Box JCA Cryptographic Module 1.0

FIPS 140-2 Level 1 Security Policy

Version Number: 1.5

Date: February 29, 2016

Table of Contents

1. MODULE OVERVIEW 3

2. MODES OF OPERATION 5

2.1 APPROVED CRYPTOGRAPHIC FUNCTIONS 5
2.2 ALL OTHER ALGORITHMS 6

3. PORTS AND INTERFACES 6

4. ROLES AND SERVICES 7

4.1 CRYPTO-OFFICER AND USER ROLES 7
4.2 APPROVED SERVICES 7

5. CRYPTOGRAPHIC KEYS AND CSPS 8

6. SELF-TESTS 9

1. Module Overview

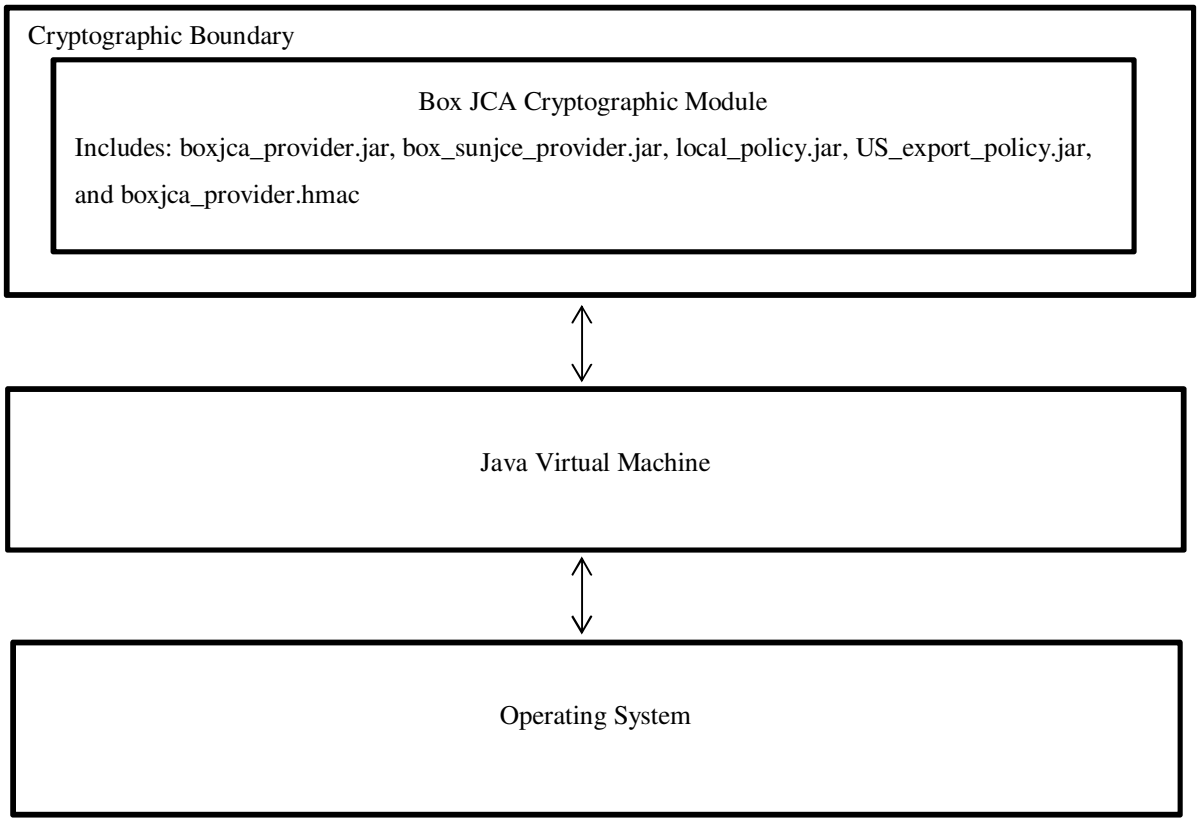
The Box Java Cryptography Architecture (JCA) Cryptographic Module is a Java Cryptography Architecture provider that provides encryption, hashing and random number generation utilizing FIPS 140-2 approved algorithms.

The JCA is a software module that executes in a modifiable operational environment by a general purpose computer.

The logical boundary of the software module includes of the following components (Figure 1):

- boxjca_provider.jar
- box_sunjce_provider.jar
- local_policy.jar
- US_export_policy.jar
- boxjca_provider.hmac

Figure 1: Box JCA Cryptographic Module Block Diagram



FIPS 140-2 conformance testing was performed at Security Level 1. The configuration tested by the lab is described in Table 1.

Table 1: Configuration tested by the lab

Software Component	Operating System	Java Run-Time Environment (JRE)	CPU
• boxjca_provider.jar • box_sunjce_provider.jar • local_policy.jar • US_export_policy.jar • boxjca_provider.hmac	Scientific Linux 6.4	• JRE 1.6.0 • JRE 1.7.0	Dell PowerEdge R610 using Intel(R) Xeon(R) X5675

The Box JCA Cryptographic Module meets FIPS 140-2 Level 1 requirements as described in Table 2.

Table 2: Module Security Level Statement

FIPS Security Area	Security Level
Cryptographic Module Specification	1
Module Ports and Interfaces	1
Roles, Services and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A

2. Modes of Operation

In the FIPS 140-2 approved mode of operation, the JCA module performs FIPS approved security functions as listed in section 2.1 Approved Cryptographic Functions. When the Java virtual machine begins execution, it loads the class BoxProvider of the boxjca_provider and instantiates the provider if the power-up self-tests have been passed successfully. The power-up self-tests are called from the constructor of the class BoxProvider.

2.1 Approved Cryptographic Functions

The following approved cryptographic algorithms are used in the FIPS approved mode of operation.

Table 3: Approved Cryptographic Functions

Algorithm	CAVP Certificate
AES ECB (128 , 192 , 256) AES CTR (128 , 192 , 256)	2666
SHS (SHA1, SHA256, SHA384 and SHA512)	2239
HMAC (HMAC-SHA1, HMAC-SHA256, HMAC-SHA384, and HMAC-SHA512)	1657
SP 800-90A HMAC_DRBG	429

2.2 All Other Algorithms

In the non-FIPS mode of operation the module performs non-approved functions listed in this section. With the exception of NDRNG these functions shall not be used in FIPS approved mode of operation.

- AES (CBC, PCBC, CTS, CFB, CFB8, CFB128, OFB, OFB8, and OFB128 modes; non-compliant)
- Blowfish
- DES
- Triple-DES (non-compliant)
- RC2
- Diffie-Hellman (non-compliant)
- PBEWithMD5AndDES
- PBEWithMD5AndTripleDES
- PBEWithSHA1AndTripleDES
- PBEWithSHA1AndRC2_40
- ARCFOUR
- RSA (key wrapping; non-compliant)
- HMAC-MD5
- PBKDF2WithHmacSHA1
- DSA (non-compliant)
- MD2
- MD5
- SHA1PRNG
- NDRNG

3. Ports and Interfaces

The logical interfaces to the module are implemented via an Application Programming Interface (API). The following table describes each logical interface.

Table 4: FIPS 140-2 Logical Interfaces

Logical Interface	Description
Data Input	Input parameters that are supplied to the API commands
Data Output	Output parameters that are returned by the API commands
Control Input	API commands
Status Output	Exceptions and return status provided by API commands

4. Roles and Services

4.1 Crypto-Officer and User roles

There are two roles the module supports: Crypto-Officer role and a User role.

4.2 Approved Services

The module provides the following cryptographic services.

Table 5: Roles and Services

Service	Corresponding Roles	Types of Access to Cryptographic Keys and Critical Security Parameters (CSPs) R – Read W – Write or Create Z – Zeroize
AES Cipher Encrypt / Decrypt	User Crypto-Officer	AES Key: R
SHA hash computation	User Crypto-Officer	None
HMAC hash computation	User Crypto-Officer	HMAC Key: R
HMAC-SHA256 DRBG Random output	User Crypto-Officer	Entropy Input: R DRBG seed: R, W AES key: W HMAC key: W
Perform Self-Tests	User Crypto-Officer	N/A
Zeroization by power cycling or rebooting the system	User Crypto-Officer	All: Z
Show status	User Crypto-Officer	N/A
Installation of the module	Crypto-Officer	N/A

The module provides the following APIs:

- Cipher.getInstance
- Cipher.init
- Cipher.doFinal
- MessageDigest.getInstance
- MessageDigest.digest
- Mac.getInstance
- boxProvider.powerOnSelfTest
- Mac.init
- Mac.doFinal
- SecureRandom.getInstance
- SecureRandom.setSeed
- SecureRandom.nextBytes

The user must call the module's APIs with appropriate parameters to invoke the module's services. See user guidance for details.

Non-Approved cryptographic services are implementations of Non-Approved algorithms. They are listed in the Section 2.2.

5. Cryptographic Keys and CSPs

The table below describes cryptographic keys and CSPs used by the module.

Table 6: Cryptographic keys and CSPs employed by the module

Key	Description/Usage	Origin	Zeroization
AES Key	Used during AES encryption and decryption	Generated using HMAC_DRBG	Zeroized during power cycle or reboot
HMAC Key	Used during computation of HMAC	Generated using HMAC_DRBG	Zeroized during power cycle or reboot
Entropy Input	Used during construction of the seed	Generated using NDRNG	Zeroized during power cycle or reboot
DRBG Seed	Used during instantiation of the DRBG	Constructed in accordance with SP 800-90A	Zeroized during power cycle or reboot

The keys must be generated by using SP 800-90A HMAC_DRBG. The Keys and CSPs are stored in plain text within the module.

6. Self-tests

The module performs power-up and conditional self-tests that are listed in the table below. Upon failure of a power-up self-test the module throws an exception and fails to load. Upon failure of a conditional self-test the module throws an exception and returns no data to the caller.

The following table describes self-tests implemented by the module.

Table 7: Self-Tests

Algorithm	Test
AES encrypt	Known Answer Test [KAT]
AES decrypt	KAT
SHA1	KAT
SHA256	KAT
SHA384	KAT
SHA512	KAT
HMAC-SHA1	KAT
HMAC-SHA256	KAT
HMAC-SHA384	KAT
HMAC-SHA512	KAT
SP800-90A HMAC_DRBG	KAT Continuous Random Number Generator test DRBG Health test
Native NDRNG	Continuous Random Number Generator test
Box entropy source NDRNG	Continuous Random Number Generator test
Software integrity test	KAT using HMAC