KeyNexus, Inc. Key Management Security Module (KMSM) Cryptographic Module

Version No: 1.0

FIPS 140-2 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 1.1
Date: November 26, 2019

Prepared For:

KeyNexus, Inc.
650 W. Georgia St., Suite 3200
Vancouver, B.C.
Canada V6B 4P7
https:// www.keynexus.com

Prepared By:

EWA-Canada, Ltd.
1223 Michael Street North, Suite 200
Ottawa, Ontario
Canada K1J 7T2
www.ewa-canada.com

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose

This non-proprietary Security Policy for the Key Management Security Module cryptographic module by KeyNexus, Inc. describes how the module meets the security requirements of FIPS 140-2 and how to run the module in a secure FIPS 140-2 mode.

This document was prepared as part of the Level 1 FIPS 140-2 validation of the module. The following table lists the module's FIPS 140-2 security level for each section.

Table 1 - FIPS 140-2 Section Security Levels

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
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self-Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	1

1.2 Background

Federal Information Processing Standards Publication (FIPS PUB) 140-2 – *Security Requirements for Cryptographic Modules* details the requirements for cryptographic modules. More information on the National Institute of Standards and Technology (NIST) and the Communications Security Establishment (CSE) Cryptographic Module Validation Program (CMVP), the FIPS 140-2 validation process, and a list of validated cryptographic modules can be found on the CMVP website:

https://csrc.nist.gov/Projects/Cryptographic-Module-Validation-Program

More information about KeyNexus, Inc.products and cryptographic module can be found on the KeyNexus, Inc. website:

https://keynexus.net/

1.3 Document Organization

This non-proprietary Security Policy is part of the KeyNexus, Inc.software cryptographic module FIPS 140-2 submission package. Other documentation in the submission package includes:

- Product documentation
- Vendor evidence documents
- Finite state model
- Additional supporting documents

The Key Management Security Module (KMSM) cryptographic module is also referred to in this document as the cryptographic module, or the module.

2 Module Overview

The KeyNexus Inc. KMSM cryptographic module is a software library which provides cryptographic functionality to the KeyNexus Key Management Service.

KeyNexus Key Management Service (KMS) allows enterprises to leverage a single, central key management server to support many different encryption and security use cases. KeyNexus KMS can be deployed as a hosted service, on-premise in any virtual environment, in public or private clouds and embedded in a variety of chipset compute environments.

The KMSM module operates with the following components:

- A general purpose computer (GPC)
- A host OS
- A guest OS
- A hypervisor
- A Java Virtual Machine (JVM)

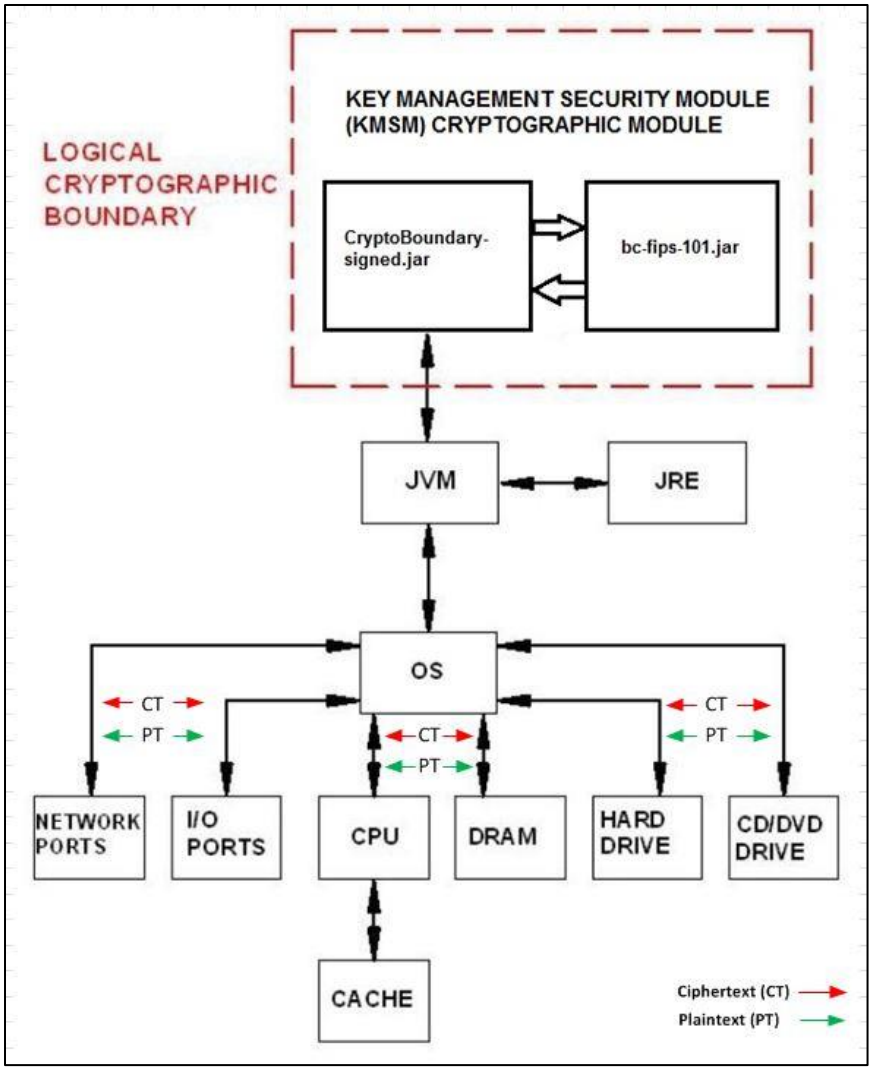
The module will run on other combinations of guest OS, host OS, hypervisor, GPC, and JVM while maintaining its compliance to FIPS 140-2 level 1 requirements. However, the CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

2.1 Cryptographic Module Specification

The cryptographic module is a multi-chip standalone, software only embodiment. The physical boundary of the module is the physical case of the computer, which in this case is a Dell Optiplex desktop. The logical boundary consists of the commands/calls made by the KeyNexus software to the KMSM API.

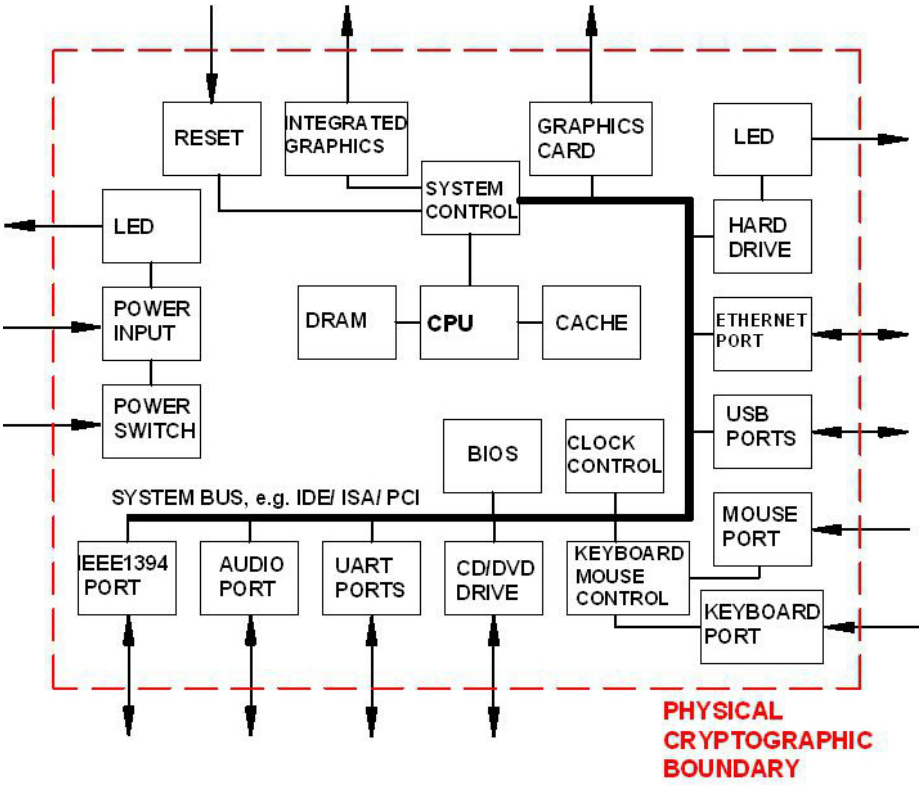
The KMSM cryptographic module consists of two files: CryptoBoundary-signed.jar and bc-fips-1.0.1.jar. CryptoBoundary-signed.jar is the KeyNexus a wrapper which directs API calls to the correct cryptography in the cryptographic library. bc-fips-1.0.1.jar is the Bouncy Castle software cryptographic library version 1.0.1. For FIPS purposes the logical boundary is the API exposed by the CryptoBoundary-signed.jar file.

Figure 1 - Block Diagram of the Key Management Security Module (KMSM) Cryptographic Module



The physical boundary is the same as a typical GPC.

Figure 2 - Block Diagram of the Physical Components of a typical GPC



2.2 Module Configuration

2.2.1 Cryptographic Functionality

The Bouncy Castle cryptographic library provides the cryptographic functionality (services) for the KMSM cryptographic module. The KeyNexus wrapper acts as a funnel/filter for the KeyNexus calling application/processes so that each API call is provided the correct service, (which includes return codes or values from the Bouncy Castle cryptographic library).

2.2.2 FIPS Approved Algorithms

The following algorithms are approved, can be used in FIPS approved mode and have CAVP certificates.

Table 2 - Approved Algorithms with Certificates[1]

Algorithm	Description	Certificate #
AES	[FIPS 197, SP 800-38A] Encryption, Decryption Modes: ECB, CBC, OFB, CFB8, CFB128, CTR Key size: 128, 192, 256 bits	5453
AES – CBC	[Addendum to SP 800-38A, October 2010]	Based on 5453

[1] The KMSM uses a subset of the CAVP tested algorithm functionality

Ciphertext Stealing (CS)	Encryption, Decryption Modes: CBC-CS1, CBC-CS2, CBC-CS3 Key Sizes: 128, 192, 256 bits	(Vendor Affirmed)
AES CCM	[SP800-38C] Generation, Authentication Key sizes: 128, 192, 256 bits	5453
CMAC (AES and Triple-DES)	[SP 800-38B] Generation, Authentication Key sizes: 128,192, 256 bits and 2, 3-key Triple-DES[2][3]	5453 (AES) 2741 (Triple-DES)
GCM/GMAC[4]	[SP 800-38D] Generation, Authentication Key sizes: 128, 192, 256 bits	5453
DRBG	[SP 800-90A] Hash DRBG (SHA-512) Security: 256 bits	2137[5]
ECDSA	[FIPS 186-4] signature generation component, public key generation, signature generation, signature verification, public key validation Curves: 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 [P-192, K-163, B-163 for verification only]	1454
HMAC	[FIPS 202] Integrity test, data authentication, hashing of authentication data SHA size: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA- 512/256	3611
Key Wrapping Using Block Ciphers	[SP 800-38F] Modes: AES KW, KWP Key sizes: 128, 192, 256 bits (provides between 128 and	5453 (AES)

[2] The operator shall ensure that no more than 2^{16} encryptions with the same Triple-DES key are performed in order to be compliant with IG A.13, please refer to section 12.3 User Guidance.

[3] In approved mode of operation, the use of 2-key Triple-DES to generate MACs for anything other than verification purposes is non-compliant, please refer to section 12.3 User Guidance.

[4] GCM with an internally generated IV, see section 7.4 concerning external IVs. IV generation is compliant with IG A.5.

[5] This algorithm certificate contains more functionality that is used by the module

	256 bits of strength)	
	[SP 800-38F] Mode: Triple-DES TKW Key size: 3-key (provides 112 bits of strength)	2741 (Triple-DES)
RSA	[FIPS 186-4, FIPS 186-2, ANSI X9.31-1998, PKCS #1 v2.1 [PSS and PKCS 1.5]] Key pair generation, signature generation, signature verification Key size: 2048, 3072 [1024, 1536, 4096 verification only]	2927
SHS	[FIPS 180-4] Hashing of authentication data SHA size: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256	4373
SHA-3/SHAKE	[FIPS 186-4] SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256	46
Triple-DES	[SP 800-67, SP 800-38F] Encryption/Decryption, key wrap/unwrap Modes: [TECB, TCBC, TCFB8, TCFB64, TOFB, CTR, key wrap/unwrap] Key size: 2-key [decryption only][6], 3-key	2741

2.2.3 Vendor Affirmed Algorithms

Table 3 - Vendor Affirmed Approved Cryptographic Functions

Algorithm	Description	IG
CKG[7]	[SP 800-133] Section 7.1 (Symmetric from DRBG) Using DRBG #2137	Vendor Affirmed IG D.12

2.2.4 Non-Approved but Allowed Cryptographic Functions

Table 4 - Non-Approved but Allowed Cryptographic Functions

Algorithm	Description

[6] The operator shall ensure that no more than 2^{16} encryptions with the same Triple-DES key are performed in order to be compliant with IG A.13, please refer to section 12.3 User Guidance, 2-key encryption is disabled.

[7] Resulting Symmetric keys and seeds used for asymmetric key generation are an unmodified output from the Approved DRBG.

RSA Key Transport	RSA (key wrapping; key establishment methodology provides 112 or 128 bits encryption strength) Key sizes: 2048 and 3072 bits
NDRNG	Provides seed input to the KMSM Approved DRBG

3 Ports and Interfaces

The KMSM module is a software only module and as such, does not have physical ports. For FIPS 140-2 purposes, Key Nexus KMSM cryptographic 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 computer (general purpose computer [GPC]) on which it resides. These physical ports are separated into the logical interfaces defined by FIPS 140-2, as shown in Table 3, and are mapped to the GPC physical ports in Table 4.

The KMSM module is a software only module, and, therefore, control of the physical ports is outside of the module's scope. The module does provides a set of logical interfaces which are mapped to the following FIPS 140-2 defined logical interfaces: data input, data output, control input, status output, and power.

When the module performs self-tests, is in an error state, is generating keys, or performing zeroization, the module prevents all output on the logical data output interface as only the thread performing the operation has access to the data. If in an error state, the module does not return any output data, only an error value.If the module is performing self-tests, all access to the cryptographic functons is inhibited and no data is output.

The mapping of the FIPS 140-2 logical interfaces to the module is described in table 4.

The "show status" service is allocated to both the CO and the User. The run "self-tests" service is accomplished by launching the module or by power- cycling the module for the "on demand " self-tests. Both the CO and the User can begin the power –on self-tests or the on-demand self-tests.

The FIPS standard requires that a module has a data input interface, a data output interface, control input interface, a status output interface and a power interface. Because the KMSM is a software only module, it relies on the Dell Optiplex to provide the physical interfaces.

Table 5 - Module Interfaces

FIPS 140-2 Interface	Physical Interface
Data Input	API input parameters – plaintext (PT) and/or ciphertext (CT) data
Data Output	API output parameters and return values – PT and/or CT 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/shut down of a process containing the module

3.1 Logical Interface to Physical Interface Mappings

The following table is a mapping of the module logical interfaces to the GPC physical interfaces. The GPC used in this validation is a Dell Optiplex.

Table 6 - Logical to Physical Mapping

Logical Interface	Physical Port/Interface
Data Input	Ethernet port(s), keyboard, mouse
Data Output	Ethernet port(s)
Control Input	Keyboard, mouse
Status Output	Display
Power	Power supply

4 Roles and Services

4.1 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. The module does not support a maintenance role and/or bypass capability. The module does not support authentication.

The Crypto Officer role is responsible for initialization of the module. The CO has the same cryptographic services available to them as the User once the installation of the KMSM module is completed.

The User role is not authenticated to the module.It has all the services available to the CO except installation of the KMSM cryptographic module.

Each role (CO and User) is assumed based on the services available to them. There are no additional roles.

4.2 Authentication Mechanisms

The Key Management Security Module is a Security Level 1 module and it does not claim authentication of users.

4.3 Services

All services implemented by the Module are listed in Table 5 below. In approved only mode no non-approved services are accessible.

Table 7 - Services

Service	Description	CO	User
Initialize Module and self-tests on demand	The JRE will call the static constructor for self-tests on module initialization. The RSA self-test for the wrapper (RSA-2048 with SHA-256) and the HMAC-SHA-256 integrity tests for the library are run.	x	
Show status	A user can call *FipsStatus.IsReady()* at any time to determine if the module is ready. FipsStatus.isReady() will only return true if all the classes providing implementations in the FIPS package have loaded successfully. *CryptoServicesRegistrar.IsInApprovedOnlyMode()* can be called to determine the FIPS mode of operation. *CryptoServicesRegistrar.IsInApprovedOnlyMode()will only return true if the module is operating in FIPS Approved mode.*		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 Generation	Used to generate signatures (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
Key Wrapping	Used to encrypt a key value (RSA, AES and Triple-DES)		x
Key Unwrapping	Used to decrypt a key value (RSA, AES and Triple-DES)		x

5 Physical Security

The Key Management Security Module is a software only module seeking a Security Level 1 FIPS validation. The module is a multichip standalone module. It does not implement physical security mechanisms.

6 Operational Environment

The KMSM module was tested inside a virtual machine on a Dell OptiPlex 980 with Java JRE 1.8. The host OS is Windows 7 Professional 64-bit and the guest OS is Ubuntu 14.04.1. The hypervisor used was VirtualBox 5.2 and the processor is an Intel i7 chip.

The module operates in a modifiable operational environment under the FIPS 140-2 definitions. The module runs on a GPC running the above stated operating systems and configuration. The 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.

KeyNexus distributes the KMSM always operating in FIPS mode. To verify that a module is in the Approved Mode of operation, the user can call a FIPS-approved mode status method (CryptoServicesRegisrar.isInApprovedOnlyMode()).

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.

7 Cryptographic Key Management

The following table indicates the keys/CSPs used by the module's services.

7.1 Cryptographic Keys, Key Components, and CSPs

Table 8 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 8 - Keys/CSPs used by KMSM Services

Service	AES Keys	DRBG Keys	ECDSA Keys	HMAC Keys	RSA Keys	Triple-DES Keys
Initialize Module and self-tests on demand						
Show status						
Zeroize / Power-off	Z	Z	Z	Z	Z	Z
Data Encryption	R					R
Data Decryption	R					R
MAC Calculation	R			R		R
Signature Generation			R		R	
Signature Verification			R		R	
DRBG (SP800-90A) output	G		G	G	G	G
Message Hashing						
Keyed Message Hashing				R		
Key Wrapping/Transport (RSA, AES, Triple-DES)	R				R	R
Key Unwrapping (RSA, AES, Triple-DES)	R				R	R

Table 9 - Keys and Critical Security Parameters (CSPs)

Key/CSP	Description/Usage
AES Encryption Key	[FIPS-197, SP 800-38D, Addendum to SP 800-38A, CKG] AES (128/192/256) encrypt key[8]
AES Decryption Key	[FIPS-197, SP 800-38D, Addendum to SP 800-38A, CKG] AES (128/192/256) decrypt key
AES Authentication Key	[FIPS-197, CKG] AES (128/192/256) CMAC/GMAC key

[8] 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.

AES Wrapping Key	[SP 800-38F, CKG] AES (128/192/256) key wrapping key
DRBG (Hash)	V (440/888 bits) and C (440/888 bits), entropy input (length dependent on security strength)
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)
RSA Signing Key	[FIPS 186-4] RSA (>= 2048) signature generation key
RSA Key Transport Key	[FIPS 186-4] RSA (2048, 3072) key transport (decryption) key
Triple-DES Authentication Key	[SP 800-67, CKG] Triple-DES (128/192) CMAC key
Triple-DES Encryption Key	[SP 800-67, CKG] Triple-DES (192) encryption key
Triple-DES Decryption Key	[SP 800-67, CKG] Triple-DES (128/192) decryption key
Triple-DES Wrapping Key	[SP 800-38F, CKG] Triple-DES (192 bits) key wrapping/unwrapping key, (128 unwrapping only).

Table 10 - Public Keys

Key/CSP	Description/Usage
EC Verification Key	[FIPS 186-4] ECDSA (All NIST defined B, K, and P curves) signature verification key
RSA Key Transport Key	[FIPS 186-4] RSA (2048, 3072) key transport (encryption) key
RSA Verification Key	[FIPS 186-4] RSA (>= 1024) signature verification key

7.2 Storage

Keys are introduced into the module through API calls. The module performs input and output of plaintext keys and CSPs to the controlling application from the logical cryptographic boundary. These keys and CSPs are not output beyond the physical cryptographic boundary. The keys reside in RAM and are not stored in the module. There are no visible intermediate key values.

7.3 Zeroization

Since the KMSM has to be placed in RAM in order to become operational, there are no stored /persistent CSPs or keys. All keys utilized by the module are in RAM.

CSPs created within the KMSM module, or imported by the user will be zeroized by the Java Virtual Machine’s garbage collector before the memory they occupy is made available for re-use, likewise all CSPs can be destroyed by powering off the JVM hosting the module.

It is assumed that the environment hosting the JVM is sufficiently capable that the memory space used by the JVM is reserved for the JVM's use and protected from interference. Java also allows for the zeroisation of fields in an object when it is deallocated using an objects finalize method. The finalize method is run when an object is identified as elligible for garbage collection.

Where possible finalize methods have also been added to objects to zero out fields and values containing information related to CSPs. It should be noted that finalize methods will not be run if an object is not identified for garbage collection however their use, particularly on a busy JVM, does improve the chances of the CSP being existent in memory only for only the minimum time necessary.

7.4 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 DRGB in line with Section 8.2.2 of SP 800-38D. The entropy source used for GCM IV generation is the same entropy source used for the seeding of the Approved DRBG, which meets FIPS 140-2 requirements.

The DRBG seed is generated by the NDRNG implemented in the operating system; which is resides within the same physical cryptographic boundary as the module itself (defined as the GPC enclosure).

The module enforces a minimum GCM IV length of 96-bits.

In approved mode, importing a GCM IV is non-conformant, doing so would cause the module to operate in a non-FIPS conformant manner.

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

8 Electromagnetic Interference / Electromagnetic Compatibility

The KMSM is a software only module and uses the host, the Dell Optiplex 980, FCC certification. The Dell machine is conformant to the FCC EMI/EMC requirements in 47 Code of Federal Regulation, Part 15, Subpart B, Unintentional Radiators, Digital Devices, Class B.

9 Self Tests

9.1 Power Up 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 7 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 Non-recoverable error state and the operator should then contact KeyNexus, Inc. for support. The power up self tests are performed each time the KMSM is launched and put into RAM. They are performed each time the module is powered on in order to verify that the module works correctly and that sensitive data is not exposed or damaged.Until all power up self-tests are executed correctly, no data can be output from the module and no cryptographic services are available.

The module's self-tests are given in Table 11 below.

Table 11 - Self-Tests

Test	Description
Software integrity	• HMAC SHA-256 • RSA signature verification (RSA-2048 with SHA-256)
AES	KATs: Encryption / Decryption, Modes: ECB Key sizes: 128
CCM	KATs: Generation, Verification Key sizes: 128 bits
AES-CMAC	KATs: Generation, Verification Key sizes: AES with 128 bits
DRBG	KATs: HASH_DRBG Security Strengths: 256 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
RSA	KATs: Signature Generation, Signature Verification Key sizes: 2048 bits
SHS[9]	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
Key Wrapping	KATs per IG D.4

The power on self-tests are automatically run upon launch of the KMSM module. The operator does not interfere in the running of the self-tests. No data can be ouput until the self-tests are successfully completed. No cryptographic services are available until the self-tests are completed successfully. If there is a self-test failure, the module will output a detailed error message when FipsStatus.isReady() is called. If one of the KATs fails, the module enters the “Non-recoverable” error state. The error state can only be cleared by reloading the module and calling FipsStatus.isReady() again to confirm successful completion of the KATs. Absence of a self-test error code indicates self-tests have passed, however FipsStatus.isReady() will return a value of “true” if called and all self-tests have passed successfully.

The self-tests can also be run “on demand” by power cycling the module.

9.2 Conditional tests

[9] SHA KATs are implemented as separate tests from HMAC KATs

Conditional tests are run for a specific occurance. For example if a signature keypair is generated, a pairwise consistency test must be run to determine that the algorithms are operating correctly. The conditional tests are given in Table 9 and a brief description of each is also given.

The module performs the conditional self tests found in Table 10.

Table 12 - Conditional tests

Test	Description
NDRNG CRNG test	Continuous test whenever a random number is requested from the NDRNG
DRBG CRNG test	DRBG Continuous Test performed when a random value is requested from the DRBG
ECDSA	Pairwise consistency test on ECC key pair generation
RSA	Pairwise consistency test on RSA key pair generation
DRBG Health checks	Performed conditionally on DRBG, per SP 800-90A Section 11.3. Continual health checks are performed on both the DRBG and the entropy requested from the containing JVM to ensure quality of any key material generated. A module failure will occur if the DRBG is unable to obtain sufficient entropy from the entropy source of the containing JVM in order to satisfy the number of bits of security the DRBG is configured for.

10 Design Assurance

KeyNexus uses GitHub to control the module's source code. Documents such as release notes and API documents are also managed by GitHub. These are normally shipped with the module. Any documentation outside of that is manually versioned. This would include the User manual and any evidence documentation presented for this FIPS validation.

11 Mitigation of Other Attacks

The module claims the mitigation of two attacks: 1) AES timing attack and 2) RSA blinding attack.

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.

12 Secure Operation

The module is distributed by KeyNexus to be always in FIPS mode so after installation, there is no further steps or instructions for the Crypto Officer to place the module into FIPS mode. The KeyNexus component of the module places the cryptographic library into FIPS mode and this is a non-modifiable parameter.

Security Rules and Guidance

12.1 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.

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.

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.

12.2 Crypto Officer Guidance

The Crypto Officer (CO) is responsible for the installation of the module. 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.The following permissions need to be set in order for the module to operate in FIPS mode. The CO may double check these permissions if allowed by KeyNexus but KeyNexus is responsible for setting them.

- java.lang.RuntimePermission "getProtectionDomain"
- org.bouncycastle.crypto.CryptoServicesPermission “changeToApprovedModeEnabled”
- In order to check for configuration properties the policy permission: java.util.PropertyPermission "java.runtime.name", "read" needs to be set.
- java.lang.RuntimePermission "accessDeclaredMembers" allows the use of later than JDK 1.5 classes.
- If the JCA/JCE provider is to be installed during execution, the policy permission: java.security.SecurityPermission "putProviderProperty.BCFIPS"; is also required.
- org.bouncycastle.crypto.CryptoServicesPermission"tlsAlgorithmsEnabled” is equivalent to setting these two permissions together: org.bouncycastle.crypto.CryptoServicesPermission "tlsNullDigestEnabled";
- org.bouncycastle.crypto.CryptoServicesPermission "tlsPKCS15KeyWrapEnabled";
- The following permission is required to set the default secure random using the CryptoServicesRegistrar permission: org.bouncycastle.crypto.CryptoServicesPermission “defaultRandomConfig”

12.3 User Guidance

The assumption for this FIPS module is that the Users are not malicious and are sufficiently trained to use the module.

Further guidance on the use of the module’s cryptographic functions is as follows:

The operator shall ensure that no more than 2^{16} encryptions with the same Triple-DES key are performed in order to be compliant with IG A.13.

The operator shall not use 2-key Triple-DES for any other purpose than generating a MAC for message authentication verification. Doing so would result in operating the module in a non-FIPS conformant manner.

Importing a GCM IV is non-conformant and doing so would result in operating the module in a non-FIPS conformant manner.

13 References and Acronyms

Table 13 - References and Acronym Definitions

Acronym	Definition
FIPS 140-2	Security Requiremets for Cryptographic modules, May 25, 2001
FIPS 180-4	Secure Hash Standard (SHS), August 2015
FIPS 186-4	Digital Signature Standard (DSS), July 2013
FIPS 197	Advanced Encryption Standard, November 26, 2001
FIPS 198-1	The Keyed-Hash Message Authentication Code (HMAC), July 2008
FIPS 202 Message Authentication Code (HMAC)	SHA-3 Standard: Permutation-Based Hash and Extendable-Output, July 7, 2014
SP 800-38A	Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode, October 2010
SP 800-38B	Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, May 2005
SP 800-38C	Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, July 7, 2007
SP 800-38D	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC November 2007,
SP 800-38F	Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, December 2012
SP 800-56A Rev 2	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography,, May 2013
SP 800-56C	Recommendation for Key Derivation through Extraction-then-Expansion, , Novembe 20111
SP 800-67 Rev 2	Recommendation, for the Triple Data Encryption Algorithm (TDEA) Block Cipher, November 2017
SP 800-90A Rev 1	Recommendation for Random Number Generation Using Deterministic Random Bit Ge, June 2015nerators,
SP 800-108	Recommendation for Key Derivation Using Pseudorandom Functions, October 2009
SP 800-132	Recommendation for Password-Based Key Derivation, December 2010
SP 800-135 Rev 1	Recommendation for Existing Application –Specific Key Derivation, December 2011
AES	Advanced Encryption Standard,
API	Application Programming Interface
BC	Bouncy Castle
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CO	Crypto Officer
CSE	Communications Security Establishment
CSP	Critical Security Parameter
D-H	Diffie-Hellman
DRAM	Dynamic Random Access Memory
DRBG	Deterministic Random Bit Generator
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm

Acronym	Definition
EMI/EMC	Electromagnetic Interference / Electromagnetic Compatibility
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standards
GPC	General Purpose Computer
HMAC	(Keyed-) Hash Message Authentication Code, July 16 2008
IG	Implementation Guidance
KAS	Key Agreement Scheme
KAT	Known Answer Test
KDF	Key Derivation Function
KW	Key Wrap
KWP	Key Wrap with Padding
MAC	Message Authentication Code
N/A	Not Applicable
NDRNG	Non-Deterministic Random Number Generator
NIST	National Institute of Standards and Technology
NDRBG	Non-Deterministic Random Bit Generator
NVM	Non-Volatile Memory
OS	Operating System
PKCS	Public Key Cryptography Standards
ROM	Read Only Memory
RSA	Rivest, Shamir, and Adleman
SHA	Secure Hash Algorithm
TCBC	TDEA Cipher-Block Chaining
TCFB	TDEA Cipher Feedback Mode
TDEA	Triple Data Encryption Algorithm
TECB	TDEA Electronic Codebook
TOFB	TDEA Output Feedback
TLS	Transport Layer Security

