OpenSSH Server Cryptographic Module

versions 1.0, 1.1 and 1.2

FIPS 140-2 Non-Proprietary Security Policy

Version 3.0

Last update: 2021-01-13

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

Table of Contents

1. Cryptographic Module Specification 4
1.1. Module Overview 4
1.2. Modes of Operation 8
2. Cryptographic Module Ports and Interfaces 9
3. Roles, Services and Authentication 10
3.1. Roles 10
3.2. Services 10
3.3. Algorithms 12
3.4. Operator Authentication 15
4. Physical Security 16
5. Operational Environment 17
5.1. Applicability 17
5.2. Policy 17
6. Cryptographic Key Management 18
6.1. Random Number Generation 19
6.2. Key Generation 19
6.3. Key Derivation 19
6.4. Key Entry / Output 19
6.5. Key / CSP Storage 19
6.6. Key / CSP Zeroization 19
7. Electromagnetic Interference / Electromagnetic Compatibility (EMI/EMC) 20
8. Self-Tests 21
8.1. Power-Up Tests 21
8.1.1. Integrity Tests 21
8.1.2. Cryptographic Algorithm Tests 21
8.2. On-Demand Self-Tests 21
8.3. Conditional Tests 21
9. Guidance 22
9.1. Crypto Officer Guidance 22
9.1.1. Operating Environment Configurations 23
9.1.2. Module Installation 23
9.2. User Guidance 24
9.2.1. Starting an OpenSSH Server 24
9.2.2. Handling Self-Test Errors 25
10. Mitigation of Other Attacks 26

Copyrights and Trademarks

1. Cryptographic Module Specification

This document is the non-proprietary FIPS 140-2 (Federal Information Processing Standards Publication 140-2) Security Policy for version 1.0 of the Ubuntu OpenSSH Server Cryptographic Module, which is based on the openssh-7.2p2-4ubuntu2.fips.2.2.1 deliverable package, version 1.1 of the Ubuntu OpenSSH Server Cryptographic Module, which is based on the openssh-7.2p2-4ubuntu2.fips.2.8.1 deliverable package and version 1.2 of the Ubuntu OpenSSH Server Cryptographic Module, which is based on the openssh-7.2p2-4ubuntu2.fips.2.10.1 deliverable package.
It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-2 for a Security Level 1 module.

The following sections describe the cryptographic module and how it conforms to the FIPS 140-2 specification in each of the required areas.

1.1. Module Overview

The Ubuntu OpenSSH Server Cryptographic Module version 1.0, Ubuntu OpenSSH Server Cryptographic Module version 1.1 and Ubuntu OpenSSH Server Cryptographic Module version 1.2, (also referred to as "the module") is a server daemon implementing the Secure Shell (SSH) protocol in the Ubuntu Operating System user space. The module interacts with other entities acting as SSH clients via the SSH protocol. The module only supports SSHv2 protocol.

The module uses the Ubuntu OpenSSL Cryptographic Module as a bound module (also referred to as "the bound OpenSSL module"), which provides the underlying cryptographic algorithms necessary for establishing and maintaining SSH sessions. The Ubuntu OpenSSL Cryptographic Module is a FIPS validated module (certificates #2888 and #3725). Software versions 1.0 and 1.1 are bound to Ubuntu OpenSSL with certificate #2888 and software version 1.2 is bound to Ubuntu OpenSSL with certificate #3725.

For the purpose of the FIPS 140-2 validation, the module is a software-only, multi-chip standalone cryptographic module validated at overall security level 1. The table below shows the security level claimed for each of the eleven sections that comprise the FIPS 140-2 standard:

FIPS 140-2 Section		Security Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services and Authentication	1
4	Finite State Model	1
5	Physical Security	N/A
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self-Tests	1

FIPS 140-2 Section		Security Level
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A
Overall Level		1

Table 1 - Security Levels

The cryptographic logical boundary consists of the SSH daemon and its integrity verification file. The following table enumerates the files that comprise each module variant:

Component	Description
/usr/sbin/sshd	SSH daemon
/usr/sbin/.sshd.hmac	Integrity verification file for SSH daemon

Table 2 - Cryptographic Module Components

The software block diagram below shows the module, its interfaces with the bound OpenSSL module, the operational environment and the delimitation of its logical boundary, which is depicted in the blue box:

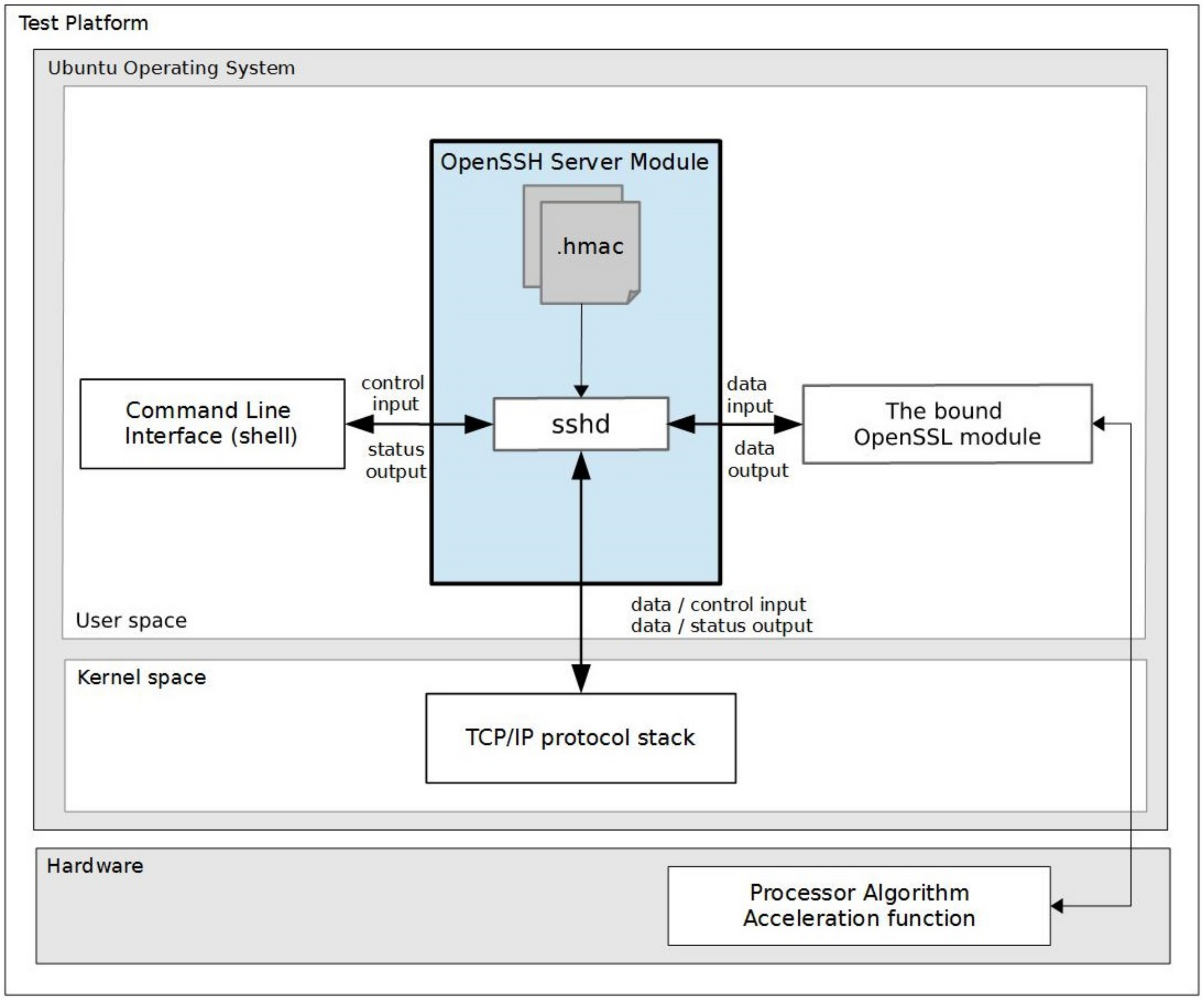

Figure 1 - Software Block Diagram

The module is aimed to run on a general purpose computer (GPC); the physical boundary is the surface of the case of the tested platforms, as shown in the diagram below:

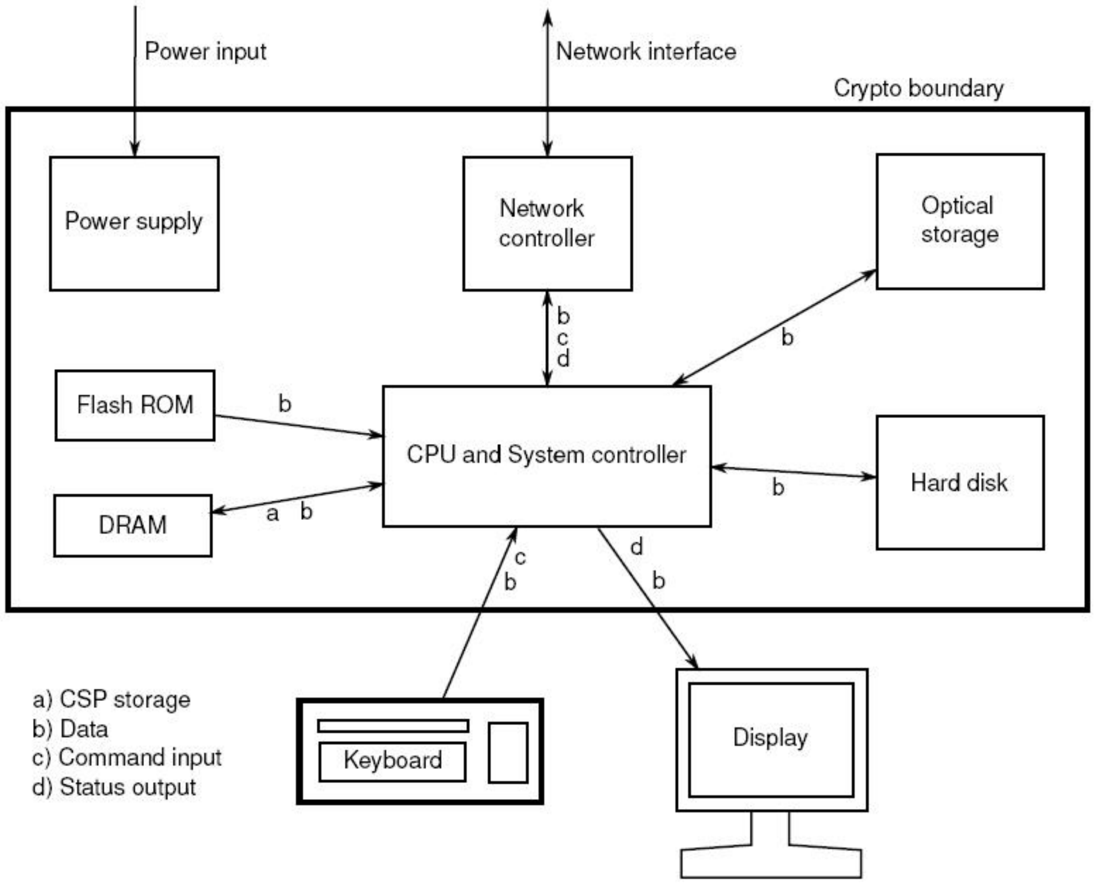

Figure 2 - Cryptographic Module Physical Boundary

Ubuntu OpenSSH Server Cryptographic Module have been tested on the platforms shown below:

Test Platform	Processor	Test Configuration	Version
IBM Power System S822L (PowerNV 8247-22L)	POWER8	Ubuntu 16.04 LTS 64-bit Little Endian with/ without Power ISA 2.07 (PAA)	1.0 and 1.1
IBM Power System S822LC (PowerNV 8001-22C)	POWER8	Ubuntu 16.04 LTS 64-bit Little Endian with/without Power ISA 2.07 (PAA)	1.0 and 1.1
IBM Power System S822LC (PowerNV 8335-GTB)	POWER8	Ubuntu 16.04 LTS 64-bit Little Endian with/without Power ISA 2.07 (PAA)	1.0 and 1.1
Supermicro SYS-5018R-WR	Intel® Xeon® CPU E5-2620v3	Ubuntu 16.04 LTS 64-bit with/without AES-NI (PAA)	1.0, 1.1, and 1.2
IBM z13	z13	Ubuntu 16.04 LTS 64-bit running on LPAR with/without CPACF (PAI)	1.0 and 1.1

Table 3 - Tested Platforms

1.2. Modes of Operation

The module supports two modes of operation:

- **"FIPS mode"** (the Approved mode of operation): only approved or allowed security functions with sufficient security strength can be used.
- **"non-FIPS mode"** (the non-Approved mode of operation): only non-approved security functions can be used.

The module enters FIPS mode after power-up tests succeed. Once the module is operational, the mode of operation is implicitly assumed depending on the security function invoked and the security strength of the cryptographic keys.
Critical security parameters used or stored in FIPS mode are not used in non-FIPS mode and vice versa.

2. Cryptographic Module Ports and Interfaces

As a software-only module, the module does not have physical ports. For the purpose of the FIPS 140-2 validation, the physical ports are interpreted to be the physical ports of the hardware platform on which it runs.

The logical interfaces are the sshd command, the messages sent to and received from the SSH client, and the application program interface (API) provided by the bound OpenSSL module. The following table summarizes the four logical interfaces:

FIPS Interface	Physical Port	Logical Interface
Data Input	Keyboard, Ethernet port	Input parameters and data sent to the sshd command through the command line with the host key files in /etc/ssh directory, ~/.ssh/authorized_keys, input data received via the SSHv2 channel, input data received via local or remote port-forwarding port, input data received from the bound OpenSSL module via its API parameters.
Data Output	Display, Ethernet port	Output data returned by the sshd command, output data sent via the SSHv2 channel, output data sent via local or remote port-forwarding port, output data sent to the bound OpenSSL module via its API parameters.
Control Input	Keyboard, Ethernet port	Invocation of the sshd command on the command line, control parameters via the sshd command or via the /etc/ssh/sshd_config file, SSHv2 protocol message requests received from SSH client.
Status Output	Display, Ethernet port	Status messages returned after execution of sshd command, status of processing SSHv2 protocol message requests.
Power Input	PC Power Supply Port	N/A

Table 4 - Ports and Interfaces

3. Roles, Services and Authentication

3.1. Roles

The module supports the following roles:

- **User role:** performs services of establish, maintain and close SSH session, show status and self-tests.
- **Crypto Officer role:** performs services of module installation, configuration and terminate SSH daemon.

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

3.2. Services

The module provides services to users that assume one of the available roles. All services are shown in Table 5 and Table 6, and described in detail in the user documentation.

Table 5 shows the Approved services in FIPS mode, the cryptographic algorithms supported for the service, the roles to perform the service, the cryptographic keys or Critical Security Parameters (CSPs) involved and how they are accessed. Please see Appendix A of this document for the complete list of supported cipher suites by the module in FIPS mode.

Note: In Table 5, only the NIST SP800-135 SSH Key Derivation Function (KDF) algorithm is provided by the Ubuntu OpenSSH Server Cryptographic Module, and the Cryptographic Algorithm Validation Program (CAVP) certificate numbers are listed in Table 7(A). All the other cryptographic algorithms listed in Table 5 are provided by the bound OpenSSL module, and the CAVP certificate numbers are listed in Table 7(B).

Service	Algorithms	Role	Access	Key / CSP
Establish SSH Session	Key exchange: • Diffie-Hellman with SHA-256 and key size between 2048-bit and 8192-bit[1]; • EC Diffie-Hellman with SHA-256/SHA-384/SHA-512 and curve P-256/P-384/P-521[2]	User	Create	Server's Diffie-Hellman or EC Diffie-Hellman public and private keys; Client's Diffie-Hellman or EC Diffie-Hellman public keys; Diffie-Hellman or EC Diffie-Hellman shared secret
	Key derivation: SP800-135 SSH KDF with SHA-	User	Read	Diffie-Hellman or EC Diffie-Hellman shared secret

[1] Diffie-Hellman key agreement provides between 112 and 192 bits of encryption strength.
[2] EC Diffie-Hellman key agreement provides between 128 and 256 bits of encryption strength.

Service	Algorithms	Role	Access	Key / CSP
	1/SHA-256/SHA-384/SHA-512		Create	Session encryption keys (128/192/256-bit AES keys, 192-bit Triple-DES keys); Session data authentication keys (at least 112-bit HMAC keys)
Maintain SSH Session	Data Encryption and Decryption: AES (CBC and CTR modes), Triple-DES (CBC mode)	User	Read	Session encryption keys
	Data Integrity (MAC): HMAC with SHA-1/SHA-256/SHA-512	User	Read	Session data authentication keys
Server Signature Generation	Signature generation used in SSH key-based authentication and certificate-based authentication: • ECDSA with SHA-256/SHA-384/SHA-512 and curve P-256/P-384/P-521; • RSA with SHA-256/SHA-512 and key size between 2048-bit and 16384-bit	User	Read	Server's private keys; Server's public keys
Client Signature Verification	Signature verification used in SSH key-based authentication and certificate-based authentication: • DSA with SHA-1 and 1024-bit key size; • ECDSA with SHA-256/SHA-384/SHA-512 and curve P-256/P-384/P-521; • RSA with SHA-256/SHA-512 and key size between 1024-bit and 16384-bit	User	Read	Client's public keys
Close SSH Session	N/A	User	Zeroize	All aforementioned CSPs

Service	Algorithms	Role	Access	Key / CSP
Terminate SSH Daemon (sshd)	N/A	Crypto Officer	Zeroize	All aforementioned CSPs
Configure SSH Server	N/A	Crypto Officer	N/A	None
Show status	N/A	User	N/A	None
Self-test	SP800-135 SSH KDF and HMAC	User	N/A	None
Module installation	N/A	Crypto Officer	N/A	None

Table 5 - Services in FIPS mode

Table 6 lists the service that uses the non-Approved algorithms or non-compliant key sizes, which cause the module to transition to the non-FIPS mode implicitly.

Note: In Table 6, only the Ed25519 signature generation and verification algorithm is provided by the Ubuntu OpenSSH Server Cryptographic Module. All the other cryptographic algorithms listed in Table 6 are provided by the bound OpenSSL module.

Service	Algorithms / Key sizes	Role
Server or Client Digital Signature using non-Approved algorithms or key sizes	Digital signature used in SSH key-based authentication and certificate-based authentication: • DSA signature generation with SHA-1 and 1024-bit key; • RSA signature generation with SHA-1 and/or, RSA signature generation with key size equal to or greater than 768-bit and smaller than 2048-bit, RSA signature verification with key size equal to or greater than 768-bit and smaller than 1024-bit; • Ed25519 signature generation and verification with curve25519	User

Table 6 - Services in non-FIPS mode

3.3. Algorithms

Table 7(A) shows the Approved algorithm provided by the module in FIPS mode, which is tested and validated by CAVP.

Algorithm	CAVP Cert.	Standard	Use
SP800-135 SSH KDF with SHA-1/SHA-256/SHA-384/SHA-512	CVL #1085, #1087, #1088, #1090, #1091	[SP800-135]	Key derivation in the SSHv2 protocol.
SP800-135 SSH KDF with SHA-256/SHA-384/SHA-512	CVL #1086		
SP800-135 SSH KDF with SHA-1/SHA-256	CVL #1089		

Table 7(A) - Approved Algorithms provided by the OpenSSH Server Module

Table 7(B) shows the Approved and non-Approved but Allowed algorithms that are used by the module, but provided by the bound OpenSSL module in FIPS mode. The CAVP certificates are listed if applicable.

Algorithm	CAVP Cert. for #2888 used by versions 1.0 and 1.1	CAVP Cert. for #3725 used by version 1.2
AES	#4354, #4355, #4356, #4357, #4358, #4359, #4360, #4361	#C1258, #C1259, #C1260, #C1261, #C1264, #C1265, #C1266, #C1267, #C1270
Partial Diffie-Hellman	CVL #1053, #1056, #1059, #1062, #1065, #1067, #1069	CVL #C1269, #C1304 and #C1305
Partial EC Diffie-Hellman	CVL #1053, #1056, #1059, #1063, #1065, #1068, #1069	CVL #C1269, #C1304 and #C1305
ECC CDH Primitive	CVL #1054, #1057, #1060, #1063, #1065, #1067, #1069	CVL #C1269, #C1304 and #C1305
DRBG	#1390, #1391, #1392, #1393, #1394, #1395, #1396, #1397	#C1269, #C1304, #C1305
DSA	#1156, #1157, #1158, #1159, #1160, #1161, #1162	#C1269, #C1304, #C1305
ECDSA	#1031, #1032, #1033, #1034, #1035, #1036, #1037	#C1269, #C1304, #C1305
HMAC	#2895, #2896, #2897, #2898, #2899, #2900, #2901	#C1269, #C1304, #C1305
RSA	#2351, #2352, #2353, #2354, #2355, #2356, #2357	#C1269, #C1304, #C1305
SHS	#3593, #3594, #3595, #3596, #3597, #3598, #3599	#C1269, #C1304, #C1305
Triple-DES	#2355, #2356, #2357	#C1257

Algorithm	CAVP Cert. for #2888 used by versions 1.0 and 1.1	CAVP Cert. for #3725 used by version 1.2
Diffie-Hellman key agreement (non-approved but allowed in [FIPS140-2_IG] D.8)	CVL #1053, #1056, #1059, #1062, #1065, #1067, #1069	CVL #C1269, #C1304 and #C1305
EC Diffie-Hellman key agreement (non-approved but allowed in [FIPS140-2_IG] D.8)	CVL #1053, #1054, #1056, #1057, #1059, #1060, #1063, #1065, #1067, #1068, #1069	CVL #C1269, #C1304 and #C1305
NDRNG (non-approved but allowed to seed the DRBG)	N/A	N/A
RSA signature verification with key size greater than 3072 bits, and signature generation with key size greater than 4096 bits (non-approved but allowed in [SP800-131A])	N/A	N/A

Table 7(B) – Approved or Allowed Algorithms provided by the bound OpenSSL Module

Table 8(A) shows the non-Approved algorithm provided by the module in non-FIPS mode.

Algorithm	Use
Ed25519 signature generation and verification with curve25519	Signature generation and verification

Table 8(A) - Non-Approved Algorithms provided by the OpenSSH Server Module

The OpenSSH and the bound OpenSSL module together provide the Diffie Hellman and EC Diffie Hellman key agreement. The OpenSSH module only implements the KDF portion of the key agreement as stated in the above table and the bound OpenSSL module provides the shared secret computation:

- Diffie-Hellman (CVL Certs. #1053, #1056, #1059, #1062, #1065, #1067, #1069, #C1269, #C1304 and #C1305 with CVL Certs. #1085, #1086, #1087, #1088, #1089, #1090 and #1091 key agreement; key establishment methodology provides between 112 and 192 bits of encryption strength);

- EC Diffie-Hellman (CVL Certs. #1053, #1054, #1056, #1057, #1059, #1060, #1063, #1065, #1067, #1068, #1069, #C1269, #C1304 and #C1305 with CVL Certs. #1085, #1086, #1087, #1088, #1089, #1090 and #1091 key agreement; key establishment methodology provides between 128 and 256 bits of encryption strength);

Table 8(B) shows the non-Approved algorithms that are used by the module, but provided by the bound OpenSSL module in non-FIPS mode.

Algorithm	Use
DSA signature generation with SHA-1 and 1024-bit key	Signature generation
RSA signature generation with SHA-1 and/or, RSA signature generation with key size smaller than 2048-bit, RSA signature verification with key size smaller than 1024-bit	Signature generation and verification

Table 8(B) - Non-Approved Algorithms provided by the bound OpenSSL Module

3.4. Operator Authentication

The module does not implement operator authentication. The role of the user is implicitly assumed based on the service requested.

4. Physical Security

The module is comprised of software only and therefore this security policy does not make any claims on physical security.

5. Operational Environment

5.1. Applicability

The module operates in a modifiable operational environment per FIPS 140-2 level 1 specifications. The module runs on a commercially available general-purpose operating system executing on the hardware specified in Table 3.

5.2. Policy

The operating system is restricted to a single operator (concurrent operators are explicitly excluded). The application that requests cryptographic services is the single user of the module.

6. Cryptographic Key Management

The following table summarizes the cryptographic keys and CSPs that are used by the cryptographic services implemented in the module:

Name	Generation	Entry and Output	Zeroization
Server's private keys (ECDSA, RSA)	N/A	Entry: The keys are read from the host key files. Output: The keys are output to the bound OpenSSL module via API parameters.	Zeroized automatically when closing SSH session or terminating SSH daemon.
Server's public keys (ECDSA, RSA)	N/A	Entry: The keys are read from the host key files. Output: The keys are output during SSH handshake.	
Client's public keys (DSA, ECDSA, RSA)	N/A	Entry: The keys are entered during SSH handshake. Output: The keys are output to the bound OpenSSL module via API parameters.	
Server's Diffie-Hellman or EC Diffie-Hellman public keys	N/A (generated by the bound OpenSSL module)	Entry: The keys are entered from the bound OpenSSL module via API parameters. Output: The keys are output during SSH handshake.	
Server's Diffie-Hellman or EC Diffie-Hellman private keys	N/A (generated by the bound OpenSSL module)	Entry: The keys are entered from the bound OpenSSL module via API parameters. Output: The keys are output to the bound OpenSSL module via API parameters.	
Client's Diffie-Hellman or EC Diffie-Hellman public keys	N/A	Entry: The keys are entered during SSH handshake. Output: The keys are output to the bound OpenSSL module via API parameters.	
Diffie-Hellman or EC Diffie-Hellman shared secret	N/A (generated by the bound OpenSSL module)	Entry: The CSPs are entered from the bound OpenSSL module via API parameters. Output: N/A	
Session encryption keys (AES, Triple-DES)	N/A (derived from the shared secret via SP800-	Entry: N/A Output: The keys are output to the	

	135 SSH KDF)	bound OpenSSL module via API parameters.	
Session data authentication keys (HMAC)	N/A (derived from the shared secret via SP800-135 SSH KDF)	Entry: N/A Output: The keys are output to the bound OpenSSL module via API parameters.	

Table 9 - Life cycle of Keys/CSPs

The following sections describe how CSPs, in particular cryptographic keys, are managed during their life cycle.

6.1. Random Number Generation

The module does not implement any random number generator. Instead, it uses the Random Number Generation service provided by the bound OpenSSL module, which implements a Deterministic Random Bit Generator (DRBG) based on [SP800-90A].

6.2. Key Generation

The module does not implement key generation.

6.3. Key Derivation

The module implements SP800-135 SSH KDF for the SSHv2 protocol.

6.4. Key Entry / Output

The module does not support manual key entry or intermediate key generation key output. The keys are entered from or outputted to the module electronically.

6.5. Key / CSP Storage

The module does not perform persistent storage of keys. The keys and CSPs are temporarily stored as plaintext in the RAM.
The server's public and private keys are stored in the host key files in /etc/ssh directory, which are within the module's physical boundary but outside its logical boundary.
The HMAC key used for the Integrity Test is stored in the module and relies on the operating system for protection.

6.6. Key / CSP Zeroization

The memory occupied by keys is allocated by regular memory allocation operating system calls. The module calls appropriate key zeroization functions provided by the bound OpenSSL module, which overwrite the memory occupied by keys with "zeros" and deallocate the memory with the regular memory deallocation operating system call. The module also provides the key zeroization method to overwrite the memory occupied by keys with "zeros" when the module receives internal error codes or the module is terminated.

7. Electromagnetic Interference / Electromagnetic Compatibility (EMI/EMC)

The test platforms listed in Table 3 have been tested and found to conform to the EMI/EMC requirements specified by 47 Code of Federal Regulations, FCC PART 15, Subpart B, Unintentional Radiators, Digital Devices, Class A (i.e., Business use). These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. Each of the test platforms shall be installed and used in accordance with its instruction manual.

8. Self-Tests

8.1. Power-Up Tests

The module performs power-up tests when it is loaded into memory without operator intervention. Power-up tests ensure that the module is not corrupted and that the cryptographic algorithms work as expected.
While the module is executing the power-up tests, services are not available, and input and output are inhibited. The module is not available for use until the power-up tests complete successfully. If any power-up test fails, the module will return the error message listed in section 9.2.2 and enter the error state. In error state, the SSH daemon is terminated and thus no cryptographic operations or data output are possible.
Note: The bound OpenSSL module performs its own power-up tests automatically when it is loaded into memory. The Ubuntu OpenSSH Server Cryptographic Module ensures that the bound OpenSSL module must complete its power-up tests successfully.

8.1.1. Integrity Tests

The integrity of the module is verified by comparing an HMAC-SHA-256 value calculated at run time with the HMAC value stored in the .hmac file that was computed at build time. The HMAC-SHA-256 algorithm is provided by the bound OpenSSL module. If the HMAC values do not match, the test fails and the module enters the error state.

8.1.2. Cryptographic Algorithm Tests

The module performs the self-test on the following FIPS-Approved cryptographic algorithm supported in FIPS mode using the known answer test (KAT) shown below:

Algorithm	Test
SP800-135 SSH KDF	KAT KDF for SSH

Table 10- Self-Tests

For the KAT, the module calculates the result and compares it with the known answer. If the calculated value does not match the known answer, the KAT fails and the module enters the error state.

8.2. On-Demand Self-Tests

On-Demand self-tests can be invoked by powering off and reloading the module, which cause the module to run the power-up tests again. During the execution of the on-demand self-tests, services are not available and no data output or input is possible.

8.3. Conditional Tests

The module does not perform conditional tests.

9. Guidance

9.1. Crypto Officer Guidance

The binaries of the module are contained in the Debian packages for delivery. The Crypto Officer shall follow this Security Policy to configure the operational environment and install the module to be operated as a FIPS 140-2 validated module.
The following table lists the 2.2.1 Debian packages containing the FIPS validated module: Ubuntu OpenSSH Server Cryptographic Module version 1.0.

Processor Architecture	Debian packages
x86_64	openssh-server_1:7.2p2-4ubuntu2.fips.2.2.1_amd64.deb openssh-server-hmac_1:7.2p2-4ubuntu2.fips.2.2.1_amd64.deb
Power	openssh-server_1:7.2p2-4ubuntu2.fips.2.2.1_ppc64el.deb openssh-server-hmac_1:7.2p2-4ubuntu2.fips.2.2.1_ppc64el.deb
z System	openssh-server_1:7.2p2-4ubuntu2.fips.2.2.1_s390x.deb openssh-server-hmac_1:7.2p2-4ubuntu2.fips.2.2.1_s390x.deb

Table 11(A) – 2.2.1 Debian packages

The following table lists the 2.8.1 Debian packages containing the FIPS validated module: Ubuntu OpenSSH Server Cryptographic Module version 1.1.

Processor Architecture	Debian packages
x86_64	openssh-server_1:7.2p2-4ubuntu2.fips.2.8.1_amd64.deb openssh-server-hmac_1:7.2p2-4ubuntu2.fips.2.8.1_amd64.deb
Power	openssh-server_1:7.2p2-4ubuntu2.fips.2.8.1_ppc64el.deb openssh-server-hmac_1:7.2p2-4ubuntu2.fips.2.8.1_ppc64el.deb
z System	openssh-server_1:7.2p2-4ubuntu2.fips.2.8.1_s390x.deb openssh-server-hmac_1:7.2p2-4ubuntu2.fips.2.8.1_s390x.deb

Table 11(B) – 2.8.1 Debian packages

The following table lists the 2.10.1 Debian packages containing the FIPS validated module: Ubuntu OpenSSH Server Cryptographic Module version 1.2.

Processor Architecture	Debian packages
x86_64	openssh-server_1:7.2p2-4ubuntu2.fips.2.10.1_amd64.deb openssh-server-hmac_1:7.2p2-4ubuntu2.fips.2.10.1_amd64.deb

Table 11(C) – 2.10.1 Debian Packages

Note: The prelink is not installed on Ubuntu, by default. For proper operation of the in-module integrity verification, the prelink should be disabled.

9.1.1. Operating Environment Configurations

To configure the operating environment to support FIPS, the following shall be performed with the root privilege:

(1) Install the linux-fips Debian package.

(2) Install the fips-initramfs Debian package. (Optional)

(3) For x86_64 and Power systems, create the file /etc/default/grub.d/99-fips.cfg with the content: GRUB_CMDLINE_LINUX_DEFAULT=”$GRUB_CMDLINE_LINUX_DEFAULT fips=1”. For z system, edit /etc/zipl.conf file and append the "fips=1" in the parameters line for the specified boot image.

(4) If /boot resides on a separate partition, the kernel parameter bootdev=UUID=<UUID of partition> must also be appended in the aforementioned grub or zipl.conf file. Please see the following **Note** for more details

(5) Execute update-grub or zipl for z system to update the boot loader.

(6) Reboot the system to apply the settings.

Now, the operating environment is configured to support FIPS operation. The Crypto Officer should check the existence of the file, /proc/sys/crypto/fips_enabled, and that it contains “1”. If the file does not exist or does not contain “1”, the operating environment is not configured to support FIPS and the module will not operate as a FIPS validated module properly.

Note: If /boot resides on a separate partition, the kernel parameter bootdev=UUID=<UUID of partition> must be supplied. The partition can be identified with the command df /boot. For example:

```
$ df /boot
Filesystem     1K-blocks   Used Available Use% Mounted on
/dev/sdb2         241965 127948    101525  56% /boot
```

The UUID of the /boot partition can be found by using the command grep /boot /etc/fstab. For example:

```
$ grep /boot /etc/fstab
# /boot was on /dev/sdb2 during installation
UUID=cec0abe7-14a6-4e72-83ba-b912468bbb38 /boot    ext2    defaults    0    2
```

Then, the following string needs to be appended to the /etc/default/grub file:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet bootdev=UUID=cec0abe7-14a6-4e72-83ba-b912468bbb38 fips=1"
```

9.1.2. Module Installation

Once the operating environment configuration is finished, the Crypto Officer can install the openssh-server and openssh-server-hmac Debian packages listed in Table 11(A), Table 11(B) and Table 11(C) using normal packaging tool such as Advanced Package Tool (APT). All the Debian packages are associated with hashes for

integrity check. The integrity of the Debian package is automatically verified by the packaging tool during the module installation. The Crypto Officer shall not install the Debian package if the integrity of the Debian package fails.
To download the FIPS validated version of the module, please contact the Canonical representative for the repository path.

Once the module is installed successfully, a subsequent manual install/upgrade of the Debian packages (i.e., sudo apt install openssh-server openssh-server-hmac) is prohibited. It could upgrade the FIPS validated module to latest non-FIPS validated OpenSSH application, and this cannot be prevented by "holding" the Debian packages.

Note: During a system update, the installed FIPS validated module could get updated to a later non-FIPS validated OpenSSH application. It is recommended to put a "hold" on the module's Debian packages (i.e., openssh-server and openssh-server-hmac) to exclude the FIPS validated module from automatic system updating/upgrading. The FIPS validated module will remain installed on the system after system update.

To hold the Debian package of the module,

```
$ sudo apt-mark hold openssh-server openssh-server-hmac
```

To unhold the Debian packages of the module,

```
$ sudo apt-mark unhold openssh-server openssh-server-hmac
```

9.2. User Guidance

This module is designed to be used by connecting it with an SSH client. To use a FIPS validated SSH client, please see the Ubuntu OpenSSH Client Cryptographic Module FIPS 140-2 Non-Proprietary Security Policy Version 3.0.

9.2.1. Starting an OpenSSH Server

To start the sshd daemon, use the following command:

```
$ systemctl start sshd
```

To stop the sshd daemon, use the following command:

```
$ systemctl stop sshd
```

To start the sshd daemon automatically at the system boot time, use the following command:

```
$ systemctl enable sshd
```

To prevent the sshd daemon from automatically starting, use the following command:

```
$ systemctl disable sshd
```

To operate the module in FIPS mode, please consider the following restrictions:

- Only the SSHv2 cipher suites listed in Appendix A are available to be used.
- Use of curve25519 keys for signature generation and verification in SSH key-based or certificate-based authentication will result in the module entering non-FIPS mode implicitly because Ed25519 signature generation and verification are non-Approved algorithms.

- Use of 1024-bit DSA keys for signature generation in SSH key-based or certificate-based authentication will result in the module entering non-FIPS mode implicitly. The DSA signature verification with 1024-bit key is only for legacy use.
- Use of less than 2048-bit RSA keys for signature generation or less than 1024-bit RSA keys for signature verification will result in the module entering non-FIPS mode implicitly.
- See the man pages of sshd command for more information about how to operate the module.

9.2.2. Handling Self-Test Errors

When the module fails any self-test, it will return an error message to indicate the error and enters the error state. The following table shows the list of error messages when the module fails any self-test.

Error Events	Error Messages
When the Integrity Test fails at the power-up	"fips: mandatory checksum failed – aborting" "fips: mandatory checksum data missing – aborting"
When the KAT fails at the power-up	"ssh self test failed, aborting"

Table 12 - Self-Tests

To recover from the error state, the module must be restarted and perform power-up tests again. If the failure persists, the module must be reinstalled.

Note: Self-test failures in the bound OpenSSL module will prevent the Ubuntu OpenSSH Server Cryptographic Module from operating. See the Guidance section in the Ubuntu OpenSSL Cryptographic Module FIPS 140-2 Non-Proprietary Security Policy [OPENSSL-SP] for instructions on handling OpenSSL self-test failures.

10. Mitigation of Other Attacks

The module does not implement security mechanisms to mitigate other attacks.

Appendix A. SSHv2 Cipher Suites

In FIPS mode, the module only supports the following cipher suites for the different aspects of the SSHv2 protocol:

Encryption / Decryption	Description	Reference
3des-cbc	Three-key Triple-DES in CBC mode	RFC4253
aes128-cbc	AES in CBC mode with 128-bit key	RFC4253
aes192-cbc	AES in CBC mode with 192-bit key	RFC4253
aes256-cbc/rijndael-cbc@lysator.liu.se	AES in CBC mode with 256-bit key	RFC4253
aes128-ctr	AES in CTR mode with 128-bit key	RFC4344
aes192-ctr	AES in CTR mode with 192-bit key	RFC4344
aes256-ctr	AES in CTR mode with 256-bit key	RFC4344

Table 13 – Symmetric ciphers allowed in FIPS mode

Data Integrity (MAC)	Description	Reference
hmac-sha1/ hmac-sha1-etm@openssh.com	HMAC-SHA-1	RFC4253
hmac-sha2-256/ hmac-sha2-256-etm@openssh.com	HMAC-SHA-256	RFC6668
hmac-sha2-512/ hmac-sha2-512-etm@openssh.com	HMAC-SHA-512	RFC6668

Table 14 - Data Integrity algorithms (MAC) allowed in FIPS mode

Key Exchange	Description	Reference
diffie-hellman-group-exchange-sha256	Diffie-Hellman with SHA-256	RFC4419
ecdh-sha2-nistp256	EC Diffie-Hellman with SHA-256 and NIST P-256 key	RFC5656
ecdh-sha2-nistp384	EC Diffie-Hellman with SHA-384 and NIST P-384 key	RFC5656
ecdh-sha2-nistp521	EC Diffie-Hellman with SHA-512 and NIST P-521 key	RFC5656

Table 15 - Key Exchange Methods allowed in FIPS mode

Appendix B. Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
CAVP	Cryptographic Algorithm Validation Program
CAVS	Cryptographic Algorithm Validation System
CBC	Cipher Block Chaining
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
DRBG	Deterministic Random Bit Generator
EC	Elliptic Curve
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standards Publication
HMAC	Hash Message Authentication Code
KAT	Known Answer Test
KDF	Key Derivation Function
MAC	Message Authentication Code
NDRNG	Non-Deterministic Random Number Generator
NIST	National Institute of Science and Technology
PAA	Processor Algorithm Acceleration
PAI	Processor Algorithm Implementation
RSA	Rivest, Shamir, Adleman
SHA	Secure Hash Algorithm
SSH	Secure Shell

Appendix C. References

FIPS140-2	**FIPS PUB 140-2 - Security Requirements For Cryptographic Modules** May 2001 http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf
FIPS140-2_IG	**Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program** June 17, 2016 http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf
OPENSSL-SP	**Ubuntu OpenSSL Cryptographic Module FIPS 140-2 Non-Proprietary Security Policy** **Certificate# 2888** http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140sp/140sp2888.pdf **Ubuntu OpenSSL Cryptographic Module FIPS 140-2 Non-Proprietary Security Policy** **Certificate# 3725** https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp3725.pdf
SP800-90A	**NIST Special Publication 800-90A - Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP800-131A	**NIST Special Publication 800-131A Revision 2- Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths** November 2015 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf
SP800-135	**NIST Special Publication 800-135 Revision 1 - Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf