Certes Networks, Inc.

CryptoFlow Net Creator Java Crypto Module

Software Version: 1.0

FIPS 140-2 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 1.14

Prepared for:	**Prepared by:**
Certes Networks, Inc.	**Corsec Security, Inc.**
300 Corporate Center Drive, Suite 140	13921 Park Center Road, Suite 460
Pittsburgh, PA 15108	Herndon, VA 20171
United States of America	United States of America
Phone: +1 412 262 2571	Phone: +1 703 267 6050
www.certesnetworks.com	www.corsec.com

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1 Purpose

This is a non-proprietary Cryptographic Module Security Policy for the CryptoFlow Net Creator Java Crypto Module (software version: 1.0) from Certes Networks, Inc. (Certes). This Security Policy describes how the CryptoFlow Net Creator Java Crypto 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 Communications Security Establishment (CSE) Cryptographic Module Validation Program (CMVP) website at http://csrc.nist.gov/groups/STM/cmvp.

This document also describes how to run the module in a secure FIPS-Approved mode of operation. This policy was prepared as part of the Level 1 FIPS 140-2 validation of the module. The CryptoFlow Net Creator Java Crypto Module is referred to in this document as the CryptoFlow Net Creator Java Crypto Module or the module.

1.2 References

This document deals only with operations and capabilities of the 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 Certes website (www.certesnetworks.com) contains information on the full line of products from Certes.
- The CMVP website (http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm) contains contact information for individuals responsible for answer technical or sales-related questions for the module.

1.3 Document Organization

The Security Policy document is organized into two primary sections. Section 2 provides an overview of the validated modules. This includes a general description of the capabilities and the use of cryptography, as well as a presentation of the validation level achieved in each applicable functional area of the FIPS standard. It also provides high-level descriptions of how the module meets FIPS requirements in each functional area. Section 3 documents the guidance needed for the secure use of the module, including initial setup instructions and management methods and policies.

2. CFNC Java Crypto Module

2.1 Overview

Certes Networks, Inc. CryptoFlow Solution combines the flexibility and power of virtualized networking with the most secure cryptographic technology. The CryptoFlow Solution is a suite of components consisting of Certes Enforcement Point (CEP) encryption appliances and CryptoFlow Net Creator (CFNC) central security and policy management software.

The CEP encryption appliance provides high-speed processing capabilities to protect data travelling over untrusted networks while in transit between sites. The CEP encryption appliances are managed locally via a CLI[1] hosted on the appliance itself or remotely via the CFNC GUI[2] running on a separate Linux based platform. The CEP CLI is used for initial setup, troubleshooting, and diagnostic tasks

CFNC is a web-based management GUI that is used to configure and monitor the CEP encryption appliances, store and deploy policies (or rules), and provide key management and auditing capabilities. The CEP appliances encrypt traffic according to the keys and policies received from the CFNC over a secure channel.

The CryptoFlow Net Creator Java Crypto Module v1.0 is a cryptographic library embedded in the CFNC application software. The CryptoFlow Net Creator Java Crypto Module v1.0 supplies the cryptographic functionality to support TLS[3] -secured communication between remote administrators using CFNC and the CEP appliances.

The CFNC Java Crypto Module is validated at the FIPS 140-2 Section levels shown in Table 1 below.

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[4]
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC[5]	1
9	Self-tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

[1] CLI – Command Line Interface
[2] GUI – Graphical User Interface
[3] TLS – Transport Layer Security
[4] N/A – Not Applicable
[5] EMI/EMC – Electromagnetic Interference / Electromagnetic Compatibility

2.2 Module Specification

The CFNC Java Crypto Module is a software module with a multiple-chip standalone embodiment. The overall security level of the module is 1. The module consists of a software cryptographic library. The classes within the library are executed within the Java Virtual Machine (JVM) using the classes of the Java Runtime Environment (JRE). The Java Security Manager (JSM) is configured within the JVM to ensure the module is operated in a FIPS-Approved mode of operation. The JVM interfaces with the operating system (OS).

For FIPS 140-2 conformance testing, the module was tested and found compliant when running on the following environment(s):

- Dell PowerEdge R330 XL server with an Intel Xeon E3 v5 processor running CentOS 6.7

The module implements the FIPS-Approved algorithms listed in Table 2 below.

Table 2 – FIPS-Approved Cryptographic Algorithms

Certificate Number	Algorithm	Standard	Mode / Method	Key Lengths / Curves / Moduli	Use
5028	AES[6]	FIPS PUB 197	CBC[7], ECB[8]	128, 192, 256 bits	Encryption/decryption
		NIST SP[9] 800-38C	CCM[10]	128, 192, 256 bits	Encryption/decryption
		NIST SP 800-38D	GCM[11]	128, 192, 256 bits	Encryption/decryption
Vendor Affirmation	CKG[12]	NIST SP 800-133	-	-	Key generation
1573	CVL[13]	NIST SP 800-135 rev1	TLS v1.2		Key derivation function **No parts of the TLS protocol, other than the KDF, have been tested by the CAVP[14] and CMVP.*
1574	CVL	NIST SP 800-56A	ECCCDH[15] primitive	NIST-defined curves (P-256 and P-384)	Shared secret computation
1842	DRBG[16]	NIST SP 800-90A rev1	CTR-based	AES-128, AES-256	Deterministic random bit generation **AES-128 is not used by any of the module's services*
1284	ECDSA[17]	FIPS PUB 186-4	PKG[18], PKV[19]	P-256, P-384	Key pair generation

[6] AES – Advance Encryption Standard
[7] CBC – Cipher Block Chaining
[8] ECB – Electronics Code Book
[9] SP – Special Publication
[10] CCM – Counter with CBC-MAC
[11] GCM – Galois Counter Mode
[12] CKG – Cryptographic Key Generation
[13] CVL – Component Validation List
[14] CAVP – Cryptographic Algorithm Validation Program
[15] ECCCDH – Elliptic Curve Cryptography Cofactor Diffie-Hellman
[16] DBRG – Deterministic Random Bit Generator
[17] ECDSA – Elliptic Curve Digital Signature Algorithm
[18] PKG – Pairwise Key Generation
[19] PKV – Public Key Validation

Certificate Number	Algorithm	Standard	Mode / Method	Key Lengths / Curves / Moduli	Use
			SIG(gen), SIG(ver)	P-256, P-384	Digital signature generation and verification
3342	HMAC[20]	FIPS PUB 198-1	SHA-256. SHA-384, SHA-512	256, 384, 512 bits	Message authentication
Vendor Affirmation	PBKDF[21]	NIST SP 800-132	Option 1a with HMAC SHA-512	-	Password to key conversion
4087	SHS[22]	FIPS PUB 180-4	SHA-1[23], SHA-256, SHA-384, SHA-512	-	Message digest

The vendor affirms the following cryptographic security methods:

- As per *NIST SP 800-133*, the module uses its FIPS-Approved counter-based DRBG to generate cryptographic keys. The resulting symmetric key or generated seed is an unmodified output from the DRBG. The module's DRBG is seeded by a GET request to `/dev/random`, a non-deterministic random number generator (NDRNG) outside the module's logical boundary. Each GET request returns a minimum of 112 bits of entropy.

- As per *NIST SP 800-132*, the module uses PBKDF2 option 1a (note that keys derived using PBKDF2 may only be used in storage applications). This function takes an input salt that is 128 bits in length with a 128-bit alphanumeric password and produces a random value of 256 bits. The iteration count is 2048. The upper bound for the probability of guessing the password at random is 1/((26+26+10)^16).

Table 3 lists the module's non-FIPS-Approved algorithm implementations that are allowed for use in FIPS-mode.

Table 3 – Allowed Cryptographic Algorithms

Algorithm	Caveat	Use
NDRNG	The module generates cryptographic keys whose strengths are modified by the available entropy	Seeding for the DRBG

As a software module, the CFNC Java Crypto Module has both a logical cryptographic boundary and a physical cryptographic boundary. The physical and logical boundaries are described in sections 2.2.1 and 2.2.2, respectively.

2.2.1 Physical Cryptographic Boundary

As a software cryptographic module, the module has no physical components. Therefore, the physical boundary of the cryptographic module is defined by the hard enclosure around the host server on which it runs.

[20] HMAC – (keyed-) Hashed Message Authentication Code
[21] PBKDF – Password Based Key Derivation Function
[22] SHS – Secure Hash Standard
[23] The SHA-1 algorithm is only used during installation as a hash for license verification. There are no other uses of this algorithm within the module.

The host server hardware consists of a motherboard, a central processing unit (CPU), random access memory (RAM), read-only memory (ROM), hard disk(s), hardware case, power supply, and fans. Other devices may be attached to the hardware appliance such as a monitor, keyboard, mouse, DVD[24] drive, printer, video adapter, audio adapter, or network adapter. In the validated configuration, the processor is an Intel Xeon processor. Please see Figure 1 below for the host server block diagram and physical cryptographic boundary.

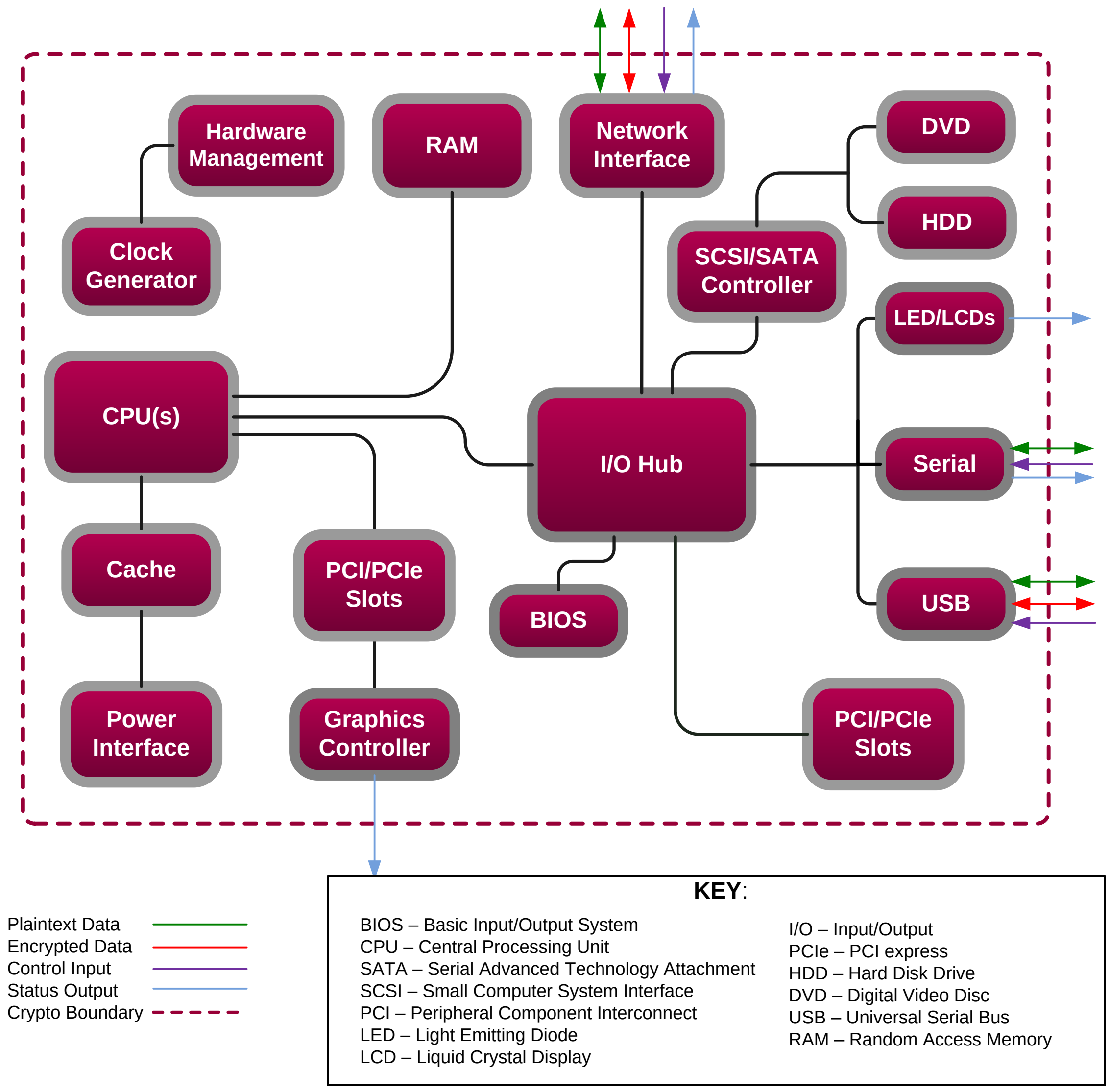


Figure 1 – Host Server Physical Block Diagram

2.2.2 Logical Cryptographic Boundary

The logical cryptographic boundary surrounds the cryptographic library executable. The module is the single software component within the logical cryptographic boundary. The cryptographic module is used by the calling application to provide symmetric and asymmetric cipher operation, signature generation and verification, hashing, cryptographic key generation, random number generation, message authentication functions, and secure key

[24] DVD – Digital Versatile Disc

agreement/key exchange protocols. The module is entirely contained within the physical cryptographic boundary described in section 2.2.1.

Figure 2 below shows the logical block diagram of the module executing in memory and its interactions with surrounding software components as well as the module's logical cryptographic boundary.

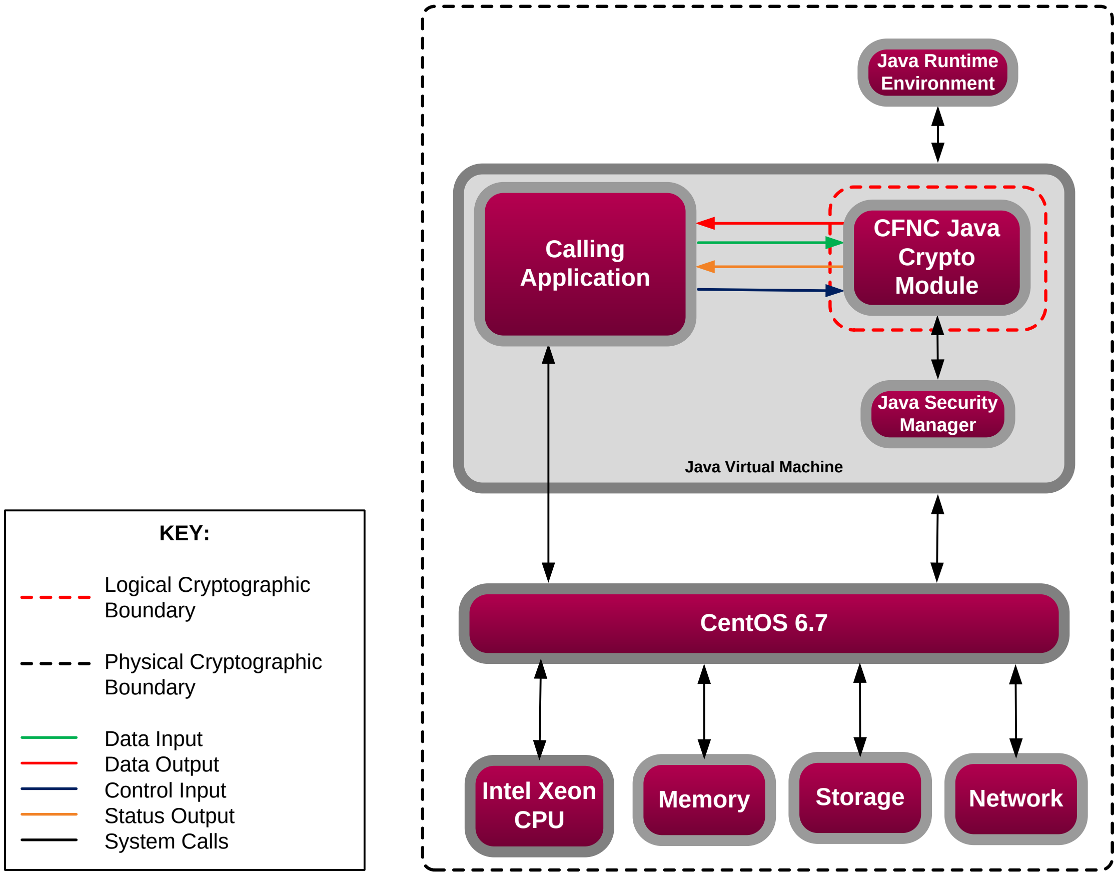


Figure 2 – Module Block Diagram (with Logical Cryptographic Boundary)

2.3 Module Interfaces

The module isolates communications to logical interfaces that are defined in the software as an API. The API interface is mapped to the following four logical interfaces:

- Data Input
- Data Output
- Control Input
- Status Output

The module's physical boundary features the physical ports of a host server. The module's manual controls; physical indicators; and physical, logical, and electrical characteristics are those of the host server. The module's logical interfaces are at a lower level in the software. The physical data and control input through physical mechanisms is translated into the logical data and control inputs for the software module. A mapping of the FIPS 140-2 logical interfaces, the physical interfaces, and the module interfaces can be found in Table 4 below.

Table 4 – FIPS 140-2 Logical Interface Mappings

FIPS Interface	Physical Interface	Module Interface (API)
Data Input	USB ports, network interface, serial ports	The API call parameters that accept input data for processing through their arguments
Data Output	Graphics controller, USB ports, network interface, serial ports	The API call parameters that return generated or processed data back to the caller
Control Input	USB ports, network interface, serial ports	The API call parameters that are used to initialize and control the operation of the module
Status Output	Graphic controller, network interface, serial ports, Audio ports, LCD[25]/LED[26]s	The API call return values indicating the status of the module or service performed
Power Input	AC[27] power interface	-

2.4 Roles and Services

There are two authorized roles that module operators may assume: Crypto Officer (CO) role and a User role. Authorized operators implicitly assume the set of roles comprised of the CO role and User role when exercising any of the module's services. The module does not allow multiple concurrent operators in the FIPS-Approved mode of operation. Per section 6.1 of the *FIPS 140-2 Implementation Guidance*, the calling application that loaded the module is the only operator.

Descriptions of the services available are provided in Table 5 below. Please note that the keys and Critical Security Parameters (CSPs) listed in the table indicate the type of access required using the following notation:

- R – Read: The CSP is read.
- W – Write: The CSP is established, generated, or modified.
- X – Execute: The CSP is used within an Approved or Allowed security function or authentication mechanism.
- D – Delete: The CSP is zeroized.

Input parameters of an API call that are not specifically a signature, hash, message, plaintext, ciphertext, or a key are **not** itemized in the "Input" column, since it is assumed that most API calls will have such parameters. Values in the "Input" and "Output" columns are sent from the module's logical boundary via API calls.

[25] LCD – Liquid Crystal Display
[26] LED – Light Emitting Diode
[27] AC – Alternating Current

Table 5 – Mapping of Operator Services to Inputs, Outputs, CSPs, and Type of Access

Service	Operator		Description	Input	Output	CSP and Type of Access
	CO	User				
Show status	✓	✓	Used to determine if the module is ready or returns the current mode of the module	None	Status	None
Zeroize/Power-off	✓	✓	Used to unload the module, zeroized CSPs, and power off the environment	Power cycle; unload module	None	All CSPs – D
Power cycling	✓	✓	Used to zeroized keys reboot the module, and execute power-up self-tests on demand	Power cycle; reload module	Status	All CSPs – D
Data encryption	✓	✓	Used to encrypt data	API call parameters	Status, ciphertext	AES key – R, X AES GCM IV – R, X AES GCM key – R, X
Data decryption	✓	✓	Used to decrypt data	API call parameters	Status, plaintext	AES key – R, X AES GCM IV – R, X AES GCM key – R, X
MAC[28] calculation	✓	✓	Used to calculate data integrity codes for HMAC	API call parameters	Status, hash	HMAC key – R, X
Signature generation	✓	✓	Used to generate signature	API call parameters	Status, signature	ECDSA private key – R, X
Signature verification	✓	✓	Used to verify a digital signature	API call parameters	Status	ECDSA public key – R, X
DRBG	✓	✓	Used for random number and key generation per *NIST SP 800-90A rev1*	API call parameters	Status, generated keys	AES key – W AES GCM key – W DRBG Seed – R, W, X DRBG Entropy – R, X DRBG "Key" value – R, W, X DRBG "V" value – R, W, X ECDSA public key – W ECDSA private key – W HMAC key – W
Message hashing	✓	✓	Used to generate SHA-2 message digest	API call parameters	Status, hash	None
Keyed message hashing	✓	✓	Used to create data integrity code for HMAC and in the PBKDF generation	API call parameters	Status, hash	HMAC key – R, X
Shared secret generation	✓	✓	Used to calculate a shared secret suitable for use as input to a TLS KDF	API call parameters	Status, shared secret	ECCCDH shared secret – W

[28] MAC – Message Authentication Code

Service	Operator		Description	Input	Output	CSP and Type of Access
	CO	User				
TLS key derivation	✓	✓	Used to produce TLS session and integrity keys using a shared secret	API call parameters	Status, TLS keys	ECCCDH shared secret – R, X TLS master secret – W, X AES GCM key – W HMAC key – W
Password-based key derivation	✓	✓	Used to generate an AES-CCM key using a password and message hash	API call parameters	Status, key	Key store HMAC key – W Key store AES key – W Key store passphrase – R, X

2.5 Physical Security

The cryptographic module is a software module and does not include physical security mechanisms. Therefore, per section G.3 of the *FIPS Implementation Guidance*, requirements for physical security are not applicable.

2.6 Operational Environment

The module was tested and found to be compliant with FIPS 140-2 requirements on the following operational environment(s):

- Dell PowerEdge R330 XL server with an Intel Xeon E3 v5 processor running CentOS 6.7

All cryptographic keys and CSPs are under the control of the OS, which protects its CSPs against unauthorized disclosure, modification, and substitution. The module only allows access to CSPs through its well-defined API.

2.7 Cryptographic Key Management

The module supports the CSPs listed in Table 6 below.

Table 6 – Cryptographic Keys, Cryptographic Key Components, and CSPs

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
AES key	128, 192, 256-bit key	Generated internally via Approved DRBG	Output in plaintext	Plaintext in volatile memory	Unload module, remove/cycle power	Encryption, decryption
AES GCM key	128, 192, 256-bit key	Generated internally via Approved DRBG	Output in plaintext	Plaintext in volatile memory	Unload module, remove/cycle power	Encryption, decryption and authentication
AES GCM initialization vector[29]	Minimum 96-bit value	Generated externally, input via API call parameter	Never exits the module	Plaintext in volatile memory	Unload module, remove/cycle power	Initialization vector for AES GCM
DRBG Seed	384-bit random value	Generated internally using nonce along with DRBG entropy input	Never exits the module	Plaintext in volatile memory	Unload module, remove/cycle power	Seeding material for SP 800-90A DRBG
DRBG Entropy	Minimum 112-bit value	Generated internally[30] and passed into the module upon request	Never exits the module	Plaintext in volatile memory	Unload module, remove/cycle power	Entropy material for SP 800-90A DRBG
DRBG 'Key' Value	Internal state value	Generated internally	Never exits the module	Plaintext in volatile memory	Unload module, remove/cycle power	Used for CTR_DRBG
DRBG 'V' Value	128-bits	Generated internally	Never exits the module	Plaintext in volatile memory	Unload module, remove/cycle power	Used for CTR_DRBG
ECCCDH shared secret	256-bit shared secret	Established internally via ECCCDH shared secret computation	Output in plaintext	Plaintext in volatile memory	Zeroized after service completes	Used for deriving TLS master secret in TLS KDF
ECDSA private key	P-256, P-384 curves	Generated internally via Approved DRBG	Output in plaintext	Plaintext in volatile memory	Unload module, remove/cycle power	Signature generation, decryption
ECDSA public key	P-256, P-384 curves	Generated internally via Approved DRBG	Output in plaintext	Plaintext in volatile memory	Unload module, remove/cycle power	Signature verification, encryption
HMAC key	HMAC 256, 384, or 512-bit key	Generated internally via Approved DRBG	Output in plaintext	Plaintext in volatile memory	Unload module, remove/cycle power	Message authentication with SHS
Key store HMAC key	HMAC SHA-512 key	Generated internally using PBKDF input parameters	Never exits the module	Plaintext in volatile memory	Unload module, remove/cycle power	Used for deriving the key store AES key

[29] The module conforms to TLSv1.2 GCM cipher suites as specified in section 3.3.1 of *NIST SP 800-52 rev1*. When the nonce_explicit part of the IV exhausts the maximum number of possible values for a given session key, a new encryption key will be established per RFC 5246 section 7.4.1.1. and 7.4.1.2.

[30] The module employs a non-deterministic random number generator specific to the host operating system, which is outside of the logical cryptographic boundary.

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
Key store AES key	AES-CCM 128, 192, 256-bit key	Generated internally using PBKDF	Never exits the module	Plaintext in volatile memory	Unload module, remove/cycle power	Used for encryption of key storage
Key store passphrase	Passphrase used for PBKDF	Generated externally, input via API call parameter	Never exits the module	Plaintext in volatile RAM	Unload module, remove/cycle power	Used for deriving key store HMAC and AES key
TLS master secret	384-bit master secret for TLS	Derived internally using the TLS pre-master secret via TLS KDF	Never exits the module	Plaintext in volatile RAM	End TLS session, unload module, remove/cycle power	Used for deriving AES and HMAC keys used in TLS

2.8 EMI / EMC

The CFNC Java Crypto Module was tested on the server listed in section 2.6 above. This server was tested and found conformant to the EMI/EMC requirements specified by 47 Code of Federal Regulations, Part 15, Subpart B, Unintentional Radiators, Digital Devices, Class A (business use).

2.9 Self-Tests

Cryptographic self-tests are performed by the module when the module is first powered up and loaded into memory as well as when a random number or asymmetric key pair is created. The following sections list self-tests performed by the module, their expected error statuses, and the error resolutions.

2.9.1 Power-Up Self-Tests

The module performs the following self-tests at power-up:

- Software integrity check (using HMAC SHA-256)
- Known Answer Tests (KATs)
    - AES-ECB encrypt KAT
    - AES-ECB decrypt KAT
    - AES-CCM generation/verification KAT
    - AES-GCM generation/verification KAT
    - CTR_DRBG KAT
    - ECDSA Signature Generation/Signature Verification
    - SHA-1 KAT
    - HMAC (with SHA-256) KAT
    - HMAC (with SHA-384) KAT
    - HMAC (with SHA-512) KAT
    - ECCCDH Primitive “Z” Computation[31] test (with P-256 curve)

2.9.2 Conditional Self-Tests

The module performs the following conditional self-tests:

- Continuous RNG[32] Test for the DRBG
- Continuous RNG Test for the NDRNG
- ECDSA pairwise consistency test

2.9.3 DRBG Health Checks

The module performs the following DRBG health checks at power-up and conditionally as needed:

- SP 800-90A DRBG Health Checks
    - SP 800-90 DRBG (CTR) Instantiate Test
    - SP 800-90 DRBG (CTR) Generate Test

[31] The ECCCDH Primitive “Z” Computation KAT is compliant with FIPS 140-2 IG 9.6.
[32] RNG – Random Number Generator

  - SP 800-90 DRBG (CTR) Reseed Test

2.9.4 Self-Test Failure Handling

Upon completion of all self-tests, `CryptoServicesRegistrar.isInApprovedOnlyMode()` is checked. If a successful status of `TRUE` is returned, then the module is ready to operate in a FIPS-Approved mode of operation. If an error is returned, then the calling application must be rebooted (thus restarting the module) to clear the error state.

If rebooting the host server does not result in the successful execution of self-tests, then the module will not be able to operate in FIPS-Approved mode. The CO must contact Certes Networks, Inc. for assistance.

2.10 Mitigation of Other Attacks

This section is not applicable. The module does not claim to mitigate any attacks beyond the FIPS 140-2 Level 1 requirements for this validation.

3. Secure Operation

The CFNC Java Crypto Module meets Level 1 requirements for FIPS 140-2. The sections below describe how to configure the CFNC Java Crypto Module as a validated module. **If the following steps are not executed, the CFNC Java Crypto Module will be operating outside the scope of this Security Policy and will not be operating as a validated cryptographic module.**

The following paragraphs describe the steps necessary to configure the CFNC Java Crypto Module as a validated module. Once configured as a validated module, the CFNC Java Cryptographic Library only operates in the FIPS approved mode of operation.

3.1 Installation

The module comes factory-installed with Certes' CFNC application software[33]. Thus, the module has no independent installation steps that end-users must follow.

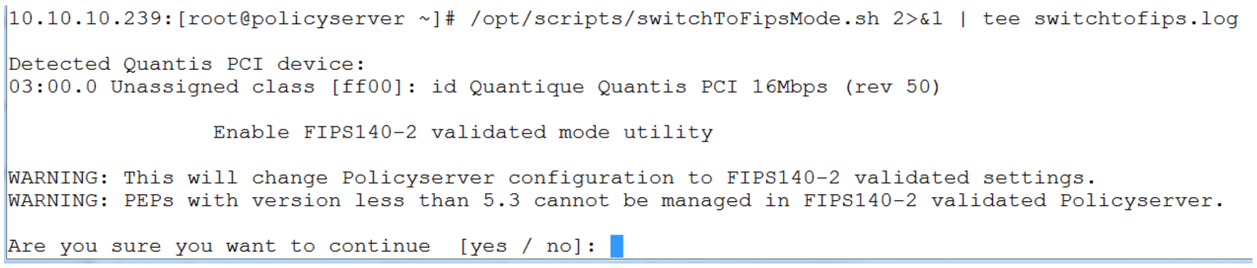
3.2 Configuration

The following steps **shall** be performed to configure the CFNC Java Crypto Module as a validated cryptographic module.:

1. Open a CLI for the CFNC server. Enter the following command:

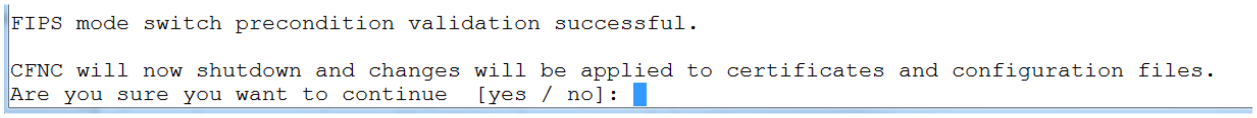
```
/opt/scripts/switchToFipsMode.sh 2>&1 | tee switchtofips.log.
```

The following message is displayed:



```
10.10.10.239:[root@policyserver ~]# /opt/scripts/switchToFipsMode.sh 2>&1 | tee switchtofips.log

Detected Quantis PCI device:
03:00.0 Unassigned class [ff00]: id Quantique Quantis PCI 16Mbps (rev 50)

                Enable FIPS140-2 validated mode utility

WARNING: This will change Policyserver configuration to FIPS140-2 validated settings.
WARNING: PEPs with version less than 5.3 cannot be managed in FIPS140-2 validated Policyserver.

Are you sure you want to continue  [yes / no]:
```

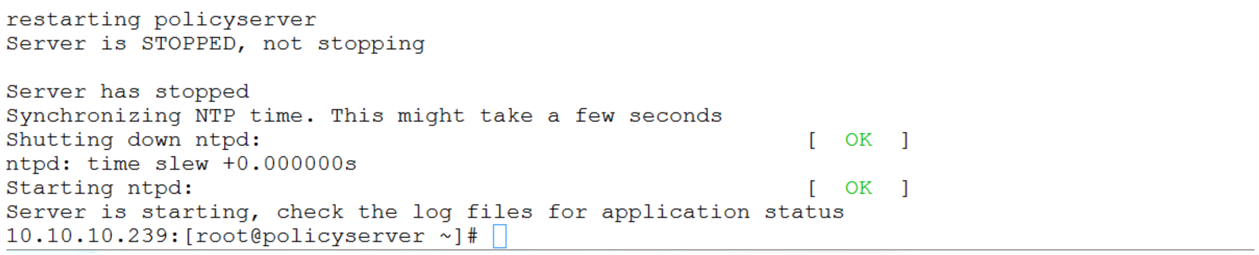
2. Enter **Yes**. The following message is displayed:



```
FIPS mode switch precondition validation successful.

CFNC will now shutdown and changes will be applied to certificates and configuration files.
Are you sure you want to continue  [yes / no]:
```

3. Enter **Yes**. Process notifications scroll by and the server is restarted.

[33] NOTE: While CFNC v5.3 was used for testing, the module is not dependent on any specific version of the calling application. The module can be run using any version of the calling application with which the module is installed.

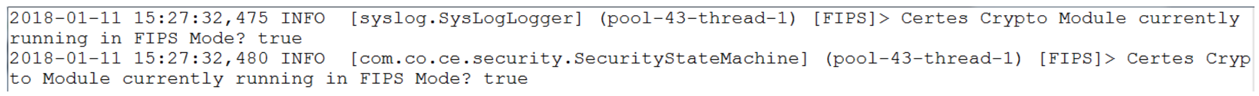


```
restarting policyserver
Server is STOPPED, not stopping

Server has stopped
Synchronizing NTP time. This might take a few seconds
Shutting down ntpd:                                         [  OK  ]
ntpd: time slew +0.000000s
Starting ntpd:                                              [  OK  ]
Server is starting, check the log files for application status
10.10.10.239:[root@policyserver ~]#
```

Upon restart, all self-tests are automatically performed. Once all self-tests have completed successfully the module is operating in the validated configuration. The calling application can make a call to the `CryptoServicesRegistrar.isInApprovedOnlyMode()` API to retrieve the system operating status. Additionally, the CO can run the following command to view the log file displaying the operating status of the module:

```
grep -i fips /opt/jboss/server/policyserver/log/server.log | less
```

When properly configured, the module will return the following in response to the `grep` command:



```
2018-01-11 15:27:32,475 INFO  [syslog.SysLogLogger] (pool-43-thread-1) [FIPS]> Certes Crypto Module currently
running in FIPS Mode? true
2018-01-11 15:27:32,480 INFO  [com.co.ce.security.SecurityStateMachine] (pool-43-thread-1) [FIPS]> Certes Cryp
to Module currently running in FIPS Mode? true
```

Once the system has been configured according to section 3 of this Security Policy, the module does not allow the use of any non-approved security functions and will only operate in the FIPS approved mode. To restore the module to its original state, the host appliance must be restored to a factory state, which can be done by contacting Certes Customer Support (http://support.certesnetworks.com). If the self-tests do not run successfully, reboot the appliance. If this does not clear the problem, the CO must contact Certes Customer Support for additional assistance.

3.3 Initialization

The module's power-up self-tests are automatically executed when it is loaded into memory for execution by the calling application. Once all power-up self-tests have successfully completed, the module is running in its FIPS-Approved mode of operation. No initialization steps are required.

3.4 Operator Guidance

The following sections provide guidance to module operators for the correct and secure operation of the module. As it relates to this module, the calling application (i.e. the product software) takes on the role of both Crypto Officer and User.

3.4.1 Crypto Officer Guidance

No specific management activities are required to ensure that the module runs securely; once operational, the module only executes in a FIPS-Approved mode of operation. However, if any irregular activity is observed or the module is consistently reporting errors, then Certes Customer Support should be contacted.

3.4.2 User Guidance

Although the User does not have any ability to modify the configuration of the module, they should notify the CO if any irregular activity is observed.

3.4.3 General Operator Guidance

The following provide further guidance for the general operation of the module:

- The calling application uses the `CryptoServicesRegistrar.isInApprovedOnlyMode()` API function to retrieve the operating status of the module.
- To execute the module's power-up self-tests on-demand, the module's host server can be rebooted/power-cycled.

3.5 Additional Guidance and Usage Policies

The notes below provide additional guidance and policies that must be followed by module operators:

- As a software cryptographic library, the module's services are intended to be provided to a calling application. Excluding the use of the NIST-defined elliptic curves as trusted third-party domain parameters, all other assurances from *FIPS PUB 186-4* (including those required of the intended signatory and the signature verifier) are outside the scope of the module and are the responsibility of the calling application.
- The calling application shall use entropy sources that meet the security strength required for the CTR_DRBG as shown in Table 3 of *NIST SP 800-90A rev1*.
- The module requests entropy through a `GET` command. Each GET request returns a minimum of 112 bits of entropy. Responses to requests for entropy are blocked by the entropy mechanism until there is sufficient entropy to satisfy the request. If, during power-up self-tests, the minimum entropy strength cannot be met, the module enters a critical error state causing the calling application to shut down, and an error of "Insufficient entropy provided by entropy source" is returned. If rebooting the host server does not result in the successful execution of power-up self-tests, then the module will not be able to operate in FIPS-Approved mode. The CO must contact Certes Networks, Inc. customer support for assistance.
- As the module does not persistently store keys, the calling application is responsible for the storage and zeroization of keys and CSPs passed into and out of the module.
- If power to the module is lost and subsequently restored, the calling application must ensure that any AES-GCM keys used for encryption or decryption are re-distributed.
- In its unconfigured state, the module offers non-compliant implementations of RSA[34] and Diffie-Hellman for key exchange and authentication. When the module is installed and configured as per section 3 of this Security Policy, these algorithms are disabled and no longer available for use.

[34] RSA – Rivest, Shamir, Adelman

3.6 Non-FIPS-Approved Mode

When built and distributed as described in this Security Policy, the module does not support a non-Approved mode of operation.

4. Acronyms

Table 7 below provides definitions for the acronyms used in this document.

Table 7 – Acronyms

Acronym	Definition
AC	Alternating Current
AES	Advanced Encryption Standard
API	Application Programming Interface
BIOS	Basic Input/Output System
CBC	Cipher Block Chaining
CCM	Counter with CBC-MAC
CEP	Certes Enforcement Point
CFNC	CryptoFlow Net Creator
CLI	Command Line Interface
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CO	Cryptographic Officer
CPU	Central Processing Unit
CS	Ciphertext Stealing
CSE	Communications Security Establishment
CSP	Critical Security Parameter
CTR	Counter
CVL	Component Validation List
DEA	Data Encryption Algorithm
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
DVD	Digital Video Disc
ECB	Electronic Code Book
ECCCDH	Elliptic Curve Cryptography Cofactor Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EMI/EMC	Electromagnetic Interference /Electromagnetic Compatibility
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
GMAC	Galois Message Authentication Code
GUI	Graphical User Interface
HDD	Hard Disk Drive

Acronym	Definition
HMAC	(keyed-) Hash Message Authentication Code
IV	Initialization Vector
JRE	Java Runtime Environment
JSM	Java Security Manager
JVM	Java Virtual Machine
KAT	Known Answer Test
KDF	Key Derivation Function
KW	Key Wrap
KWP	Key Wrap with Padding
LCD	Liquid Crystal Display
LED	Light Emitting Diode
N/A	Not Applicable
DRNG	Deterministic Random Number Generator
NIST	National Institute of Standards and Technology
OS	Operating System
PBKDF	Password Based Key Derivation Function
PCIe	PCI express
PKCS	Public Key Cryptography Standard
PKG	Pairwise Key Generation
PKV	Pairwise Key Verification
PSS	Probabilistic Signature Scheme
RAM	Random Access Memory
RNG	Random Number Generator
RSA	Rivest, Shamir, Adelman
SATA	Serial Advanced Technology Attachment
SHA	Secure Hash Algorithm
SCSI	Small Computer System Interface
SHS	Secure Hash Standard
SP	Special Publication
SRTP	Secure Real-time Transport Protocol
TLS	Transport Layer Security
USB	Universal Serial Bus

Prepared by:
Corsec Security, Inc.

13921 Park Center Road, Suite 460
Herndon, VA 20171
United States of America

Phone: +1 703 267 6050
Email: info@corsec.com
http://www.corsec.com