles

2.4.2 Services

All services implemented by the module are listed in Table 11 – Module Services, Roles, and Descriptions. The second column provides a description of each service, and availability to the Crypto Officer and User is indicated in columns 3 and 4, respectively.

Service	Description	CO	User
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
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

Service	Description	CO	User
NDRNG Callback	Gathers entropy in a passive manner from a user-provided function.		X
Utility	Miscellaneous utility functions, does not access CSPs.		X

Table 11 – Module Services, Roles, and Descriptions

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 12 – CSP Access Rights within Services 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.

Services	CSPs									
	AES Keys	DH Keys	DRBG Keys	DSA Keys	EC Agreement Key	ECDSA Keys	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		
X9.63 Derivation Function								R		
SP 800-56A-rev2 Concatenation Derivation Function								R		

Services	CSPs									
	AES Keys	DH Keys	DRBG Keys	DSA Keys	EC Agreement Key	ECDSA Keys	HMAC Keys	KDF Secret Values	RSA Keys	Triple-DES Keys
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										

Table 12 – CSP Access Rights within Services

2.5 Physical Security

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

2.6 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 in this section. 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 module was tested on the following platforms:

- CentOS 6 and OpenJDK 1.7 running on HP ProLiant DL360 G7 Server with Intel Xeon X5670

FIPS 140-2 validation compliance is maintained for other versions of the respective operating system family where the module source code is unmodified, and the requirements outlined in NIST IG G.5 are met. No claim can be made as to the correct operation of the module or the security strengths of the generated keys when ported to an operational environment which is not listed on the validation certificate.

The GPC(s) used during testing met Federal Communications Commission (FCC) FCC Electromagnetic Interference (EMI) and Electromagnetic Compatibility (EMC) requirements for business use as defined by 47 Code of Federal Regulations, Part15, Subpart B.

2.6.1 Use of External RNG

The module does not include an entropy source. Entropy is loaded via callbacks using the NDRNG Callback service. The module does not exercise control over the amount or quality of the entropy that is provided to it in response to callbacks, therefore the caveat “no assurance of the minimum strength of generated keys” is applicable.

The operator should use an appropriate entropy source to provide entropy when required. The module will request entropy using callbacks, as appropriate to the security strength and seeding configuration for the DRBG that is using the entropy. 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 callback returns the requested amount of entropy, blocking if necessary.

2.7 Self-Tests

Each time the module is powered up, it tests that the cryptographic algorithms still operate correctly and that sensitive data has 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 13 – Power-Up Self-Tests. 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.

2.7.1 Power-Up Self-Tests

Test Target	Description
Software Integrity Check	HMAC-SHA-256 (HMAC Cert. #3114)
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-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

Test Target	Description
Key Transport Using RSA	KATs: SP 800-56B specific KATs per IG D.4 Key sizes: 2048 bits

Table 13 – Power-Up Self-Tests

2.7.2 Conditional Self-Tests

The module implements the following conditional self-tests upon key generation, or random number generation (respectively):

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.
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.

Table 14 – Conditional Self-Tests

2.8 Mitigation of Other Attacks

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

The first countermeasure 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 countermeasure 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 of 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 Security Rules and Guidance

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 provides two distinct operator roles: User and Cryptographic Officer.
2. The module does not provide authentication.
3. The operator may command 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 is 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.

3.1.1 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 2.1.3 – 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.

3.1.2 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 using 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.

3.1.3 Enforcement and Guidance for AES 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, 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 2.2.1 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.

3.1.4 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.

3.1.5 Software Installation

The module is provided directly to solution developers and is not available for direct download to the general public. The module and its host application are to be installed on an operating system specified in Section 2.6 or on an operating system where portability is maintained.

4 References and Acronyms

4.1 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 800-67	*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*

Abbreviation	Full Specification Name
SP 800-108	*Recommendation for Key Derivation Using Pseudorandom Functions*
SP 800-132	*Recommendation for Password-Based Key Derivation*
SP 800-135	*Recommendation for Existing Application–Specific Key Derivation Functions*

Table 15 – References

4.2 Acronyms

The following table defines acronyms found in this document:

Acronym	Term
AES	Advanced Encryption Standard
API	Application Programming Interface
CBC	Cipher-Block Chaining
CCM	Counter with CBC-MAC
CCCS	Canadian Centre for Cyber Security
CDH	Computational Diffie-Hellman
CFB	Cipher Feedback Mode
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CO	Crypto 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 Algorithm
DSTU4145	Ukrainian DSTU-4145-2002 Elliptic Curve Scheme
EC	Elliptic Curve
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standard

Acronym	Term
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	(Keyed-) Hash Message Authentication Code
IG	Implementation Guidance
IV	Initialization Vector
JAR	Java ARchive
JCA	Java Cryptography Architecture
JCE	Java Cryptography Extension
JDK	Java Development Kit
JRE	Java Runtime Environment
JVM	Java Virtual Machine
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, 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

Acronym	Term
TLS	Transport Layer Security
USB	Universal Serial Bus
XOF	Extendable-Output Function

Table 16 – Acronyms and Terms