NETSCOUT

NETSCOUT FIPS Object Module *by* NETSCOUT SYSTEMS, INC.

FIPS 140-2 Non-Proprietary Security Policy

Software Version: 1.0

FIPS 140-2, Level 1 Validation

May 2019

Contents

1. Introduction

NETSCOUT SYSTEMS, INC. is a provider of application and network performance management products. Headquartered in Westford, Massachusetts, NETSCOUT serves enterprises community, government agencies and telecommunications service providers.

1.1. Module Overview

This document is a FIPS 140-2 Security Policy for the NETSCOUT FIPS Object Module; also referred to as "the module". This policy describes how the module meets the FIPS 140-2 security requirements and how to operate the module in a FIPS compliant manner.

This policy was created as part of the FIPS 140-2, Level 1 validation effort of the module. Federal Information Processing Standards Publication 140-2 "***Security Requirements for Cryptographic modules (FIPS 140-2)***" details the United States Federal Government requirements for cryptographic modules. Detailed information about the FIPS 140-2 standard and validation program is available on the NIST website at *http://csrc.nist.gov/groups/STM/cmvp/index.html*.

The NETSCOUT FIPS Object Module provides cryptographic functionality to NETSCOUT's series of applications. The module is classified under FIPS 140-2 as a software based, multi-chip standalone module embodiment. The module itself is a statically linked object module (fipscanister.o), intended to be linked to a calling application at build time. The physical cryptographic boundary is considered as the general-purpose computing (GPC) platforms on which the module was tested. The logical cryptographic boundary of the module is the pre-compiled object file which provides the necessary cryptographic functions. Within the logical boundary lies the algorithmic boundary (NETSCOUT Cryptographic Library), represented by the NIST CAVP tested algorithms specified in Table 2.

The tar archive containing the NETSCOUT FIPS Object Module prior to build time, contains the HMAC-SHA-1 digest: **7f486fbb598f3247ab9db10c1308f1c19f384671** when the following command is issued:
openssl sha1 -hmac etaonrishdlcupfm openssl-fips-2.0.8.tar.gz

The module was tested in the following operational environments, and is only considered to be a FIPS 140-2 validated module when operating in these environments:

- Custom Linux 3.7.5 running on a NETSCOUT 1400 Series {1410} (Intel Xeon E5) with PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 1400 Series {1410} (Intel Xeon E5) without PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 1900 Series {1910D} (Intel Xeon E5) with PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 1900 Series {1910D} (Intel Xeon E5) without PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 2400 Series {2410} (Intel Xeon E5) with PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 2400 Series {2410} (Intel Xeon E5) without PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 2600 Series {2695} (Intel Xeon E5) with PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 2600 Series {2695} (Intel Xeon E5) without PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 2900 Series {2910D} (Intel Xeon E5) with PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 2900 Series {2910D} (Intel Xeon E5) without PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 3300 Series {3300} (Intel Xeon E5) with PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 3300 Series {3300} (Intel Xeon E5) without PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 4500 Series {4595D} (Intel Xeon E5) with PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 4500 Series {4595D} (Intel Xeon E5) without PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 4700 Series {4795} (Intel Xeon E5) with PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 4700 Series {4795} (Intel Xeon E5) without PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 4800 Series {4895} (Intel Xeon E5) with PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 4800 Series {4895} (Intel Xeon E5) without PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 6600 Series {6695} (Intel Xeon E5) with PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 6600 Series {6695} (Intel Xeon E5) without PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 9700 Series {9795} (Intel Xeon E5) with PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 9700 Series {9795} (Intel Xeon E5) without PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 9800 Series {9802} (Intel Xeon E5) with PAA;
- Custom Linux 3.7.5 running on a NETSCOUT 9800 Series {9802} (Intel Xeon E5) without PAA;
- Custom Linux 2.6.39.1 running on a NETSCOUT 6900C Series {69XXC} (Intel Xeon X 5620) with PAA;

- Custom Linux 2.6.39.1 running on a NETSCOUT 6900C Series {69XXC} (Intel Xeon X 5620) without PAA;
- Custom Linux 2.6.39.1 running on a NETSCOUT 7900C Series {79XXC} (Intel Xeon X 5620) with PAA;
- Custom Linux 2.6.39.1 running on a NETSCOUT 7900C Series {79XXC} (Intel Xeon X 5620) without PAA;
- Custom Linux 3.10 running on a NETSCOUT PowerEdge R730 {enhanced} (Intel Xeon E5-2697 v3) with PAA;
- Custom Linux 3.10 running on a NETSCOUT PowerEdge R730 {enhanced} (Intel Xeon E5-2697 v3) without PAA;
- Custom Linux 3.10 running on a NETSCOUT PowerEdge R740 {standard} (Intel Xeon Silver) with PAA;
- Custom Linux 3.10 running on a NETSCOUT PowerEdge R740 {standard} (Intel Xeon Silver) without PAA;
- Custom Linux 3.10 running on a NETSCOUT PowerEdge R740 {enhanced} (Intel Xeon Gold) with PAA;
- Custom Linux 3.10 running on a NETSCOUT PowerEdge R740 {enhanced} (Intel Xeon Gold) without PAA;
- Custom Linux 3.10 running on a NETSCOUT 2400 Series {2410} (Intel Xeon Silver) with PAA;
- Custom Linux 3.10 running on a NETSCOUT 2400 Series {2410} (Intel Xeon Silver) without PAA;
- Custom Linux 3.10 running on a NETSCOUT 4700 Series {4795} (Intel Xeon Gold) with PAA;
- Custom Linux 3.10 running on a NETSCOUT 4700 Series {4795} (Intel Xeon Gold) without PAA;
- Custom Linux 3.10 running on a NETSCOUT 4800 Series {4895} (Intel Xeon Gold) with PAA;
- Custom Linux 3.10 running on a NETSCOUT 4800 Series {4895} (Intel Xeon Gold) without PAA;
- Custom Linux 3.10 running on a NETSCOUT 9700 Series {9795} (Intel Xeon Gold) with PAA;
- Custom Linux 3.10 running on a NETSCOUT 9700 Series {9795} (Intel Xeon Gold) without PAA;
- Custom Linux 3.10 running on a NETSCOUT 9800 Series {9802} (Intel Xeon Gold) with PAA;
- Custom Linux 3.10 running on a NETSCOUT 9800 Series {9802} (Intel Xeon Gold) without PAA;

- ArbOS 7.0 running on a NETSCOUT Arbor Edge Defense and APS Systems 2600 (Intel Xeon E5) with PAA;
- ArbOS 7.0 running on a NETSCOUT Arbor Edge Defense and APS Systems 2600 (Intel Xeon E5) without PAA;
- ArbOS 7.0 running on a NETSCOUT Arbor Edge Defense and APS Systems 2800 (Intel Xeon E5) with PAA; and
- ArbOS 7.0 running on a NETSCOUT Arbor Edge Defense and APS Systems 2800 (Intel Xeon E5) without PAA;

(single-user mode)

The security levels supported by the software module are as follows:

Table 1: Summary of FIPS security requirements and compliance levels

Section	Level
1. Cryptographic Module Specification	1
2. Cryptographic Module Ports and Interfaces	1
3. Roles, Services, and Authentication	2
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	3
11. Mitigation of Other Attacks	N/A
Overall Level	1

2. Modes of Operation & Cryptographic Functionality

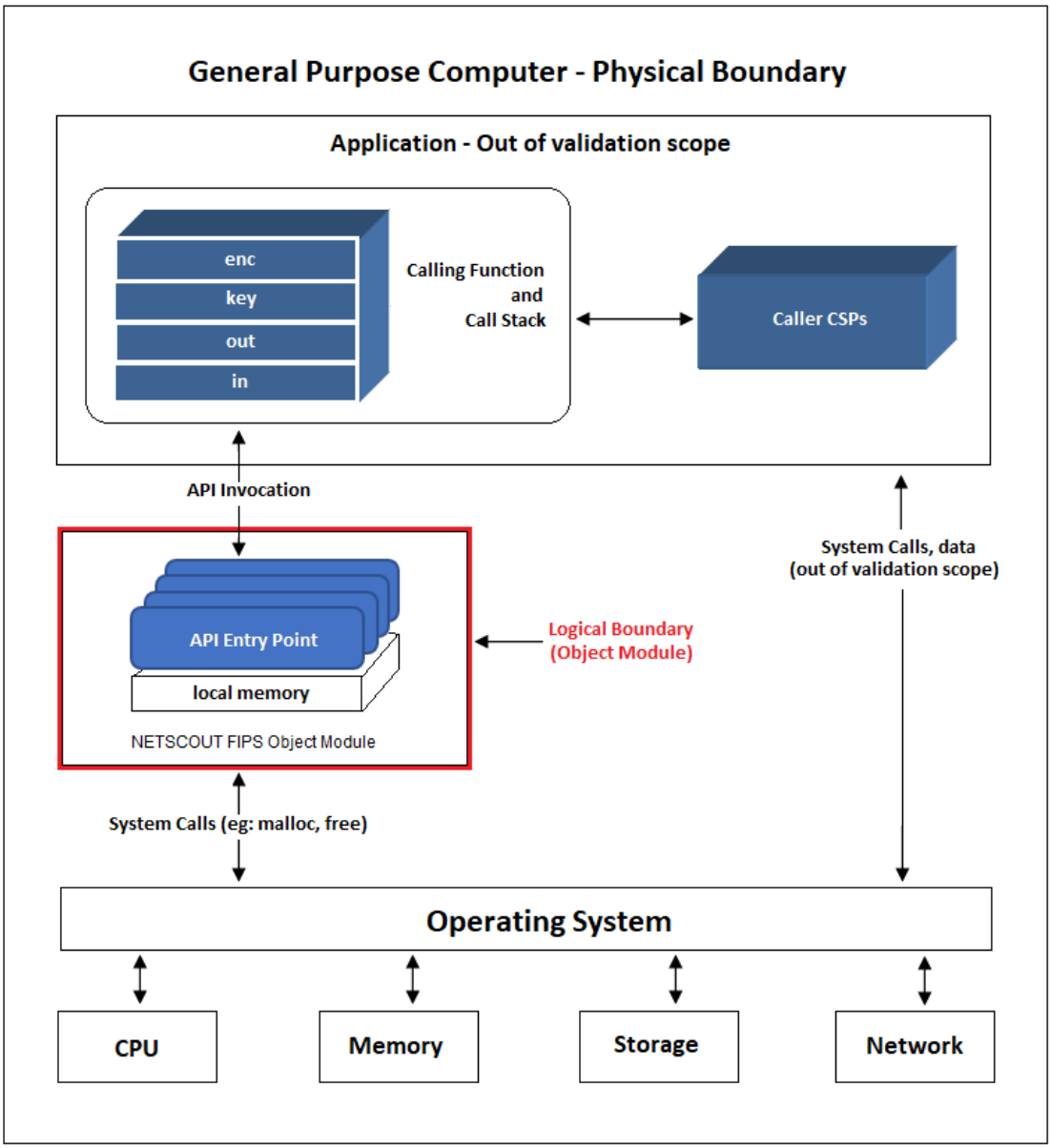

Figure 1: Block Diagram

The module supports both FIPS 140-2 Approved and non-Approved modes. There are also security functions which are non-Approved (but allowed). Tables 2, 3 and 4 list these categories respectively.

3. Approved & Allowed Cryptographic Functions

Table 2: FIPS Approved Cryptographic Functions

Algorithm	Function	Options	CAVP Cert.
AES [FIPS 197] AES [SP 80038B] CMAC [SP 80038C] CCM [SP 80038D] GCM	Encryption, Decryption and CMAC	ECB Mode: Encrypt/Decrypt Key Size: 128, 192, 256. CBC Mode: Encrypt/Decrypt Key Size: 128, 192, 256. OFB Mode: Encrypt/Decrypt Key Size: 128, 192, 256. CFB1 Mode: Encrypt/Decrypt Key Size: 128, 192, 256. CFB8 Mode: Encrypt/Decrypt Key Size: 128, 192, 256. CFB128 Mode:Encrypt/Decrypt Key Size: 128, 192, 256. CTR Mode: Encrypt only Key Size: 128 192 256 CMAC Generation using AES (128, 192, 256) CMAC Verification using AES 128, 192, 256) CCM using (128, 192, 256) AES GCM Mode: Encrypt/Decrypt – Key Size: 128, 192, 256	**Certs.** **#5669** **#5672** **#5900** **#5936** **#5938** **#C196** **#C197**
CVL	Key Agreement	SP 800-56A ECC CDH Primitive (Section 5.7.1.2) Component 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 KAS ECC: Ephemeral Unified Curves tested: P-224 w SHA-224, P-256 w SHA-256, P-384 w SHA-384, P-521 w SHA-512	**Certs.** **#2056** **#2124** **#C197**
DRBG (NIST SP 800-90A)	Random Number Generation Symmetric Key Generation	Hash_DRBG (SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512) HMAC_DRBG (SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512) CTR DRBG (AES-128, AES-192 and AES-256)	**Certs.** **#2291** **#2462** **#2489** **#2491** **#C196** **#C197**

DSA	Digital Signature Operations	PQG Generation: L= 2048, 3072 N= 224, 256 SHA = 224, 256, 384 and 512 PQG Verification: L= 1024, 2048, 3072 N= 224, 256 SHA = 224, 256, 384 and 512 Key Pair: L= 2048, 3072 N= 224, 256 Signature Generation: L= 2048, 3072 N= 224, 256 SHA = 224, 256, 384 and 512 Signature Verification: L= 1024, 2048, 3072 N= 160, 224, 256 SHA = 1, 224, 256, 384 and 512	**Certs.** **#1457** **#1495** **#C197**
ECDSA	Elliptic Curve Digital Signature Operations (The Module supports only NIST defined curves for use with ECDSA and ECDH.)	Key Pair Generation Curves: B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P- 256, P-384, P-521 Public Key Validation Curves: B-163, B-233, B-283, B-409, B-571, K-163, K-233, K-283, K-409, K- 571, P-192, P-224, P-256, P-384, P-521 Signature Generation Curve/SHA pairs tested: P = 224, 256, 384 and 521 /w SHA-224, 256, 384 and 512. K = 233, 283, 409 and 571 /w SHA-224, 256, 384 and 512. B = 233, 283, 409 and 571 /w SHA-224, 256, 384 and 512. Signature Verification Curve/SHA pairs tested: P = 192, 224, 256, 384 and 521 /w SHA-1, 224, 256, 384 and 512. K = 163, 233, 283, 409 and 571 /w SHA-1, 224, 256, 384 and 512. B = 163, 233, 283, 409 and 571 /w SHA-1, 224, 256, 384 and 512.	**Certs.** **#1535** **#1571** **#C197**

HMAC	Keyed Hashing Operations	HMAC SHA1: KeySizes tested: KS < BS KS = BS KS > BS MAC sizes tested: 10 12 16 20 HMAC SHA224: KeySizes tested: KS < BS KS = BS KS > BS MAC sizes tested: 14 16 20 24 28 HMAC SHA256: KeySizes tested: KS < BS KS = BS KS > BS MAC sizes tested: 16 24 32 HMAC SHA384: KeySizes tested: KS < BS KS = BS KS > BS MAC sizes tested: 24 32 40 48 HMAC SHA512: KeySizes tested: KS < BS KS = BS KS > BS MAC sizes tested: 32 40 48 56 64	**Certs.** **#3774** **#3880** **#C197**
RSA	RSA Digital Signature Operations	FIPS 186-2 Signature Verification 9.31: Modulus lengths: 1024, 1536, 2048, 3072, 4096 SHAs: SHA-1, SHA-256, SHA-384, SHA-512 Signature Verification PKCS1.5 Modulus lengths: 1024, 1536, 2048, 3072, 4096 SHAs: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 Signature Verification PSS: Modulus lengths: 1024, 1536, 2048, 3072, 4096 SHAs: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 FIPS 186-4 Signature Generation 9.31: Mod 2048 SHA: SHA-256, SHA-384, SHA-512 Mod 3072 SHA: SHA-256, SHA-384, SHA-512 Signature Generation PKCS1.5: Mod 2048 SHA: SHA-224, SHA-256, SHA-384, SHA-512 Mod 3072 SHA: SHA-224, SHA-256, SHA-384, SHA-512 Signature Generation PSS: Mod 2048: SHA-224: Salt Length: 0 SHA-256: Salt Length: 0 SHA-384: Salt Length: 0 SHA- 512: Salt Length: 0 Mod 3072: SHA-224: Salt Length: 0 SHA-256: Salt Length: 0 SHA-384: Salt Length: 0 SHA-512: Salt Length: 0	**Certs.** **#3051** **#3088** **#C197**

		Signature Verification 9.31: Modulus lengths: 1024, 2048, and 3072 SHAs: SHA-1, SHA-256, SHA-384, SHA-512 Signature Verification PKCS1.5 Modulus lengths: 1024, 2048, and 3072 SHAs: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 Signature Verification PSS: Modulus lengths: 1024, 2048, and 3072 SHAs: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	
SHS	Hashing	SHA-1 Byte only SHA-224 Byte only SHA-256 Byte only SHA-384 Byte only SHA-512 Byte only	**Certs.** **#4543** **#4656** **#C197**
Triple-DES[1]	Encryption, Decryption and CMAC	CBC, CFB1, CFB8, CFB64, OFB and ECB Modes: Encrypt/Decrypt Key Option = 1 (K1, K2, K3 independent) CMAC Verification using TDES (3-Key)	**Certs.** **#2841** **#2869** **#C197**

[1] As per the SP 800-67rev1 Transition specified in the CMVP Implementation Guidance, please be advised that this module shall not be used to perform more than 2^20 encryptions with the same Triple-DES key when generated as part of a recognized IETF protocol. If the key is not generated as part of a recognized IETF protocol, then the limit of 2^16 encryptions shall apply.

Table 3: Allowed Cryptographic Functions

Category	Algorithm	Description
Key Encryption, Decryption	RSA	The RSA algorithm is used by the calling application for encryption or decryption of keys. No claim is made for SP 800-56B compliance, and no CSPs are established into or exported out of the module using these services. If the implemented RSA is used in a key transport scheme, please be advised that the supported key strengths range from 1024 to 16384 bits. You must ensure that only keys between 2048 and 16384 bits (providing 112 to 270 bits of encryption strength) are used for this purpose. Failure to use this range of keys will result in a non- compliant module.
Key Agreement	CVL	EC Diffie-Hellman (CVL Certs. #2056, #2124 and #C197, key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength)
NDRNG	N/A	Underlying PRNG supplied by the OS and allowed for use in conjunction with the seeding of the Approved NIST SP 800-90A DRBG. External to the cryptographic boundary of the module.

4. Non-Approved Cryptographic Functions

The following cryptographic services, algorithms and schemes shall not be used in an Approved mode of operation. Any use of these schemes and algorithms will cause the module to be operating in a non- Approved mode. Keys and secret critical security parameters defined in the approved mode of operation, shall not be accessed or shared while in a non-approved mode of operation. Furthermore, critical security parameters shall not be generated while in a non-approved mode. The approved DRBG may be used in a non-approved mode. However, the approved DRBGs seed or seed key shall not be accessed or shared in the non-approved mode. Access rights are denoted below as Read (R), Write (W) or Execute (X).

Table 4: Non-Approved Cryptographic Security Functions & Services

Service	Role	Description	Access	Input	Output
Random number generation	User, CO	ANSI X9.31 RNG (non-compliant) Used for random number and symmetric key generation.	R,W,X	API Call	Return Code
Asymmetric key generation	User, CO	Used to generate DSA, ECDSA and RSA keys: RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK <u>Non-Approved RSA Functions</u> • 186-2 RSA Key Generation – Use of 1024 bit keys (non-compliant) • 186-2 RSA - Use of SHA-1 for Digital Signature Generation (non-compliant) <u>Non-Approved DSA Functions</u> • 186-2 DSA Key Generation – Use of 1024 bit keys (non-compliant) • 186-2 DSA - Use of SHA-1 for Digital Signature Generation (non-compliant) 186-4 DSA Key Generation – Use of 1024 bit keys (non-compliant) • 186-4 DSA - Use of SHA-1 for Digital Signature Generation (non-compliant) <u>Non-Approved ECDSA Functions</u> • 186-2 ECDSA – Use of curves PKG: CURVES(P-192 K-163 B-163) SIG(gen): 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)	R,W,X	API Call	Return Code
Key agreement	User, CO	[SP 800-56A] (5.7.1.2) - All NIST Recommended B, K and P curves sizes 163 and 192	R,W,X	API Call	Return Code
Storage Device Confidentiality	User, CO	[SP 800-38E] XTS (non-Approved due to lack of comparison test (IG A.9)	R,W,X	API Call	Return Code

5. Critical Security Parameters & Public Keys

All CSPs used by the module are described below. The CSP names are generic, corresponding to API parameter data structures.

Table 5: Module CSPs

CSP Name	Description
RSA SGK	RSA (2048 to 16384 bits) signature generation key
RSA KDK	RSA (2048 to 16384 bits) key decryption (private key transport) key
DSA SGK	[FIPS 186-4] DSA (2048/3072) signature generation key
ECDSA SGK	ECDSA (All NIST defined B, K, and P curves except sizes 163 and 192) signature generation key
EC DH Private	EC DH (All NIST defined B, K, and P curves except sizes 163 and 192) private key agreement key.
AES EDK	AES (128/192/256) encrypt / decrypt key
AES CMAC	AES (128/192/256) CMAC generate / verify key
AES GCM	AES (128/192/256) encrypt / decrypt / generate / verify key
TDES EDK	TDES (3-Key) encrypt / decrypt key (192 bits/168 key bits, providing 112 bits of strength)
TDES CMAC	TDES (3-Key) CMAC generate / verify key
HMAC Key	Keyed hash key (160/224/256/384/512)
Hash_DRBG CSPs	V (440/888 bits), C (440/888 bits), seed, and entropy input (length dependent on security strength)
HMAC_DRBG CSPs	V (160/224/256/384/512 bits), seed, Key (160/224/256/384/512 bits) and entropy input (length dependent on security strength)
CTR_DRBG CSPs	V (128 bits), seed, Key (AES 128/192/256) and entropy input (length dependent on security strength)
COADDigest	Precalculated HMACSHA1 digest used for Crypto Officer role authentication
UserADDigest	Precalculated HMACSHA1 digest used for User role authentication

Table 6: Module Public Keys

CSP Name	Description
RSA SVK	RSA (2048 to 16384 bits) signature verification public key
RSA KEK	RSA (2048 to 16384 bits) key encryption (public key transport) key
DSA SVK	[FIPS 186-4] DSA (2048/3072) signature verification key or [FIPS 186-2] DSA (1024) signature verification key
ECDSA SVK	ECDSA (All NIST defined B, K and P curves) signature verification key
EC DH Public	EC DH (All NIST defined B, K and P curves) public key agreement key.

6. Key Management

6.1. Key/CSP Storage

The Module stores DRBG state values for the lifetime of the DRBG instance. The module uses CSPs passed in by the calling application on the stack. The Module does not store any CSP persistently (beyond the lifetime of an API call), except for DRBG state values used for the Module's default key generation service.

6.2. Key/CSP Generation

The Module implements NIST SP 800-90A compliant DRBG services for the creation of symmetric keys, and for generation of DSA, elliptic curve, and RSA keys. The calling application is responsible for the storage of generated keys returned by the module. Symmetric keys are the direct output of the DRBG. Asymmetric key generation conforms to FIPS PUB 186-4, except in the case of RSA.

6.3. Key/CSP Entry

All CSPs enter the Module's logical boundary in plaintext as API parameters, associated by memory location. However, none cross the physical boundary.

6.4. Key/CSP Output

The Module may output any keys or CSPs as part of the explicit results of key generation services. The calling application is responsible for the management and protection of all keys and CSPs. The module itself does not export keys outside the physical boundary.

6.5. Key/CSP Destruction

Zeroization of sensitive data is performed automatically by API function calls for temporarily stored CSPs. In addition, the module provides functions to explicitly destroy CSPs related to random number generation services. The calling application is responsible for parameters passed in and out of the module.

Private and secret keys are provided to the module by the calling application and are destroyed when released by the appropriate API function calls. Keys residing in internally allocated data structures (during the lifetime of an API call) can only be accessed using the Module defined API. The operating system protects memory and process space from unauthorized access. Only the calling application that creates or imports keys can use or export such keys. All API functions are executed by the invoking calling application in a non-overlapping sequence such that no two API functions will execute concurrently. An authorized application as user (Crypto-Officer and User) has access to all key data generated during the operation of the module.

The AES - GCM key and IV is generated as per IG A.5, and the Initialization Vector (IV) is a minimum of 96 bits. If module power is lost and restored, the calling application shall ensure that any AES-GCM keys used for encryption or decryption are redistributed.

7. Instructions for Operating in the Approved Mode

The NETSCOUT FIPS Object Module is a software module, which is intended to be used with NETSCOUT's line of software application products. Tables 2 and 3 in this document, serve as the benchmark for cryptographic algorithms and schemes which allow the module to operate in the FIPS 140-2 compliant mode of operation. In order to maintain operation in the Approved mode, the module shall be started using the FIPS_module_mode_set() command, and only the Approved cryptographic functions specified in Table 2 and the Allowed cryptographic functions specified in Table 3 shall be used. For every security function that is executed from Table 4, the module will be automatically operating in the non-Approved mode during the time such functions are active. The calling application is responsible for invoking the module using an API call, which returns a "1" for success and "0" for failure. If the initialization process fails for any reason, then all cryptographic services fail from this point on. The specifics of the error are translated by the calling application.

8. Ports & Interfaces

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

Table 7: Mapping for Logical Interfaces to FIPS 140-2

Logical interface type	Description
Control Input	API entry point and corresponding stack parameters
Data Input	API entry point data input stack parameters
Data Output	API entry point data output stack parameters
Status Output	API entry point return values and status stack parameters

As a software module, control of the physical ports is outside the scope of the module. However, when the module is performing self-tests, or is in an error state, all output on the logical data output interface is inhibited. The module is single-threaded and when in the error state, returns only an error value. (No data output is returned).

9. Roles Services & Authentication

The module implements a very basic role-based authentication method, inherited from the FIPS Object Module on which it is based. Since the module is built and linked at NETSCOUT, and the option of having a Crypto-Officer and User password must be implemented at build time, there is no practical option for operators to use this feature. Due to the standard way in which the module is built according to the FIPS Object Module security policy, the default (and published) authentication key and passwords are implemented according to Page 97 of the "*User Guide for the OpenSSL FIPS Object Module v2.0*". (This occurs by default when the module is built without the option of injecting unique passwords.) Please be advised, that while the password can be changed to a non-default value at build time, any known, published value cannot be considered secret, and therefore does not meet the requirement.

For completeness, the strength of authentication for these known values are as follows: The minimum password length is 16 characters, the probability of a random successful authentication attempt in one try is a maximum of 1/256^16, or less than 1/10^38. The Module permanently disables further authentication attempts after a single failure, so this probability is independent of time.

Only one role may be active at a time, as the module does not allow concurrent operators. The User and Crypto-Officer roles are assumed implicitly. Access rights are denoted below as Read (R), Write (W) or Execute (X).

The underlying, operating system segregates operator processes into separate process spaces. Each process space is logically separated from all other processes by the operating system software and hardware. The module functions entirely within the process space of the calling application, and implicitly satisfies the FIPS 140-2 requirement for a single user mode of operation.

Both roles have access to all services provided by the module.

- User Role (User): Loading the Module and calling any of the API functions.
- Crypto Officer Role (CO): Installation of the Module within the non-modifiable OE and calling of any API functions.

Table 8: Approved Services & CSP Access

Service	Role	Description	Access	Input	Output
Initialize	User, CO	Module initialization. Does not access CSPs.	R,X	API Call	Return Code
Self-test	User, CO	Perform self-tests (FIPS_selftest). Does not access CSPs.	R,X	API Call	Return Code
Authenticate	User, CO	Role-based authentication using passwords published in FIPS Object Module User Guide.	R	API Call	Return Code
Show status	User, CO	Functions that provide module status information: - Version (as unsigned long or const char *) - FIPS Mode (Boolean) Does not access CSPs.	R,X	API Call	Return Code
Zeroize	User, CO	Functions that destroy CSPs: - fips_drbg_uninstantiate: for a given DRBG context, overwrites DRBG CSPs (Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs) All other services automatically overwrite CSPs stored in allocated memory.	R,W,X	API Call	Return Code

Random number generation	User, CO	Used for random number and symmetric key generation. - Seed or reseed a DRBG instance - Determine security strength of a DRBG instance - Obtain random data Uses and updates Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs.	R,W,X	API Call	Return Code
Asymmetric key generation	User, CO	Used to generate DSA, ECDSA and RSA keys: RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK There is one supported entropy strength for each mechanism and algorithm type, the maximum specified in SP800-90	R,W,X	API Call	Return Code
Symmetric encrypt/decrypt	User, CO	Used to encrypt or decrypt data. Executes using AES EDK, AES, GCM, TDES EDK (passed in by the calling process).	R,W,X	API Call	Return Code
Symmetric digest	User, CO	Used to generate or verify data integrity with CMAC. Executes using AES CMAC, TDES, CMAC (passed in by the calling process).	R,W,X	API Call	Return Code
Message digest	User, CO	Used to generate a SHA-1 or SHA-2 message digest. Does not access CSPs.	R,W,X	API Call	Return Code
Keyed Hash	User, CO	Used to generate or verify data integrity with HMAC. Executes using HMAC Key (passed in by the calling process).	R,W,X	API Call	Return Code
Key transport	User, CO	Used to encrypt or decrypt a key value on behalf of the calling process (does not establish keys into the module). Executes using RSA KDK, RSA KEK (passed in by the calling process).	R,W,X	API Call	Return Code
Key agreement	User, CO	Used to perform key agreement primitives on behalf of the calling process (does not establish keys into the module). Executes using EC DH Private, EC DH Public (passed in by the calling process).	R,W,X	API Call	Return Code
Digital signature	User, CO	Used to generate or verify RSA, DSA or ECDSA digital signatures. Executes using RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK (passed in by the calling process).	R,W,X	API Call	Return Code
Utility	User, CO	Miscellaneous helper functions. Does not access CSPs.	R,W,X	API Call	Return Code

10. Physical Security

The module maintains physical security by using production grade components and standard passivation, as allowed by FIPS 140-2 level 1.

11. Module Self-Tests

The module performs the applicable power-up self-tests listed below, when initialized (or on-demand):

Table 9: Module Power-Up Self-Tests

Algorithm/Scheme	Type	Description
Software Integrity Test	Known Answer Test	HMAC-SHA-1
HMAC	Known Answer Test	One KAT per SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 (Per IG 9.3, this testing covers SHA POST requirements.)
AES	Known Answer Test	Separate encrypt and decrypt, ECB mode, 128 bit key length
AES CCM	Known Answer Test	Separate encrypt and decrypt, 192 key length
AES GCM	Known Answer Test	Separate encrypt and decrypt, 256 key length
XTS-AES	Known Answer Test	128, 256 bit key sizes to support either the 256-bit key size (for XTSAES128) Or the 512bit key size (for XTSAES256)
AES-CMAC	Known Answer Test	Generate and verify CBC mode, 128, 192, 256 key lengths
Triple-DES	Known Answer Test	3-Key Triple-DES with separate encrypt and decrypt, ECB mode.
Triple-DES-CMAC	Known Answer Test	3-Key Triple-DES with CMAC generate and verify, CBC mode.
RSA	Known Answer Test	Sign and verify using 2048 bit key, SHA-256, PKCS#1
DSA	Known Answer Test	Sign and verify using 2048 bit key, SHA-384
NIST SP 800-90A DRBG	Known Answer Test	CTR_DRBG: AES, 256-bit with and without derivation function HASH_DRBG: SHA-256 HMAC_DRBG: SHA-256
ECDSA	Known Answer Test	Keygen, sign, verify using P224, K233 and SHA512.
EC Diffie-Hellman	Known Answer Test	Shared secret calculation per NIST SP 800-56A §5.7.1.2, IG 9.6

The initialization API call FIPS_mode_set() invokes the module itself and all subsequent power-up self-tests automatically and without operator intervention. If any component of the power-up self-test fails, an internal flag is set to prevent subsequent invocation of any cryptographic function calls. The module will only enter the FIPS Approved mode if the module is reloaded and the re-initialization succeeds. The power-up self-tests can be performed on-demand by re-initializing the module. Any failure of a power-up self-test represents a hard error, which means the module must be replaced. The operator may attempt to restart the module to clear any error, however hard errors will require replacement of the module. Upon cryptographic service failure (including initialization, self-tests and conditional failures), the operator can call the last error API function to get the error associated with the failure.

The module performs the applicable conditional self-tests listed below:

Table 10: Module Conditional Self-Tests

Algorithm/Scheme	Type	Description
NIST SP 800-90A DRBG	Known Answer Test	As per Section 11.3 of NIST SP 800-90A - Conditional upon Instantiation, Generation, Reseed and Uninstantiation.
NIST SP 800-90A DRBG	Continuous Test	Continuous test for DRBG stuck fault.
NDRNG	Continuous Test	OS based entropy source. Continuous test performed.
ANSI X9.31 DRNG	Continuous Test	Continuous test for DRNG stuck fault. (This is a non-compliant DRNG which executes only in the non-Approved mode.)
RSA	Pairwise Consistency Test	Performed upon the condition of RSA keypair generation. *(Note: this test is performed in the non-Approved mode only, as the implemented RSA Key Generation does not conform to FIPS 186-4.)*
DSA	Pairwise Consistency Test	Performed upon the condition of DSA keypair generation.
ECDSA	Pairwise Consistency Test	Performed upon the condition of ECDSA keypair generation.

Notes:

- In the event of a DRBG self-test failure, it is necessary for the calling application to uninstantiate and re-instantiate the DRBG as per the requirements of SP 800-90A. The implemented NIST SP 800-90A DRBGs (Hash, HMAC and CTR) all contain the critical functions tests for instantiate, generate, reseed and un-instantiate.
- Pairwise Consistency Tests are performed for both Sign/Verify and Encrypt/Decrypt.
- The Module supports all NIST defined curves.
- The resulting symmetric key or generated seed is an unmodified output from the DRBG.
- The application developer ***shall*** ensure that the blocking */dev/random* character device is utilized.

12. Mitigation of Other Attacks

The module is not designed to mitigate against attacks which are outside of the scope of FIPS 140-2.

NETSCOUT SYSTEMS, INC.
310 Littleton Road
Westford, MA 01886-4105
Phone: 978-614-4000
Toll Free: 888-357-7667
Fax: 978-614-4004