Cisco Systems, Inc.

Cisco Systems NSS Module Non-Proprietary Security Policy

Software Versions 3.36, 3.44

Document Version 1.6

Table of Contents

1. Introduction 4
1.1. Purpose 4
2. Cryptographic Module 5
2.1. Cryptographic Module Specification 5
2.2. Validation Level Detail 5
2.3. Modes of Operation 5
3. Module Interfaces 6
3.1. Inhibition of Data Output 7
3.2. Disconnecting the output Data Path from the Key Process 7
4. Key Management 8
4.1. Approved Cryptographic Algorithms 8
4.2. Non-Approved but Allowed Cryptographic Algorithms 8
4.3. Non-Approved Mode of Operation 8
4.4. Critical Security Parameters 10
4.5. Random Number Generation 11
4.6. Key/CSP Storage 11
4.7. Key/CSP Zeroization 11
5. Roles, Services, and Authentication 12
5.1. Roles 12
5.2. Assumption of Roles 12
5.3. Authentication 12
5.4. Services 12
6. Physical Security 17
7. Operational Environment 18
8. EMI/EMC 19
9. Self-Tests 20
9.1. Power-Up Self-Tests 20
9.2. Conditional Self-Tests 20
9.3. Mitigation of Other Attacks 20
10. Security Rules and Guidance 21
10.1. Crypto Officer Guidance 21
10.2. Access to Audit Data 21
10.3. User Guidance 21
10.4. DSA Keys 21

10.5. Triple-DES Keys 21
10.6. AES GCM IV Generation 21
10.7. Handling Self-Test Errors 21
10.8. Basic Enforcement 22
11. Acronyms 23

1. Introduction

1.1. Purpose

This is the non-proprietary cryptographic module security policy for the Cisco Systems NSS Module with software versions 3.36 and 3.44. This security policy describes how this module meets the security requirements of FIPS 140-2 Level 1 and how to run the module in a FIPS 140-2 mode of operation. This Security Policy may be freely distributed.

FIPS 140-2 (Federal Information Processing Standards Publication 140-2 — Security Requirements for Cryptographic Modules) details the U.S. Government requirements for cryptographic modules. More information about the FIPS 140-2 standard and validation program is available on the NIST website at http://csrc.nist.gov/groups/STM/index.html.

2. Cryptographic Module

2.1. Cryptographic Module Specification

Cisco Systems NSS Module (hereafter referred to as the "Module") is a software library supporting FIPS 140-2 approved cryptographic algorithms. The software versions are 3.36 and 3.44. For the purposes of the FIPS 140-2 validation, its embodiment type is defined as multi-chip standalone. The Module is an open-source, general-purpose cryptographic library, with an API based on the industry standard PKCS #11 version 2.20.

The module's logical cryptographic boundary is the shared library of files and their integrity check signature files as listed below:

- libnssdbm3.chk
- libnssdbm3.so
- lib.libnssdbm3.chk
- lib.libnssdbm3.so
- libsoftokn3.chk
- libsoftokn3.so
- libfreeblpriv3.chk
- libfreeblpriv3.so

The module 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.

2.2. Validation Level Detail

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

- Cryptographic Module Specification: **Security Level 1**
- Cryptographic Module Ports and Interfaces: **Security Level 1**
- Roles, Services, and Authentication: **Security Level 1**
- Finite State Model: **Security Level 1**
- Physical Security: **Not applicable**
- Operational Environment: **Security Level 1**
- Cryptographic Key Management: **Security Level 1**
- Electromagnetic Interference / Electromagnetic Compatibility: **Security Level 1**
- Self-Tests: **Security Level 1**
- Design Assurance: **Security Level 1**
- Mitigation of Other Attacks: **Not applicable**

2.3. Modes of Operation

The module supports two modes of operation: Approved and Non-approved. The module will be in FIPS-approved mode when all power up self-tests have completed successfully, and only Approved algorithms are invoked. See below for a list of the supported Approved algorithms and Table 2 for allowed algorithms. The non-Approved mode is entered when a non-Approved algorithm is invoked.

3. Module Interfaces

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

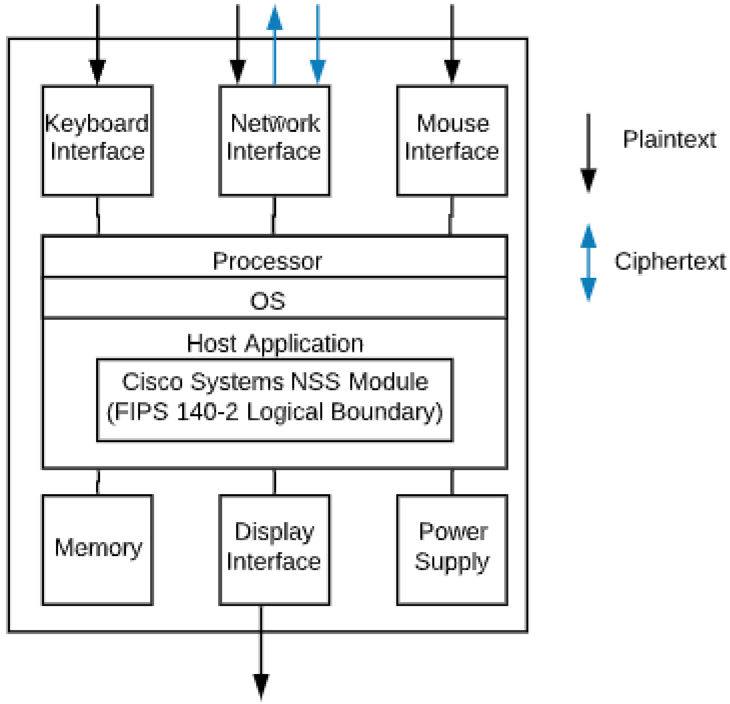

Figure 1 – Module Boundary and Interfaces Diagram

The interfaces (ports) for the physical boundary include the computer keyboard port, mouse port, network port, USB ports, display and power plug. 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) following the PKCS #11 specification, the database files in a kernel file system, the environment variables and configuration files. The logical interfaces expose services that applications directly call, and the API provides functions that may be called by a referencing application. 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 Data Input Interface:
  - Logical Interface: API input parameters – plaintext and/or ciphertext data
  - Physical Interface: Network Interface
- FIPS 140-2 Data Output Interface:
  - Logical Interface: API output parameters and return values – plaintext and/or ciphertext data
  - Physical Interface: Network Interface
- FIPS 140-2 Control Input Interface:
  - Logical Interface: API method calls- method calls, or input parameters, that specify commands and/or control data used to control the operation of the module (/proc/sys/crypto/fips_enabled)
  - Physical Interface: Keyboard Interface, Mouse Interface
- FIPS 140-2 Status Output Interface:
  - Logical Interface: API output parameters and return/error codes that provide status information used to indicate the state of the module
  - Physical Interface: Display Controller, Network Interface
- FIPS 140-2 Power Interface:

- Logical Interface: None
- Physical Interface: Power Supply

As shown in Figure 1 – Module Boundary and Interfaces Diagram and in the list above, 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.

The Module does not use the same buffer for input and output. After the Module is done with an input buffer that holds security-related information, it always zeroizes the buffer so that if the memory is later reused as an output buffer, no sensitive information can be inadvertently leaked.

The logical interfaces of the Module consist of the PKCS #11 (Cryptoki) API. The API itself defines the Module's logical boundary, i.e, all access to the Module is through this API

3.1.Inhibition of Data Output

All data output via the data output interface is inhibited when the NSS module is performing self-tests or in the Error state.

During self-tests: All data output via the data output interface is inhibited while self-tests are executed.

In Error state: The Boolean state variable sftk fatalError tracks whether the NSS module is in the Error state. Most PKCS #11 functions, including all the functions that output data via the data output interface, check the sftk_fatalError state variable and, if it is true, return the CKR_DEVICE_ERROR error code immediately. Only the functions that shut down and restart the module, reinitialize the module, or output status information can be invoked in the Error state.

These functions are FC_GetFunctionList, FC_Initialize, FC_Finalize, FC_Getlnfo, FC_GetSlotList, FC_GetSlotinfo, FC_GetTokeninfo, FC_InitToken, FC_CloseSession, FC_CloseAllSessions, and FC WaitForSlotEvent.

3.2.Disconnecting the output Data Path from the Key Process

During key generation and key zeroization, the Module may perform audit logging, but the audit records do not contain sensitive information. The Module does not return the function output arguments until the key generation or key zeroization is finished. Therefore, the logical paths used by output data exiting the module are logically disconnected from the processes/threads performing key generation and key zeroization.

4. Key Management

4.1. Approved Cryptographic Algorithms

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

Algorithm	Mode/Method	CAVP	Details	Use
AES (FIPS 197 SP 800-38A)	ECB, CBC	C 503	128, 192, 256	Encryption, Decryption
KTS (AES) (SP 800-38F)	AES KW	C 503	128, 192, 256	wrapping and unwrapping key values
KTS (Triple-DES) (SP 800-38F)	TKW	C 503	168	wrapping and unwrapping key values
GCM (SP 800-38D)	AES	C 503	128, 192, 256	Encryption, Decryption, Authentication
DRBG (SP 800-90A)	Hash DRBG	C 503	256	Random Bit Generation The module generates keys whose strengths are modified by available entropy.
DSA (FIPS 186-4)	PQG Verification, Key Pair Generation, Signature Generation, Signature Verification	C 503	1024, 2048, 3072 bits (1024 only for SigVer)	Digital Signature Services
ECDSA (FIPS 186-4)	Key Pair Generation, Signature Verification, Public Key Validation	C 503	P-256, P-384, P- 521,	Digital Signature Services
HMAC (FIPS 198-1)	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	C 503		Generation, Authentication
SHA (FIPS 180-4)	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	C 503		Digital Signature Generation, Digital Signature Verification, non-Digital Signature Applications
Triple-DES **(**SP 800-67)	TECB, TCBC and CTR	C 503	2-key, 3-key	Decryption

Table 1 – FIPS-Approved Algorithm Certificates

4.2. 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.

- Diffie-Hellman Key Agreement: 2048-bit and 15360-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 between 112 and 256 bits of encryption strength)
- Elliptic Curve Diffie-Hellman: using P-256, P-384 and P-521 curves is used for system-level key establishment; not used by the module to establish keys within the module (key agreement; key establishment methodology provides between 128 and 256 bits of encryption strength)
- NDRNG: Used to seed the FIPS approved DRBG

Table 2 – Non-Approved but Allowed Cryptographic Algorithms

4.3. 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.

- AES-GCM (non-compliant): Encryption

- AES CTS: Encryption, Decryption
- Camellia: Encryption, Decryption
- DES: Encryption, Decryption
- DSA (non-compliant): Public Key Cryptography
- ECDSA (non-compliant): Public Key Cryptography
- J-PAKE: Key Agreement
- MD2: Hashing
- MD5: Hashing
- RC2: Encryption, Decryption
- RC4: Encryption, Decryption
- RC5: Encryption, Decryption
- SEED: Encryption, Decryption
- Triple-DES with 2-key: Encryption

4.4.Critical Security Parameters

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

CSP	Generation	Storage	Entry/Output	Zeroization
AES 128, 192, and 256-bit keys	Use of NIST SP800-90A DRBG	Application memory	N/A	Automatically zeroized when freeing the cipher handle
Triple-DES 168 bits keys	Use of NIST SP800-90A DRBG	Application memory	N/A	Automatically zeroized when freeing the cipher handle
AES 128, 192, and 256-bit Wrapping keys	Use of NIST SP800-90A DRBG	Application memory	N/A	Automatically zeroized when freeing the cipher handle
Triple-DES 168 -bit Wrapping keys	Use of NIST SP800-90A DRBG	Application memory	N/A	Automatically zeroized when freeing the cipher handle
DSA 2048- and 3072-bit private keys	Use of NIST SP800-90A DRBG in DSA key generation mechanism	Application memory	N/A	Automatically zeroized when freeing the cipher handle
ECDSA private keys based on P-256, P-384 and P-521	N/A (supplied by the calling application)	Application memory	N/A	Automatically zeroized when freeing the cipher handle
HMAC keys with at least 112 bits	Use of NIST SP800-90A DRBG	Application memory	N/A	Automatically zeroized when freeing the cipher handle
DRBG entropy input string and seed	Provided by the host platform	Application memory	N/A	Automatically zeroized when freeing the DRBG handle
DRBG V and C values	Derived from the entropy input string as defined in NIST SP800-90A	Application memory	N/A	Automatically zeroized when freeing the DRBG handle
Diffie-Hellman private components with size between 2048 bits and 15360 bits	Use of NIST SP800-90ADRBG in Diffie-Hellman key agreement scheme	Application memory	N/A	Automatically zeroized when freeing the cipher handle
EC Diffie-Hellman private components based on P-256, P-384 and P-521 curves	Use of NIST SP800-90A DRBG in EC Diffie-Hellman key agreement scheme	Application memory	N/A	Automatically zeroized when freeing the cipher handle

Table 3 – Critical Security Parameters

4.5.Random Number Generation

The Module employs a NIST SP800-90A Hash_DRBG with SHA-256 as a random number generator. The random number generator and NDRNG is seeded by obtaining random data from the Intel RDRAND implementation available on the platform hardware. The entropy source provides at least 880 bits of random data available to the Module. The module generates keys whose strengths are modified by available entropy. After 2^{48} calls to the random number generator the Module reseeds automatically. The Module performs DRBG health testing as specified in section 11.3 of NIST SP800-90A. The Module provides at least 256 bits of entropy.

4.6.Key/CSP Storage

The Module supports cryptographic keys and CSPs in the FIPS Approved mode operation as listed in Table 3 – Critical Security Parameters. The module does not perform persistent storage for any keys or CSPs.

4.7.Key/CSP Zeroization

The application that used the Module is responsible for appropriate zeroization of the key material. The Module provides zeroization methods to clear the memory region previously occupied by a plaintext secret key or private key. A plaintext secret and private keys must be zeroized when it is passed to a FC_DestroyObject call. All plaintext secret and private keys must be zeroized when the Module is shut down (with a FC_Finalize call), reinitialized (with a FC_InitToken call), or when the session is closed (with a FC_Close Session or FC_CloseAllSessions call). All zeroization is to be performed by storing the value 0 into every byte of memory region that is previously occupied by a plaintext secret key or private key.

Zeroization is performed in a time that is not sufficient to compromise plaintext secret or private keys.

5. Roles, Services, and Authentication

5.1.Roles

The module supports two distinct operator roles, 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.

- Crypto Officer – Installs and initializes the module. The CO can access other general-purpose services (and status services of the Module. The CO does have access to any service that utilizes the secret and private keys of the Module. The CO must control the access to the Module both before and after installation, including the management of physical access to the computer, executing the Module code as well as management of the security facilities provided by the operating system.
- User – The User role had access to all cryptography secure services which use the secret or private keys of the Module. It is also responsible for the retrieval, updating and deletion of keys from the private key database.

5.2.Assumption of Roles

The User and Crypto Officer roles are implicitly assumed by the entity accessing the module services.

5.3.Authentication

The module is a Level 1 software-only cryptographic module and does not implement authentication. The role is implicitly assumed based on the service requested.

5.4.Services

The Module has a set of API functions denoted by FC_xxx. All the API functions are listed in Table 4 – Module Services.

Among the module's API functions, only FC_GetFunctionlist is exported and therefore callable by its name. All the other API functions must be called via the function pointers returned by

FC_GetFunctionlist. It returns a CK_FUNCTION_LIST structure containing function pointers named C_xxx such as (_Initialize and (_Finalize. The C_xxx function pointers in the CK_FUNCTION_LIST structure returned by FC_GetFunctionlist point to the FC_xxx functions .

The following convention is used to describe API function calls. Here FC_Initialize is used as

examples:

- When "call FC_Initialize" is mentioned, the technical equivalent of "call the FC_Initialize function via the (_Initialize function pointer in the CK_FUNCTION_LIST structure returned by FC_GetFunctionlist" is implied.

All services implemented by the module are listed in Table 4 – Module Services. The third column provides a description of each service and availability to the Crypto Officer and User, in columns 4 and 5, respectively.

Service	Role	Function	Description
Get the Function List	CO	FC_GetFunctionList	Return a pointer to the list of function pointers for the operational mode
Module Initialization	CO	FC_InitToken FC_InitPIN	Initialize or re-initialize a token Initialize the user’s password, i.e., set the user’s initial password
General Purpose	CO	FC_Initialize FC_Finalize FC_GetInfo	Initialize the module library Finalize (shut down) the module library Obtain general information about the module library
Slot and Token Management	CO	FC_GetSLotList FC_GetSlotInfo FC_GetTokenInfo	Obtain a list of slots in the system Obtain information about a particular slot Obtain information about the token (This function provides the Show Status service)

Service	**Role**	**Function**	**Description**
		FC_GetMechanismList FC_GetMechanismInfo	Obtain a list of mechanisms (cryptographic algorithms) supported by a token Obtain information about a particular mechanism
Session Management	CO	FC_OpenSession FC_CloseSession FC_CloseAllSessions FC_GetSessionInfo FC_GetOperationState FC_SetOperationState	Open a connection (session) between an application and a particular token Close a session Close all sessions with a token Obtain information the session (This function provides the Show Status service) Save the state of the cryptographic operations in a session (This function is only implemented for message digest operations) Restore the state of the cryptographic operations in a session (This function is only implemented for message digest operations
Object Management	CO	FC_CreateObject FC_CopyObject FC_DestroyObject FC_GetObjectSize FC_GetAttributeValue FC_SetAttributeValue FC_FindObjectsInit FC_FindObjects FC_FindObjectsFinal	Creat a new object Create a copy of an object Destroy an object Obtain the size of an object in bytes Obtain an attribute value of an object Modify an attribute value of an object Initialize an object search operation Continue an object search operation Finish an object search operation
Encryption and Decryption	User	FC_EncryptInit FC_Encrypt FC_EncryptUpdate FC_EncryptFinal FC_DecryptInit FC_Decrypt FC_DecryptUpdate FC_DecryptFinal	Initialize an encryption operation Encrypt single-part data Continue a multiple-part encryption operation Finish a multiple-part encryption operation Initialize a decryption operation Decrypt single-part encrypted data Continue a multiple-part decryption operation Finish a multiple-part decryption operation
Message Digest	CO	FC_DigestInit FC_Digest FC_DigestUpdate FC_DigestFinal	Initialize a message-digesting operation Digest single-part data Continue a multiple-part digesting operation Finish a multiple-part digesting operation
	User	FC_DigestKey	Continue a multiple-part message-digestion operation by digesting the value of a secret key as part of the data already digested
Signature Generation and Verification	User	FC_SignInit FC_Sign FC_SignUpdate FC_SignFinal FC_SignRecoverInit FC_SignRecover FC_VerifyInit FC_Verify	Initialize a signature operation Sign single part data Continue a multiple-part signature operation Finish a multiple-part signature operation Initialize a signature operation, where the data can be recovered from the signature Sign single-part data, where the data can be recovered from the signature Initialize a verification operation Verify a signature on single-part data Continue a multiple-part verification operation Finish a multiple-part verification operation

Service	Role	Function	Description
		FC_VerifyUpdate FC_VerifyFinal FC_VerifyRecoverInit FC_VerifyRecover	Initialize a verification operation, where the data is recovered from the signature Verify a signature on single-part data, where the data is recovered from the signature
Dual-function Cryptographic Operations	User	FC_DigestEncryptUPdate FC_DecryptDigestUpdate FC_SignEncryptUpdate FC_DecryptVerifyUpdate	Continue a multiple-part digesting and encryption operation Continue a multiple-part decryption and digesting operation Continue a multiple-part signing and encryption operation Continue a multiple-part decryption and verify operation
Key Management	User	FC_GenerateKey FC_GenerateKeyPair FC_WrapKey FC_UnwrapKey	Generate a secret key Generate a public/private key pair (This function performs the pair-wise consistency tests) Used to wrap (encrypt) a key Used to unwrap (decrypt) a key
Random Number Generation	CO	FC_SeedRandom FC_GenerateRandom	Mix in additional seed material to the random number generator Generate random data (This function performs the continuous random generator test)
Self Tests	CO	N/A	The self tests are performed automatically when loading the module
Zeroization	CO	FC_InitToken FC_Finalize FC_CloseSession FC_CloseAllSessions	All CSPs are automatically zeroized when freeing the cipher handle
	User	FC_DestroyObjects	

Table 4 – Module Services

Table 9 lists all the services available in non-Approved mode with API function and the non-Approved algorithm that the function may invoke. Please note that the functions are the same as the ones listed in Table 8, but the underneath non-Approved algorithms are invoked. If any service invokes the non-Approved algorithms, then the module will enter non-Approved mode implicitly.

Service	Function	Non-Approved Algorithm Invoked
Encryption and Decryption	FC_EncryptInit FC_Encrypt FC_EncryptUpdate FC_EncryptFinal	AES GCM mode, AES CTS mode, Camellia, DES, RC2, RC4, RC5, SEED, Two-key Triple-DES
	FC_DecryptInit FC_Decrypt FC_DecryptUpdate FC_DecryptFinal	AES CTS mode, Camellia, DES, RC2, RC4, RC5, SEED
Message Digest	FC_DigestInit FC_DigestUpdate	MD2, MD5

Service	Function	Non-Approved Algorithm Invoked
	FC_DigestKey FC_DigestFinal	
Signature Generation and Verification	FC_SignInit FC_Sign FC_SignUpdate FC_SignFinal FC_SignRecoverInit FC_SignRecover FC_VerifyInit FC_Verify FC_VerificyUpdate FC_VerifyFinal FC_VerifyRecoverInit FC_VerifyRecover	DSA signature generation with non-compliant key size DSA signature verification with non-compliant key size
Dual-function Cryptographic Operations	FC_DigestEncryptUpdate	MD2, MD5, AES GCM mode, AES CTS mode, Camellia, DES, RC2, RC4, RC5, SEED, Two-key Triple-DES
	FC_DecryptDigestUpdate	AES STS mode, Camellia, DES RC2, RC4, RC5, SEED, MD2, MD5
	FC_SignEncryptUpdate	DSA signature generation with non-compliant key size, AES GCM mode, AES CTS mode, Camellia, DES, RC2, RC4, RC5, SEED, Two-key Triple-DES
	FC_DecryptVerifyUpdate	AES CTS mode, Camellia, DES, RC2, RC4, RC5, SEED, DSA signature verification with non-compliant key size
Key Management	FC_GenerationKeyPair	DSA domain parameter generation, DSA domain parameter verification with non-compliant key size, DSA key pair generation with non-compliant key size
	FC_KeyWrapKey	AES key wrapping (encrypt) based on NIST SP800-38F, Triple-DES key wrapping (encrypt) using Two-key Triple-DES
	FC_UnwrapKey	AES key wrapping (encrypt) based on NIST SP800-38F, Triple-DES key wrapping (decrypt) using Two-key Triple-DES
	FC_DeriveKey	Diffie-Hellman key agreement with non-compliant key size, J-PAKE key agreement

Table 5 – Services in Non-Approved Mode

Table 6 – 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.

Service	**AES keys**	**Triple-DES bit keys**	**AES Wrapping keys**	**Triple-DES Wrapping keys**	**DSA private keys**	**ECDSA private keys**	**HMAC Keys**	**DRBG entropy input string and seed**	**DRBG V and C values**	**Diffie-Hellman private components**	**EC Diffie-Hellman private components**
Get the Function List	-	-	-	-	-	-	-	-	-	-	-
Module Initialization	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
General Purpose	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Slot and Token Management	-	-	-	-	-	-	-	-	-	-	-
Session Management	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Object Management	RWZ	RWZ	RWZ	RWZ	RWZ	RWZ	RWZ	RWZ	RWZ	RWZ	RWZ
Encryption and Decryption	R	R	R	R	-	-	-	-	-	-	-
Message Digest	-	-	-	-	-	-	R	-	-	-	-
Signature Generation and Verification	-	-	-	-	R	R	R	-	-	-	-
Dual-function Cryptographic Operations	R	R	R	R	R	R	R	-	-	-	-
Key Management	W	W	W	W	-	-	W	-	-	W	W
Random Number Generation	-	-	-	-	-	-	-	RW	RW	-	-
Parallel Function Management	-	-	-	-	-	-	-	-	-	-	-
Self Tests	-	-	-	-	R	-	-	-	-	-	-
Zeroization	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z

Table 6 – CSP Access Rights within Services

6. Physical Security

The Module is comprised of software only and thus does not claim any physical security.

7. 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.

The module was tested on the following platforms:

Hardware Platform	Processor	Operating System
Cisco UCS M4	Intel Xeon E5-2600	CentOS Linux 7.4
Cisco UCS M5	Intel Xeon Bronze	CentOS Linux 7.4

Table 7 – FIPS Tested Configurations

8. EMI/EMC

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, Class A. FIPS 140-2 validation compliance is maintained when the module is operated on other versions of the GPOS running in single user mode, assuming that the requirements outlined in NIST IG G.5 are met.

9. 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 below. All KATs must be completed successfully prior to any other use of cryptography by the module. If one of the KATs fails, the module enters the Self-Test Failure error state. The Module returns the error code CKR_DEVICE_ERROR to the calling application to indicate the Error state. The Module needs to be reinitialized in order to recover from Error state.

9.1.Power-Up Self-Tests

- AES KATs for ECB and CB modes: encryption and decryption are tested separately
- Triple-DES KATs for ECB and CBC modes: encryption and decryption are tested separately
- DSA KAT: signature generation and verification are tested separately
- ECDSA KAT: signature generation and verification are tested separately
- SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 KAT
- HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384 and HMAC-SHA-512 KAT
- NIST SP800-90A Hash DRBG KAT
- Module Integrity DSA signature verification with 2048 bit key and SHA-256

9.2.Conditional Self-Tests

The following lists of Pairwise Consistency Tests (PCT) and Continuous Random Number Generator Test (CRNGT) as the conditional self-tests. If any of the conditional test fails, the Module enters the Error sate. It returns the error code CKR_DEVICE_ERROR to the calling application to indicate the Error sate. The Module needs to be reinitialized in order to recover from the Error sate.

- DSA PCT for DSA key generation
- ECDSA PCT for ECDSA key generation
- NIST SP800-90A DRBG CRNGT
- Entropy CRNGT
- SP 800-90A Health Tests tested as required by [SP800-90] Section 11

9.3.Mitigation of Other Attacks

The module does not claim to mitigate other attacks beyond those defined in FIPS 140-2.

10.Security Rules and Guidance

10.1. Crypto Officer Guidance

Crypto Officers use the Installation instructions to install the Module in their environment.
To bring the Module into FIPS approved mode, perform the following:

- Install the dracut-fips package: install dracut-fips
- Recreate the INITRAMFS image: dracut -f

After regenerating the initramfs, the Crypto Officer has to append the following string to the kernel command line by changing the setting in the boot loader: fips=1

10.2. Access to Audit Data

The Module uses the syslog function to audit events, so that audit data are stored in the system log. Only the root user can modify the system log. The system log is usually under /var/log directory. The exact location of the system log is specified in the /ext/syslog.conf file. The Module uses the default user facility and the info, warning, and err severity levels for its log messages.

10.3. User Guidance

The Module must be operated in FIPS Approved mode to ensure that FIPS 140-2 validated cryptographic algorithms and security functions are used.

The following module initialization steps must be followed by the Crypto-Officer before starting to use the NSS module:

- Set the environment variables NSS_ENABLE_AUDIT to 1 before using the Module with an application
- Use the application to get the function pointer list using the API “FC_GetFunctionList”.
- Use the API FC_Initialize to initialize the module and ensure that it returns CKR_OK. A return code other than CKR_OK means the Module is not initialized correctly, and in that case, the module must be reset and initialized again.

All Cryptographic keys used in the FIPS Approved mode of operation must be generated in the FIPS Approved mode or imported while running in the FIPS Approved mode.

10.4. DSA Keys

The Module allows the use of 1024 bits DSA keys for legacy purposes including signature generation, which is disallowed to be used in FIPS Approved mode as per NIST SP800-131A. Therefore, the cryptographic operations with the non-approved key sizes will result in the module operating in non-Approved mode implicitly.

10.5. Triple-DES Keys

In accordance with CMVP IG A.13, when operating in a FIPS approved mode of operation, the same Triple-DES key shall not be used to encrypt more than 2^16 64-bit data blocks.

The user is responsible for ensuring that the module limits the number of encrypted blocks with the same key to no more than 2^16.

10.6. AES GCM IV Generation

The module’s AES-GCM implementation conforms to IG A.5, scenario #3, when operating in a FIPS approved mode of operation, AES GCM, IVs are generated both internally and deterministically and are a minimum of 96-bits in length as specified in SP 800-38D, Section 8.2.1.

In the event that the Module power is lost and restored, a new key for use with the AES GCM encryption/decryption shall be established. Additionally, a human operator shall reset the IV to the last one used.

10.7. Handling Self-Test Errors

When the Module enters the Error state, it needs to be reinitialized to resume normal operation. Reinitialization is accomplished by calling FC_Finalize followed by FC_Initialize.

10.8. 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.

- The module provides two distinct operator roles: User and Cryptographic Officer.
- The module does not provide authentication.
- The operator may command the module to perform the power up self-tests by cycling power or resetting the module.
- Power-up self-tests do not require any operator action.
- Data output is inhibited during key generation, self-tests, zeroization, and error states.
- Status information does not contain CSPs or sensitive data that if misused could lead to a compromise of the module.
- There are no restrictions on which keys or CSPs are zeroized by the zeroization service.
- The module does not support concurrent operators.
- The module does not have any external input/output devices used for entry/output of data.
- The module does not enter or output plaintext CSPs from the module’s physical boundary.
- The module does not output intermediate key values.

11. Acronyms

The following list defines acronyms found in this document:

- AES: Advanced Encryption Standard
- API: Application Programming Interface
- CAVP: Cryptographic Algorithm Validation Program
- CBC: Cipher-Block Chaining
- CMVP: Cryptographic Module Validation Program
- CO: Crypto Officer
- CPU: Central Processing Unit
- CSP: Critical Security Parameter
- CTR: Counter-mode
- DES: Data Encryption Standard
- DRBG: Deterministic Random Bit Generator
- DSA: Digital Signature Algorithm
- 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
- GCM: Galois/Counter Mode
- GMAC: Galois Message Authentication Code
- GPC: General Purpose Computer
- HMAC: (Keyed-) Hash Message Authentication Code
- IV: Initialization Vector
- KAS: Key Agreement Scheme
- KAT: Known Answer Test
- MAC: Message Authentication Code
- MD5: Message Digest algorithm MD5
- N/A: Non Applicable
- NDRNG: Non Deterministic Random Number Generator
- NIST: National Institute of Science and Technology
- OCB: Offset Codebook Mode
- OFB: Output Feedback
- OS: Operating System
- PKCS: Public-Key Cryptography Standards
- 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
- USB: Universal Serial Bus

--- End of Document ---