NetApp Cryptographic Security Module

Module Version 1.0

FIPS 140-2 Level 1 Non-Proprietary Security Policy

Document Version: 1.6

Last Updated: January 20, 2021

Table of Contents

1 Introduction

This document is the non-proprietary Cryptographic Module Security Policy for the NetApp Cryptographic Security Module (NCSM). This security policy describes how the NCSM (Software Version: 1.0) meets the security requirements of FIPS 140-2, and how to operate it in a secure FIPS 140-2 mode. This policy was prepared as part of the Level 1 FIPS 140-2 validation of the NetApp Cryptographic Security Module.

This document provides an overview of the Cryptographic Security Module and explains the secure configuration and operation of the module. With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Validation Submission Documentation is NetApp-proprietary and is releasable only under appropriate non- disclosure agreements. For access to these documents, please contact NetApp Inc.

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 Description

The NetApp Cryptographic Security Module is a software library that provides cryptographic services to a vast array of NetApp's storage and networking products.

The module provides FIPS 140-2 validated cryptographic algorithms for services such as IPSEC, SRTP, SSH, TLS, 802.1x, etc. The module does not directly implement any of these protocols, instead it provides the cryptographic primitives and functions to allow a developer to implement the various protocols.

In this document, the NetApp Cryptographic Security Module is referred to as NCSM, the library, or the module.

2.1 Module Specification

The module is a multi-chip standalone cryptographic module. For the purposes of the FIPS 140-2 level 1 validation, the NCSM is a single object module file named fipscanister.o. The object code in the object module file is incorporated into the runtime executable application at the time the binary executable is generated. The Module performs no communications other than with the consuming application (the process that invokes the Module services via the Module's API).

2.2 Module Block Diagram

The module's logical block diagram is shown in Figure 1 below. The dashed red border denotes the logical cryptographic boundary of the module. The physical cryptographic boundary of the module is the enclosure of the system on which it is executing and is denoted by the solid red boundary.

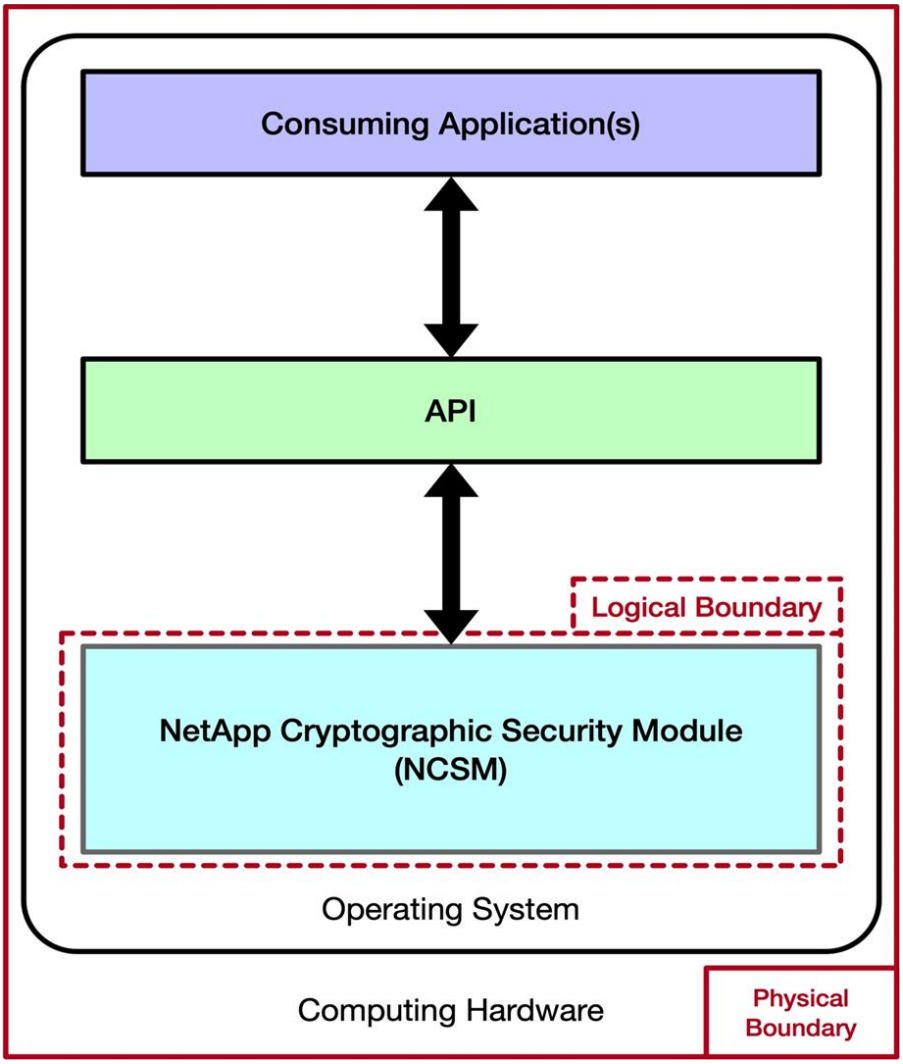

Figure 1 – NCSM block diagram

2.3 Validation Level

The following table lists the level of validation for each area in the FIPS PUB 140-2.

No.	Area Title	Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1

No.	Area Title	Level
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	Electromagnetic Interface/Electromagnetic Compatibility	1
9	Self-Tests	1
10	Design Assurance	3
11	Mitigation of Other Attacks	N/A
	Overall module validation level	**1**

Table 1 – Module Validation Level

2.4 Tested Platforms

This module was tested on the following platforms for the purposes of this FIPS validation:

#	Platform	Operating System	Processor
1	Fujitsu RX200S5	Scientific Linus 6.1	Intel Xeon E5-2609V4 (Broadwell) with PAA
2	Fujitsu RX300-S6	Scientific Linus 6.1	Intel Xeon E5645 (Westmere EP) without PAA
3	Fujitsu RX200S5	Debian Linux 8	Intel Xeon E5-2609V4 (Broadwell) with PAA
4	Fujitsu RX300-S6	Debian Linux 8	Intel Xeon E5645 (Westmere EP) without PAA
5	Fujitsu RX200S5	FreeBSD 9.1	Intel Xeon E5-2609V4 (Broadwell) with PAA
6	Fujitsu RX300-S6	FreeBSD 9.1	Intel XeonE5645 (Westmere EP) without PAA
7	Fujitsu RX200S5	SUSE Linux 11	Intel Xeon E5-2609V4 (Broadwell) with PAA
8	Fujitsu RX300-S6	SUSE Linux 11	Intel Xeon E5645 (Westmere EP) without PAA
9	NetApp AFF A800	ONTAP 9.7P6	Intel Xeon Platinum 8160 (Sky Lake-SP) with PAA

#	Platform	Operating System	Processor
			Intel Xeon Platinum 8160 (Sky Lake-SP) without PAA
10	NetApp FAS2650	ONTAP 9.7P6	Intel Xeon D-1528 (Broadwell) with PAA Intel Xeon D-1528 (Broadwell) without PAA
11	NetApp FAS8300	ONTAP 9.7P6	Intel Xeon Silver 4210 (Cascade Lake) with PAA Intel Xeon Silver 4210 (Cascade Lake) without PAA
12	NetApp FAS9000	ONTAP 9.7P6	Intel Xeon E5-2697 (Broadwell) with PAA Intel Xeon E5-2697 (Broadwell) without PAA

Table 2 – Tested Operational Environments (OEs)

3 Cryptographic Module Ports and Interfaces

The physical ports of the Module are the same as the system on which it is executing. The logical interface is a C-language application program interface (API).

The Data Input interface consists of the input parameters of the API functions. The Data Output interface consists of the output parameters of the API functions. The Control Input interface consists of the actual API functions. The Status Output interface includes the return values of the API functions.

The module provides a number of physical and logical interfaces to the device, and the physical interfaces provided by the module are mapped to the following FIPS 140-2 defined logical interfaces: data input, data output, control input, status output, and power. The logical interfaces and their mapping are described in the following table:

Interface	Description
Data Input	API input parameters - plaintext and/or ciphertext data
Data Output	API output parameters - plaintext and/or ciphertext data
Control Input	API function calls - function calls, or input arguments that specify commands and control data used to control the operation of the module
Status Output	API return codes- function return codes, error codes, or output arguments that receive status information used to indicate the status of

Table 3 – FIPS 140-2 Logical Interfaces

4 Roles, Services and Authentication

4.1 Roles

The Module meets all FIPS 140-2 level 1 requirements for Roles and Services, implementing both Crypto-User and Crypto-Officer roles. The User and Crypto Officer roles are implicitly assumed by the entity accessing services implemented by the Module. The Crypto Officer can install and initialize the Module. The Crypto Officer role is implicitly entered when installing the Module or performing system administration functions on the host operating system.

- User Role: Loading the Module and calling any of the API functions. This role has access to all of the services provided by the Module.
- Crypto-Officer Role: All of the User Role functionality as well as installation of the Module on the host computer system. This role is assumed implicitly when the system administrator installs the Module library file.

4.2 Services

Service	Role	CSP	Access
Module Installation	Crypto Officer	None	N/A
Symmetric encryption/decryption	User, Crypto Officer	Symmetric keys AES, Triple- DES	Execute
Symmetric Digest	User, Crypto Officer	AES CMAC key	Execute
Key transport	User, Crypto Officer	Asymmetric private key RSA	Execute
Key agreement	User, Crypto Officer	DH and ECDH private key	Execute
Digital signature	User, Crypto Officer	Asymmetric private key RSA, DSA, ECDSA	Execute
Key Generation (Asymmetric)	User, Crypto Officer	Asymmetric keys RSA, DSA, and ECDSA	Write/execute
Key Generation (Symmetric)	User, Crypto Officer	Symmetric keys AES, Triple- DES	Write/execute
Keyed Hash (HMAC)	User, Crypto Officer	HMAC key	Execute
Message digest (SHS)	User, Crypto Officer	None	N/A
Random Number Generation	User, Crypto Officer	Seed/entropy input, C, and S	Write/execute
Show status	User, Crypto Officer	None	N/A
Module initialization	User, Crypto Officer	None	N/A
Perform Self-test	User, Crypto Officer	None	N/A
Zeroization	User, Crypto Officer	All CSPs	N/A

Table 4 – Roles, Services, and Keys

4.3 Authentication

As allowed by FIPS 140-2, the Module does not support user authentication for the provided roles. Only one role may be active at a time and the Module does not allow concurrent operators.

5 Physical Security

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

6 Operational Environment

The Module operates in a modifiable operational environment.

The tested operating systems segregate user processes into separate process spaces. Each process space is an independent virtual memory area that is logically separated from all other processes by the operating system software and hardware. The Module functions entirely within the process space of the process that invokes it, and thus satisfies the FIPS 140-2 requirement for a single user mode of operation.

7 Cryptographic Key Management

7.1 Cryptographic Algorithms

The module implements a variety of approved and non-approved algorithms.

7.1.1 Approved Cryptographic Algorithms

The module supports the following FIPS 140-2 approved algorithm implementations:

Algorithm	Supported Modes	Algorithm Certificate Numbers
AES	**ECB** (e/d; 128, 192, 256); **CBC** (e/d; 128, 192, 256); **CFB8** (e/d; 128, 192, 256); **CFB128** (e/d; 128, 192, 256); **OFB** (e/d; 128, 192, 256); **CTR** (ext only; 128, 192, 256) **CCM (KS: 128, 192, 256) (Assoc. Data Len Range**: 0 - 0, 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 7 8 9 10 11 12 13 **(Tag Length(s)**: 4 6 8 10 12 14 16) **CMAC** (Generation/Verification) **(KS: 128;** Block Size(s): Full / Partial; **Msg Len(s)** Min: 0 Max: 2^16; **Tag Len(s)** Min: 2 Max: 16) **(KS: 192**; Block Size(s): Full / Partial; **Msg Len(s)** Min: 0 Max: 2^16; **Tag Len(s)** Min: 2 Max: 16) **(KS: 256**; Block Size(s): Full / Partial; **Msg Len(s)** Min: 0 Max: 2^16; **Tag Len(s)** Min: 2 Max: 16) **GCM (KS: AES_128**(e/d) Tag Length(s): 128 120 112 104 96 64 32) **(KS: AES_192**(e/d) Tag Length(s): 128 120 112 104 96 64 32) **(KS: AES_256**(e/d) Tag Length(s): 128 120 112 104 96 64 32) **PT Lengths Tested:** (0, 512, 1024, 504, 1016); **AAD Lengths tested:** (0, 512, 1024, 504, 1016); **IV Lengths Tested:** (256, 1024); **96BitIV_Supported**; **OtherIVLen_Supported GMAC_Supported** **XTS((KS: XTS_128** ((e/d) (f/p)) **KS: XTS_256**((e/d) (f/p))	3593 A950
Triple-DES	**TECB**(KO 1 e/d, KO 2 d only); **TCBC**(KO 1 e/d, KO 2 d only); **TCFB1**(KO 1 e/d, KO 2 d only); **TCFB8**(KO 1	2000

Algorithm	Supported Modes	Algorithm Certificate Numbers
	e/d, KO 2 d only);**TCFB64**(KO 1 e/d, KO 2 d only); **TOFB**(KO 1 e/d, KO 2 d only) **CMAC**((**KS: 3-Key**; Generation/Verification; Block Size(s): Full / Partial; Msg Len(s) Min: 0 Max: 2^16; Tag Len(s) Min: 2 Max: 8))	A950
SHS	**SHA-1** (BYTE-only) **SHA-224** (BYTE-only) **SHA-256** (BYTE-only) **SHA-384** (BYTE-only) **SHA-512** (BYTE-only)	2955 A950
HMAC	**HMAC-SHA1 (Key Sizes Ranges Tested:** KS<BS KS=BS KS>BS) **HMAC-SHA224 (Key Size Ranges Tested:** KS<BS KS=BS KS>BS) **HMAC-SHA256 (Key Size Ranges Tested:** KS<BS KS=BS KS>BS) **HMAC-SHA384 (Key Size Ranges Tested:** KS<BS KS=BS KS>BS) **HMAC-SHA512 (Key Size Ranges Tested:** KS<BS KS=BS KS>BS)	2290 A950
DRBG	**Hash_Based DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled] **HMAC_Based DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled] **CTR_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; BlockCipher_Use_df]	928 A950
RSA	**FIPS186-4:** **186-4KEY(gen):** FIPS186-4_Random_e **PGM(ProbPrimeCondition):** 2048, 3072**PPTT:**(C.3) **ALG[ANSIX9.31]** Sig(Gen): (2048 SHA(256, 384, 512)) (3072 SHA(256, 384, 512)) Sig(Ver): (1024 SHA(1, 256, 384, 512)) (2048 SHA(1, 256, 384, 512)) (3072 SHA(1, 256, 384, 512)) **ALG[RSASSA-PKCS1_V1_5]** SIG(gen) (2048 SHA (224, 256, 384, 512)) (3072 SHA(224, 256, 384, 512)) SIG(Ver) (1024 SHA(1, 224, 256, 384, 512)) (2048 SHA(1, 224, 256, 384, 512)) (3072 SHA(1, 224, 256, 384, 512)) **[RSASSA-PSS]:** Sig(Gen): (2048 SHA(224 SaltLen(16),	1847 A950

Algorithm	Supported Modes	Algorithm Certificate Numbers
	256 SaltLen(16), 384 SaltLen(16), 512 SaltLen(16))) (3072 SHA(224 SaltLen(16), 256 SaltLen(16), 384 SaltLen(16), 512 SaltLen(16))) Sig(Ver): (1024 SHA(1 SaltLen(16), 224 SaltLen(16), 256 SaltLen(16), 384 SaltLen(16), 512 SaltLen(16))) (2048 SHA(1 SaltLen(16), 224 SaltLen(16), 256 SaltLen(16), 384 SaltLen(16), 512 SaltLen(16))) (3072 SHA(1 SaltLen(16), 224 SaltLen(16), 256 SaltLen(16), 384 SaltLen(16), 512 SaltLen(16)))	
DSA	**FIPS186-4:** **PQG(gen)PARMS TESTED:** [(2048, 224)SHA(224, 256, 384, 512); (2048,256)SHA(256, 384, 512); (3072,256) SHA(256, 384, 512)] **PQG(ver)PARMS TESTED:** [(1024,160) SHA(1, 224, 256, 384, 512); (2048,256) SHA(256, 384, 512); (3072,256) SHA(256, 384, 512)] **KeyPairGen:** [(2048,224); (2048,256); (3072,256)] **SIG(gen)PARMS TESTED:** [(2048,224) SHA(224, 256, 384, 512); (2048,256) SHA(256, 384, 512); (3072,256) SHA(256, 384, 512);] **SIG(ver)PARMS TESTED:** [(1024,160) SHA(1, 224, 256, 384, 512); (2048,224) SHA(1, 224, 256, 384, 512); (2048,256) SHA(1, 224, 256, 384, 512); (3072,256) SHA(1, 224, 256, 384, 512)]	998 A950
ECDSA	**FIPS186-4:** **PKG: CURVES**(P-224 P-256 P-384 P-521 K-233 K-283 K-409 K-571 B-233 B-283 B-409 B-571 ExtraRandomBits TestingCandidates) **PKV: CURVES**(P-224 P-256 P-384 P-521 K-233 K-283 K-409 K-571 B-233 B-283 B-409 B-571) **SigGen: CURVES**(P-224: (SHA-224, 256, 384, 512) P-256: (SHA-224, 256, 384, 512) P-384: (SHA-224, 256, 384, 512) P-521: (SHA-224, 256, 384, 512) K-233: (SHA-224, 256, 384, 512) K-283: (SHA-224, 256, 384, 512) K-409: (SHA-224, 256, 384, 512) K-571: (SHA-224, 256, 384, 512) B-233: (SHA-224, 256, 384, 512) B-283: (SHA-224, 256, 384, 512) B-409: (SHA-224, 256, 384, 512) B-571: (SHA-224, 256, 384, 512)) **SigVer: CURVES**(P-192: (SHA-1, 224, 256, 384, 512) P-224: (SHA-1, 224, 256, 384, 512) P-256: (SHA-1, 224, 256, 384, 512) P-384: (SHA-1, 224, 256, 384, 512) P-	732 A950

Algorithm	Supported Modes	Algorithm Certificate Numbers
	521: (SHA-1, 224, 256, 384, 512) K-163: (SHA-1, 224, 256, 384, 512) K-233: (SHA-1, 224, 256, 384, 512) K-283: (SHA-1, 224, 256, 384, 512) K-409: (SHA-1, 224, 256, 384, 512) K-571: (SHA-1, 224, 256, 384, 512 B-163: (SHA-1, 224, 256, 384, 512) B-233: (SHA-1, 224, 256, 384, 512) B-283: (SHA-1, 224, 256, 384, 512) B-409: (SHA-1, 224, 256, 384, 512) B-571: (SHA-1, 224, 256, 384, 512))	
CVL	SP800-56A (ECDH) **Curves tested:** P-224 P-256 P-384 P-521 K-233 K-283 K-409 K-571 B-233 B-283 B-409 B-571	615 A950

Table 5 – Approved Cryptographic Algorithms

7.1.2 Non-FIPS Approved Algorithms Allowed in FIPS Mode

The module supports the following non-FIPS approved algorithms which are permitted for use in the FIPS approved mode:

- Diffie-Hellman (key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength; non-compliant less than 112-bits of encryption strength)
- EC Diffie-Hellman (CVL Certs. #615 and #A950, key agreement; key establishment methodology provides between 128 and 256 bits of encryption strength; non-compliant less than 128-bits of encryption strength)
- RSA (key wrapping; key establishment methodology provides between 112 and 128 bits of encryption strength; non-compliant less than 112-bits of encryption strength)

7.1.3 Non-FIPS Approved Algorithms Not-Allowed in FIPS Mode

The module supports the following non-FIPS approved algorithms which are not permitted for use in the FIPS approved mode:

- Diffie-Hellman with 1024-bit keys,
- EC Diffie-Hellman with B, K and P curves sizes 163 and 192,
- RSA Signature Generation with 1024-bit keys,
- Two-Key Triple-DES encryption.

7.2 Key Generation

The Module supports generation of DH, ECDH, DSA, RSA, and FIPS 186-4 ECDSA public-private key pairs. The Module employs a NIST SP800-90A random number generator for creation of both symmetric keys and the seed for asymmetric key generation.

The module passively receives entropy from the calling application via the RAND_add() API. Because the amount of entropy loaded by the application is dependent on the "entropy" parameter used by the calling application, the minimum number of bits of entropy is considered equal to the "entropy" parameter selection of the calling application. The calling application must call the RAND_add() with the "entropy" parameter of at least 32-bytes (256-bits).

7.3 Key Storage

Public and private keys are provided to the Module by the calling process and are destroyed when released by the appropriate API function calls. The Module does not perform persistent storage of keys.

7.4 Key Access

An authorized application as user (the Crypto-User) has access to all key data generated during the operation of the Module.

7.5 Key Protection and Zeroization

Keys residing in internally allocated data structures can only be accessed using the Module defined API. The operating system protects memory and process space from unauthorized access. Zeroization of sensitive data is performed automatically by API function calls for intermediate data items. Only the process that creates or imports keys can use or export them. No persistent storage of key data is performed by the Module. All API functions are executed by the invoking process in a nonoverlapping sequence such that no two API functions will execute concurrently. All CSPs can be zeroized by power-cycling the module (with the exception of the Software Integrity key). In the event Module power is lost and restored the consuming application must ensure that any AES-GCM keys used for encryption or decryption are re-distributed.

The module supports the following keys and critical security parameters (CSPs):

ID	Algorithm	Size	Description
Symmetric Keys	AES Triple-DES	AES (ECB, CFB8, OFB, CTR, CCM, GCM, XTS): 128, 192, 256 bits Triple-DES (TECB, TCFB1, TCFB8, TCFB64, TOFB): 112, 168 bits	Used for symmetric encryption/decryption

ID	Algorithm	Size	Description
		Triple-DES Notes Related to use if Two-Key algorithm: *Two-key Triple-DES may only be used to decrypt data. Two-key Triple-DES may not be used to encrypt data*	
Asymmetric Keys	RSA DSA ECDSA	RSA (FIPS 186-4): 1,024-4,096 bits DSA (FIPS 186-4): 1,024, 2,048, 3,072 bits ECDSA (FIPS 186-4): P-192, P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571	Used for signature verification. RSA: Also used for key transport.
Asymmetric Keys	RSA DSA ECDSA	RSA (FIPS 186-4): 2,048-4,096 bits DSA (FIPS 186-4): 2,048, 3,072 bits ECDSA (FIPS 186-4): P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571	Used for signature generation with SHA-2 used in key pair generation. RSA: Also used for key transport.
Diffie-Hellman/ EC DiffieHellman private key	DH ECDH	DH: Public Key – 2,048-10,000 bits Private Key – 224-512 bits ECDH: P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571	Used for key agreement
Hash_DRBG	DRBG (as per NIST SP 800-90A)	– V (440/888 bits) – C (440/888 bits) – entropy input (The length of the selected hash)	CSPs as per NIST SP800-90A.
HMAC_DRBG	DRBG (as per NIST SP 800-90A)	– V (160/224/256/384/512 bits) – Key (160/224/256/384/512 bits) – entropy input (The length of the selected hash)	CSPs as per NIST SP800-90A.
CTR_DRBG	DRBG (as per NIST SP 800-90A)	– V (128 bits) – Key (AES 128/192/256) – entropy input (The length of the selected AES)	CSPs as per NIST SP800-90A.
Keyed Hash key	HMAC	All supported key sizes for HMAC	Used for keyed hash

ID	Algorithm	Size	Description
		(Keys must be a minimum 112-bits)	
Software Integrity key	HMAC	HMAC-SHA-1	Used to perform software integrity test at power-on. This key is embedded within the module.

Table 7 – Cryptographic Keys and CSPs

8 Electromagnetic Interference/Compatibility

The Module only electromagnetic interference produced is that of the host platform on which the module resides and executes. FIPS 140-2 requires that the host systems on which FIPS 140-2 testing is performed meet the Federal Communications Commission (FCC) EMI and EMC requirements for business use as defined in Subpart B, Class A of FCC 47 Code of Federal Regulations Part 15. However, all systems sold in the United States must meet these applicable FCC requirements.

9 Self-Tests

The Module performs both power-up self-tests at module initialization[1] and continuous condition tests during operation. Input, output, and cryptographic functions cannot be performed while the Module is in a self-test or error state as the module is single threaded and will not return to the calling application until the power-up self-tests are complete. If the power-up self- tests fail subsequent calls to the module will fail and thus no further cryptographic operations are possible.

The self-tests are called when initializing the module, or alternatively can be invoked at operator discretion using the *FIPS_selftest()* function call.

9.1 Power-On Self Tests (POST)

- AES Known Answer Test (Separate encrypt and decrypt)
- AES-CCM Known Answer Test (Separate encrypt and decrypt)
- AES-GCM Known Answer Test (Separate encrypt and decrypt)
- AES-CMAC Known Answer Test
- AES-XTS Known Answer Test (Separate encrypt and decrypt)
- Triple-DES Known Answer Test (Separate encrypt and decrypt)
- RSA Known Answer Test
- DSA Sign/Verify Test
- FIPS 186-4 ECDSA Sign/Verify Test
- HMAC Known Answer Tests
    - HMAC-SHA1 Known Answer Test
    - HMAC-SHA224 Known Answer Test
    - HMAC-SHA256 Known Answer Test
    - HMAC-SHA384 Known Answer Test
    - HMAC-SHA512 Known Answer Test
- DRBG Known Answer Tests
    - HASH_DRBG Known Answer Test
    - HMAC_DRBG Known Answer Test
    - CTR_DRBG Known Answer Test
- KAS SP 800-56A ECDH Primitive "Z" KAT
- Software Integrity Test (HMAC-SHA1)

9.2 Conditional tests

- Pairwise consistency tests for RSA, DSA, and ECDSA
- Continuous random number generation test for approved DRBG.

[1] The FIPS mode initialization is performed when the application invokes the FIPS_mode_set() call which returns a "1" for success and "0" for failure

9.3 Critical Function Tests

Applicable to the DRBG, as per SP800-90A, Section 11:

- Instantiate Test
- Generate Test
- Reseed Test
- Uninstantiate Test

10 Design Assurance

10.1 Secure Distribution and Installation

The NCSM is intended only for use by NetApp personnel and as such is accessible only from the secure NetApp internal web site. Only authorized employees have access to the module.

A complete revision history of the source code from which the Module was generated is maintained in a version control database[2]. The HMAC-SHA-1 of the Module distribution file as tested by the CSTL Laboratory is verified during inclusion of the Module into NetApp products.

The module comes pre-installed on various NetApp products. No customer installation is necessary.

10.2 Secure Operation

The module is architected to be compliant with all FIPS 140-2 power-on requirements. Upon invocation of the shared library or application into which the object module has been compiled, the module begins to execute a prescribed set of startup tasks including both an integrity test and the POSTs described in section 9.1 above.

If any component of the module startup fails, an internal global error flag is set to prevent subsequent invocation of any cryptographic function calls. Any such startup failure is a hard error that can only be recovered by reinstalling the Module. Upon successful completion of all startup tasks, the Module is available to enter a FIPS mode of operation at a later time if desired.

A single initialization call, *FIPS_mode_set()*, is required to initialize the Module for operation in the FIPS 140-2 Approved mode. When the Module is in FIPS mode all security functions and cryptographic algorithms are performed in Approved mode.

The FIPS mode initialization is performed when the application invokes the *FIPS_mode_set()* call which returns a "1" for success and "0" for failure. Interpretation of this return code is the responsibility of the host application. Prior to this invocation the Module is operating in the nonFIPS mode by default.

No operator intervention is required during the running of the self-tests.

None of algorithms identified in section 7.1.3 may be used in the FIPS-approved mode of operation. The operator must ensure that these are not chosen/selected.

[2] This database is internal to NetApp since the intended use of this crypto module is by NetApp development teams