Nutanix Cryptographic Module for OpenSSH Client

Version 6.0

FIPS 140-2 Non-Proprietary Security Policy

Document Version 1.8

November 8, 2022

Prepared for:

Nutanix, Inc.
1740 Technology Drive, Suite 150
San Jose, CA 95110
nutanix.com
+1 855.NUTANIX

Prepared by:

KeyPair Consulting Inc.
987 Osos Street
San Luis Obispo, CA 93401
keypair.us
+1 805.316.5024

Table of Contents

List of Tables

List of Figures

References

Ref	Full Specification Name
	References used in Approved Algorithms Table
[38A]	NIST SP 800-38A, *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*, Dec 2001
[38C]	NIST SP 800-38C, *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality*, Jul 2007
[38D]	NIST SP 800-38D, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, Nov 2007
[38F]	NIST SP 800-38F, *Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping*, Dec 2012
[56Ar3]	NIST SP 800-56A Rev. 3, *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*, Apr 2018
[57P1]	NIST SP 800-57 Part 1 Rev. 5, *Recommendation for Key Management: Part 1 - General,* May 2020
[90A]	NIST SP 800-90A Rev. 1, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*, Jun 2015
[135]	NIST SP 800-135 Rev. 1, *Recommendation for Existing Application-Specific Key Derivation Functions*, Dec 2011
[180]	FIPS 180-4, *Secure Hash Standard (SHS)*, Aug 2015
[186]	FIPS 186-4, *Digital Signature Standard (DSS)*, Jul 2013
[197]	FIPS 197, *Advanced Encryption Standard (AES)*, Nov 2001
[198]	FIPS 198-1, *The Keyed Hash Message Authentication Code (HMAC)*, Jul 2008
	Other References
[140]	FIPS 140-2, *Security Requirements for Cryptographic Modules*, May 2001
[140DTR]	FIPS 140-2 *Derived Test Requirements*, Jan 2011
[140IG]	*Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program*, Mar 2022
[131A]	NIST SP 800-131A Rev. 2, *Transitioning the Use of Cryptographic Algorithms and Key Lengths*, Mar 2019
[UG]	Nutanix FIPS 140-2 Cryptographic Modules for OpenSSL and OpenSSH - User Guide

Acronyms and Definitions

Term	Definition
AES	Advanced Encryption Standard [197]
AHV	Acropolis Hypervisor
API	Application Programming Interface
CAVP	Cryptographic Algorithm Validation Program
CMVP	Cryptographic Module Validation Program
CO	Cryptographic Officer
CSP	Critical Security Parameter [140]
DRBG	Deterministic Random Number Generator [90A]
DTR	Derived Test Requirements, see [140DTR]
FIPS	Federal Information Processing Standard
HMAC	Keyed-Hash Message Authentication Code [198]
IG	Implementation Guidance [140IG]
KAT	Known Answer Test
KDF	Key Derivation Function
NDRNG	Non-Deterministic Random Number Generator
NIST	National Institute of Standards and Technology
RSA	Rivest, Shamir, and Adleman Algorithm [186]
SHA/SHS	Secure Hash Algorithm/Standard [180]
SP	Special Publication
SSH	Secure Shell

1 Overview

This document defines the non-proprietary Security Policy for the Nutanix Cryptographic Module for OpenSSH Client, hereafter denoted the module. The module is a cryptographic software application, designated as multi-chip standalone embodiment in [140] terminology, used in Nutanix, Inc. (Nutanix) solutions to provide FIPS 140-2 Approved SSH client-side secure communication.

The module meets FIPS 140-2 overall Level 1 requirements, with security levels as follows:

Table 1: Security Level of Security Requirements

Security Requirement	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A

In Table 1 above, [140] Section 4.5 *Physical Security* is not applicable, as permitted by [140IG] 1.16 *Software Module* and [140IG] G.3 *Partial Validations and Not Applicable Areas of FIPS 140-2*.

The module design corresponds to the module security rules. Security rules enforced by the module are described in the appropriate context of this document.

The module is bound to FIPS 140-2 Cert. #4249 - *Nutanix Cryptographic Module for OpenSSL*. This security policy represents the complete, composite functionality of this module: the SSHv2 service implemented using the cryptographic primitives of Cert. #4249. Unlike Cert. #4249, the module does not offer a set of cryptographic primitives, rather it provides only the SSHv2 subsystem. Specifically:

- The logical boundary of this module is shown in Figure 1 below.
- Table 4 lists all approved cryptographic algorithms used by this module, inclusive of those implemented in Cert. #4249.
- The CSPs and public keys listed in this security policy are those accessed by this module.
- The module provides the software integrity test described in Section 6.
- All key generation and the associated DRBG and entropy are managed by Cert. #4249.

The module operates within a general-purpose computer. Figure 1 depicts the module operational environment, with the logical boundary highlighted in red inclusive of all module entry points (API calls), conformant with [140IG] 14.3 *Logical Diagram for Software, Firmware and Hybrid Modules*.

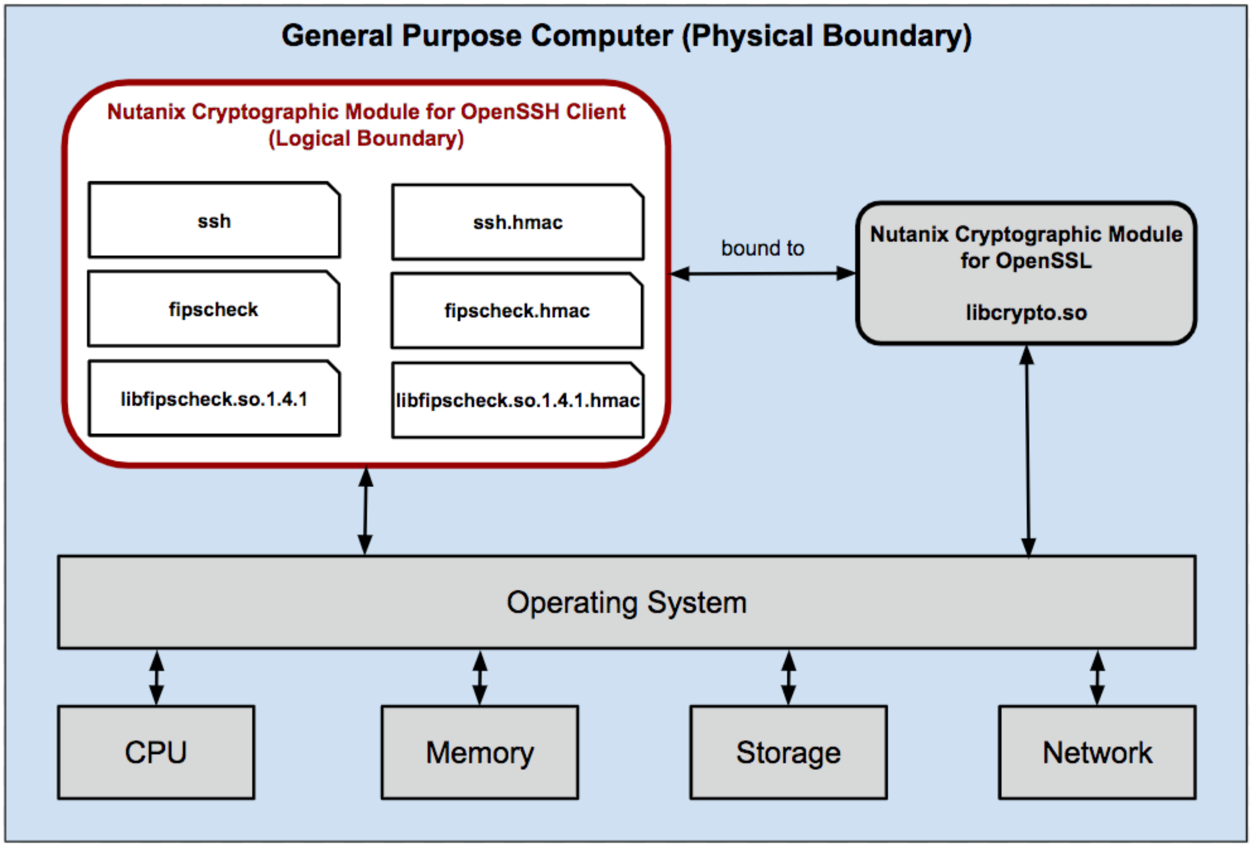

Figure 1: Module Physical and Logical Boundary

The module conforms to [140IG] 1.16 *Software Module*:

- The physical cryptographic boundary is the general-purpose computer which wholly contains the module and operating system.
- The logical cryptographic boundary is the set of shared library files and associated HMAC files:
  - ssh, ssh.hmac
  - fipscheck, fipscheck.hmac
  - libfipscheck.so.1.4.1, libfipscheck.so.1.4.1.hmac
- All components are defined in accordance with [140DTR] AS01.08; no components are excluded from [140] requirements.
- The power-up approved integrity test is performed over all components within the logical boundary.
- Updates to the module are provided as a complete replacement in accordance with [140IG] 9.7 *Software/Firmware Load Test*.
- Table 2 defines the module's [140] logical interfaces.

Table 2: Ports and Interfaces

Physical Port / Description	Logical Interface Type
Terminal I/O (command line in, display out)	Control input, status output
File I/O (Configuration, key and log files)	Control input, data input, status output
TLS module API (calls to libcrypto.so services)	Control input, data input, data output, Status output
Network traffic	Control input, data input, data output, Status output

Operational testing was performed on the Operating Environments listed in Table 3.

Table 3: Tested Operating Environments

Operating System	Processor	Platform
CentOS 7.9 on Nutanix Acropolis Hypervisor (AHV) 7.1.1	Intel Xeon Gold 6234 with PAA	Nutanix NX-3360-G7
CentOS 7.9 on Nutanix Acropolis Hypervisor (AHV) 7.1.1	Intel Xeon Gold 6234 without PAA	Nutanix NX-3360-G7
CentOS 7.9	Intel Xeon Gold 6234 with PAA	Nutanix NX-3360-G7
CentOS 7.9	Intel Xeon Gold 6234 without PAA	Nutanix NX-3360-G7

The module conforms to [140IG] 6.1 *Single Operator Mode and Concurrent Operators*. The tested environments place user processes into segregated spaces. A process is logically removed from all other processes by the hardware and Operating System. Since the module exists inside the process space of the application this environment implicitly satisfies the requirement for a single user mode.

2 Cryptographic Functionality

The module implements the FIPS Approved cryptographic functions listed in Table 4. [57P1] notation is used throughout this document to describe key sizes and security strength. Items in curly brackets { } are tested but not used by the module.

Table 4: Approved CAVP Validated Cryptographic Functions

Cert	Algorithm	Mode	Description	Functions, Caveats
Implemented by this module (Nutanix Cryptographic Module for OpenSSH Client)				
A1405	CVL: SSH KDF [135]	SSHv2 key derivation Cipher: AES-128, AES-192, AES-256, {TDES} Hash: SHA-1, SHA2-256, SHA2-384, SHA2-512		Key derivation
Implemented by Cert. #4249 (Nutanix Cryptographic Module for OpenSSL)				
A1403	AES [197]	CBC, CTR [38A]	Key sizes: 128, 192, 256 (bits)	Encryption, Decryption
		CCM [38C]	Key sizes: 128, 192, 256 (bits)	Authenticated Encryption, Authenticated Decryption
		GCM [38D]	Key sizes: 128, {192}, 256 (bits)	Authenticated Encryption, Authenticated Decryption, Message Authentication
A1403	DRBG [90A]	CTR_DRBG {Hash_DRBG} {HMAC_DRBG}	AES: {128, 192}, 256 (bits) {Other DRBG variants and strengths are tested on the bound module cert but are not used by this module}	Random bit generation
A1403	ECDSA [186]	P-256, P-384, P-521 with SHA-2 (-224, -256, -384, -512) Legacy use: P-256, P-384, P-521 with SHA-1 (Signature Verification only)		Signature Generation, Signature Verification {Key generation and key verification are not used}
A1403	HMAC [198]	HMAC-SHA-1, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512 {Other SHA sizes are tested on the bound module cert but are not used by this module}		MAC Generation, MAC Verification

Cert	Algorithm	Mode	Description	Functions, Caveats
A1403, A1405	KAS [56Ar3] (KAS-SSC [56Ar3] with CVL [135])	KAS ECC SSC (ephemeralUnified)	P-256, P-384, P-521[1]	Per [140IG] D.8 Scenario X1 path (2), [56Ar3] compliant key agreement scheme where testing is performed separately for the shared secret computation and a KDF compliant with [135]. No key confirmation.
		KAS FFC SSC (dhEphem)	FB: L ≥ 2048 N = 224 FC: L ≥ 2048 N = 256[2]	
		SSHv2 KDF	Cipher: AES-128, AES-192, AES-256, {TDES} Hash: SHA-1, SHA2-256, SHA2-384, SHA2-512	
A1403	KAS-SSC [56Ar3]	KAS ECC SSC (ephemeralUnified)	P-256, P-384, P-521[1]	Key Agreement: Shared Secret Calculations.
		KAS FFC SSC (dhEphem)	FB: $L \geq 2048$ $N = 224$ FC: $L \geq 2048$ $N = 256$[2]	
A1403	KTS [38F]	AES-GCM	Key sizes: 128, {192}, 256 (bits)	Key establishment methodology provides 128 or 256 bits of encryption strength
A1403	KTS [38F]	AES-CBC or AES-CTR with HMAC	Key sizes: 128, 192, 256 (bits)	Key establishment methodology provides between 128 and 256 bits of encryption strength
A1403	SHS [180]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		Message Digest

The module conforms to [140IG] D.11 *References to the Support of Industry Protocols* (Resolution scenario 2) by providing CAVP validated [56Ar3] components along with the CAVP validated [135] Section 4.2 KDF for SSHv2. In accordance with [140IG] D.11, the remainder of the SSHv2 protocol has not been reviewed or tested by the CAVP and CMVP.

The module does not implement any non-Approved but allowed cryptographic functions.

3 Modes of Operation, Security Rules and Guidance

The module supports a FIPS Approved mode of operation and a non-FIPS Approved mode of operation, and conforms to [140IG] 1.2 *FIPS Approved Mode of Operation* and 1.19 *non-Approved Mode of Operation*.

The module only comes pre-installed on the Nutanix NX-3360-G7. The operator does not need to take any action to install the Nutanix module. The operator can confirm the module is running in Approved mode by using the command **sudo grep -r "FIPS" /var/log.**

The conditions for using module cryptographic primitives in the [140] Approved mode of operation are:

1. Only the ssh parameters listed in Table 5 are to be used. Key pairs generated external to the module for use with key exchange and client authentication must comply to the Table 5 parameters.
2. SSH protocol version 1 is not usable in the approved mode; SSH protocol version 2 must be enabled.
3. GSS API authentication methods (e.g., Kerberos) are not allowed.
4. Cryptographic methods not allowed under [140] (e.g., ED25519 and related methods) must not be used.
5. If module power is lost and restored, a new key for use with AES GCM cipher is established.

[1] KAS-ECC-SSC: key establishment methodology provides between 128 and 256 bits of encryption strength.
[2] KAS-FFC-SSC: key establishment methodology provides 112 bits of encryption strength.

6. Data output is inhibited during self-tests, zeroization, and error states. The module runs in a single-threaded process that inhibits data output when in the self-test state by preventing invocation of any other cryptographic functions via the API.
7. Status information does not contain CSPs or sensitive data that if misused could lead to module compromise.
8. CSPs defined in an Approved mode of operation are not to be accessed or shared while in a non-Approved mode of operation. CSPs shall not be generated while in a non-approved mode.

Table 5: SSH Parameters Used in the Approved Mode of Operation

Key exchange	Client authentication	Cipher	MAC
diffie-hellman-group14-sha1	ecdsa-sha2-nistp256	aes256-cbc	hmac-sha1
diffie-hellman-group14-sha256	ecdsa-sha2-nistp384	aes192-cbc	hmac-sha2-256
diffie-hellman-group-exchange-sha256	ecdsa-sha2-nistp521	aes128-cbc	hmac-sha2-512
ecdh-sha2-nistp256		aes128-ctr	hmac-sha1-etm[3]
ecdh-sha2-nistp384		aes192-ctr	hmac-sha2-256-etm[3]
ecdh-sha2-nistp521		aes256-ctr	hmac-sha2-512-etm[3]
		aes128-gcm	
		aes256-gcm	

The SSH parameters are set to the values above via the default settings in the /etc/ssh/ssh_config configuration file. To confirm the settings for the above SSH parameters, use ssh with the -G option, e.g., ssh -G <ip address>. Note that the string “@openssh.com” is appended to some SSH parameters in the sets above; these algorithms are not named in IANA sources, but remain compliant with [140]. Note also that the ssh -Q (query) option reports on a static list of compiled options and yields misleading results. Accompany “ssh -G” with the *grep* command to filter output for the information of interest, for example (where the IP 192.168.1.1 represents a generic example IP address):

```
ssh -G 192.168.1.1 | grep '^kexalgo\|ciphers\|macs\|hostkeyalgo'
```

Note that GCM ciphers provide message integrity as well as confidentiality but have no explicit listing in the list of MACs.

Table 6 lists ciphersuites available in the non-approved mode of operation. The module has two mechanisms for determining available parameters: the majority are controlled by the module’s query of the OpenSSL libcrypto *fips_mode* function, which when set to ‘true’ limits the use to approved algorithms. Use of parameters such as the diffie-hellman-group16-sha512 (with a larger but technically unrecognized modulus size) is limited by the default ssh configuration as described in [UG].

Table 6: SSH Parameters Used in the Non-Approved Mode of Operation

Key exchange	Client authentication	Cipher	Digest
diffie-hellman-group1-sha1	sha-rsa	rijndael-cbc	All are approved
diffie-hellman-group-exchange-sha1	ssh-ed25519	3des-cbc	
curve25519-sha256	ssh-dss		
diffie-hellman-group16-sha512[4]	ssh-rsa-cert-v01		
diffie-hellman-group18-sha512[5]	ssh-dss-cert-v01		
gss-gex-sha1-, gss-group1-sha1, gss-group14-sha1			

[3] *etm* refers to “encrypt then MAC” - the HMAC is performed on the ciphertext message content
[4] Uses a 4096-bit modulus: sizes over 2048 bits are allowed but are not defined by [56Ar3]
[5] Uses a 8192-bit modulus: sizes over 2048 bits are allowed but are not defined by [56Ar3]

4 Critical Security Parameters and Public Keys

All CSPs and public keys used by the module are described in this section. The module does not implement random number generation or key pair generation, instead using those provided by the bound OpenSSL libcrypto.so services. The module and an external ssh daemon perform the SSH protocol, the module calls the libcrypto shared secret generation primitive, then calls its SSHv2 KDF to derive the session keys from the shared secret. The session keys are provided to libcrypto.so for message cipher (encrypt/decrypt) and integrity functions. Table 7 summarizes the CSPs and public keys implemented by the module.

Table 7: Critical Security Parameters (CSPs) and Public Keys

CSP	Description/Usage	Key generation	Key storage	Key entry/output*	Key Zeroization
Shared secret	Secure communications shared secret, used to derive session keys	Obtained from call to libcrypto.so	Module memory	Entry: N/A Output: N/A	Overwritten with zeros
Session keys	Secure communications session keys used for message ciphering and integrity	Derived by the module using the SSHv2 KDF	Module memory	Entry: N/A Output: N/A	Overwritten with zeros
Client key pairs	ECC or FFC key pair used for key exchange; ECC key pair used key for client authentication	N/A	Module memory†	Entry: N/A Output: public key to peer	Module removal
Server public key	ECC or FFC key used for key exchange	N/A	Module memory	Entry: From peer* Output: N/A	N/A
Software integrity key	Verification of software file integrity	N/A	Module binary	Entry: N/A Output: N/A	Module removal

Shared secret and *Session keys* are overwritten with zeros automatically after use (session key derivation and channel closure, respectively).

Session message cipher key: AES-128, AES-192, or AES-256 key.

Session message integrity key: HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512, or AES-128 or AES-256 (GCM). HMAC message authentication key lengths are equal to the digest size.

Client key pairs and server public key: ECC (P-256, P-384, P-521).
† The ECDSA public key is obtained from the local file system.

Software integrity key: HMAC-SHA-256 (128-bit key).

**Key entry/output* is interpreted based on [140IG] 7.7; no CSPs cross the physical boundary, rather, local private and public keys are obtained from files within the boundary; and the shared secret is obtained by API interaction with the libcrypto.so within the boundary. Public keys are exchanged over the network port in the SSH handshake.

5 Roles and Services

The module supports two distinct operator roles, User and Cryptographic Officer (CO), and does not support multiple concurrent operators, a maintenance role or bypass capability. The Module does not provide an authentication or identification method of its own. The CO and the User roles are implicitly identified by the service requested.

All services implemented by the module are listed in Table 8. This table describes module service access to CSPs and public keys, where ‘I’ indicates *input*, ‘O’ indicates *output*, ‘G’ indicates *generate* (use of the SSHv2 KDF), ‘X’ indicates *execute* (use of the CSP or public key by a function), and ‘Z’ indicates *zeroize*.

Table 8: Authorized Services Available in FIPS Mode

Service	Description	Role	CSP and public key access
Install	Install the module (performed at the factory)	CO	N/A
Configure	Configure the module	CO	N/A
Launch (Self-Test)	Launch the ssh application (includes self-test)	User	Software integrity key: X
Secure communications	Establish and use ssh secure communications channel	User	Client private key: X Client public key: O Server public key: I, X Shared secret: X, Z Session keys: G, X
Close	Close ssh secure communications channel	User	Session keys: Z
Show status	Show ssh status (ssh -v)	User	N/A
Terminate (Zeroize)	Shut down the ssh application	User	Shared secret: Z Session keys: Z

6 Self-Test

The module performs the software integrity check using HMAC-SHA-256 on initialization of the ssh executable. This is executed without operator intervention. The ssh application calls the fipscheck utility via the fipscheck.so library, which tests the fipscheck binary, the fipscheck shared library and the calling application binary (ssh). The fipscheck utility returns a non-zero error code on failure, with a corresponding error log message “FIPS Integrity verification test failed”, and on successful completion “FIPS mode initialized”.

All algorithms provided by the Nutanix Cryptographic Module for OpenSSL (the cryptographic primitives library this module is bound to) are self-tested when the library (libcrypto.so) is loaded.

To execute the power-on tests on demand, the module must be reloaded.

The module returns an error code and enters the error state if any of the power-on self-tests fail. When the module enters the error state, cryptographic operations are no longer possible.

The fipscheck utility exit value indicates the comparison result, either zero if the integrity tests succeed, or an error code if any of the integrity tests fail (resulting in termination of the ssh application).