Forcepoint NGFW Cryptographic Library

FIPS 140-2 Non-Proprietary Security Policy

Version 2.5

Last Update: 2018-02-23

Prepared by:
atsec information security Corp.
9130 Jollyville Road, Suite 260
Austin, TX 78759
www.atsec.com

1. Introduction

This document is a non-proprietary FIPS 140-2 Security Policy for the Forcepoint NGFW Cryptographic Library module. The current version of the module is 2.0. An earlier version of this module has gone through FIPS 140-2 validation under certificate #2031. This document contains a specification of the rules under which the module must operate and describes how this module meets the requirements as specified in the Federal Information Processing Standards Publication (FIPS PUB) 140-2 for a Security Level 1 multi-chip standalone software module.

1.1. Purpose of the Security Policy

There are three major reasons that a security policy is required:

- For FIPS 140-2 validation,
- Allows individuals and organizations to determine whether the cryptographic module, as implemented, satisfies the stated security policy, and
- Describes the capabilities, protection, and access rights provided by the cryptographic module, allowing individuals and organizations to determine whether it will meet their security requirements.

1.2. Target Audience

This document is intended to be part of the package of documents that are submitted for FIPS validation. It is intended for the following people:

- Developers working on the release
- FIPS 140-2 testing lab
- Cryptographic Module Validation Program (CMVP)
- Consumers

2. Cryptographic Module Specification

This document is the non-proprietary security policy for the Forcepoint NGFW Cryptographic Library and was prepared as part of the requirements to FIPS 140-2, Level 1.

The following section describes the module and how it complies with the FIPS 140-2 standard in each of the required areas.

2.1. Description of Module

The Forcepoint NGFW Cryptographic Library is a shared library that provides a C-language application programming interface for use by Forcepoint applications. Assembly language optimizations are used in the cryptographic module implementation.

The files consisting of the logical boundary of the module are the module binary file libqscrypto.so.2 and the checksums.fips file that contains the HMAC-SHA-256 value needed for the module integrity check. The module contains the following cryptographic functionality:

- Pseudo random number generation
- Cryptographic hash functions
- Message authentication code functions
- Symmetric key encryption and decryption
- Public key cryptography: key pair generation, digital signature generation and verification
- Key agreement and establishment

The following table shows the overview of the security level for each of the eleven sections of the validation.

Security Component	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services and Authentication	1
Finite State Model	1
Physical Security	1
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self Tests	1
Design Assurance	3
Mitigation of Other Attacks	N/A

Table 1: Security Levels

The module has been tested on the following platforms:

Manufacturer	Model	O/S & Ver.	AES-NI
Forcepoint	MIL-320	Debian GNU/Linux 6.0-based distribution running on Intel Atom D (single-user mode)	Not Supported
Forcepoint	5206	Debian GNU/Linux 6.0-based distribution running on Intel Xeon E5 (single-user mode)	With AES-NI
Forcepoint	3206	Debian GNU/Linux 6.0-based distribution running on Intel Xeon E5 (single-user mode)	With and Without AES-NI
Forcepoint	3202	Debian GNU/Linux 6.0-based distribution running on Intel Xeon Processor E5 (single-user mode)	With and Without AES-NI
Forcepoint	1402	Debian GNU/Linux 6.0-based distribution running on Intel Xeon Processor E5 (single-user mode)	With AES-NI
Forcepoint	1065	Debian GNU/Linux 6.0-based distribution running on Intel Core i3 (single-user mode)	With AES-NI
Forcepoint	1035	Debian GNU/Linux 6.0-based distribution running on Intel Celeron (single-user mode)	With AES-NI
Forcepoint	325-C2	Debian GNU/Linux 9.0-based distribution running on Intel Atom C (single-user mode)	With AES-NI
Forcepoint	2105	Debian GNU/Linux 9.0-based distribution running on Intel Xeon D (single-user mode)	With AES-NI
Forcepoint	3305	Debian GNU/Linux 9.0-based distribution running on Intel Xeon E5 (single-user mode)	With and Without AES-NI
Forcepoint	6205	Debian GNU/Linux 9.0-based distribution running on Intel Xeon E5 (single-user mode)	With AES-NI

Table 2A: Tested Platforms

The following are vendor affirmed platforms:

Manufacturer	Model	O/S & Ver.	AES-NI
Forcepoint	321-C2	Debian GNU/Linux 9.0-based distribution running on Intel Atom C (single-user mode)	With AES-NI
Forcepoint	2101	Debian GNU/Linux 9.0-based distribution running on Intel Xeon D (single-user mode)	With AES-NI
Forcepoint	3301	Debian GNU/Linux 9.0-based distribution running on Intel Xeon E5 (single-user mode)	With AES-NI

Table 2B: Vendor Affirmed Platforms

Note: Per IG G.5, the CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when the module is ported to the vendor affirmed platforms that are not listed on the validation certificate.

2.2. Description of Approved Mode

The cryptographic module supports two modes of operation, a FIPS 140-2 Approved mode and a Non-Approved mode. The mode of operation is implicitly assumed. After the power-on self tests (POSTs) are successful, by default the module is placed in the Approved mode. The calling application can invoke ssh_crypto_get_certification_mode() to check the status of the module. It returns SH_CRYPTO_CERTIFICATION_FIPS_140_2 to indicate that the module is in an operational mode. Any calls to the Non-Approved service listed in Table 5, will implicitly put the module into the Non-Approved mode.

The module provides the following algorithms and services:

- AES: encryption and decryption; ECB, CBC, OFB, CFB128 and GCM modes
- Triple-DES: encryption and decryption; ECB, CBC, OFB and CFB64 modes
- DSA: key generation, digital signatures, and verification
- RSA: key generation, digital signatures, and verification
- ECDSA: key generation, digital signature, and verification
- DRBG: random number generation
- SHS: hashing
- HMAC: message authentication code

In addition, the module provides the following key establishment methods:

- Diffie-Hellman key agreement as key establishment method
- EC Diffie-Hellman: key agreement as key establishment method

2.3. Cryptographic Module Boundary

2.3.1. Software Block Diagram

The logical boundary of the module is the Cryptographic Library itself, which is indicated by the "Cryptographic Boundary" rectangle as illustrated in the diagram below.

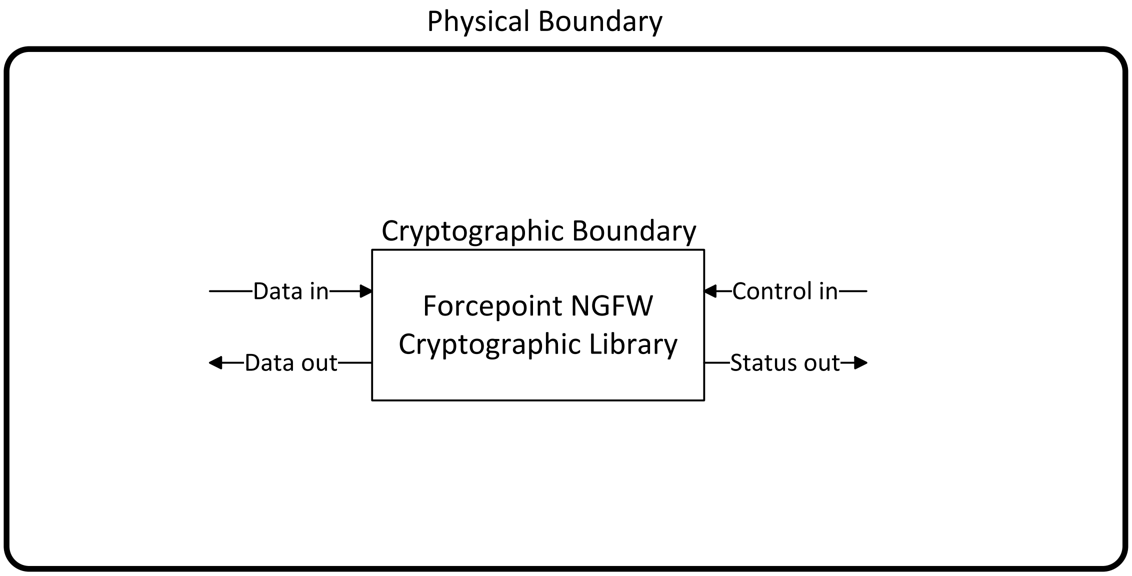
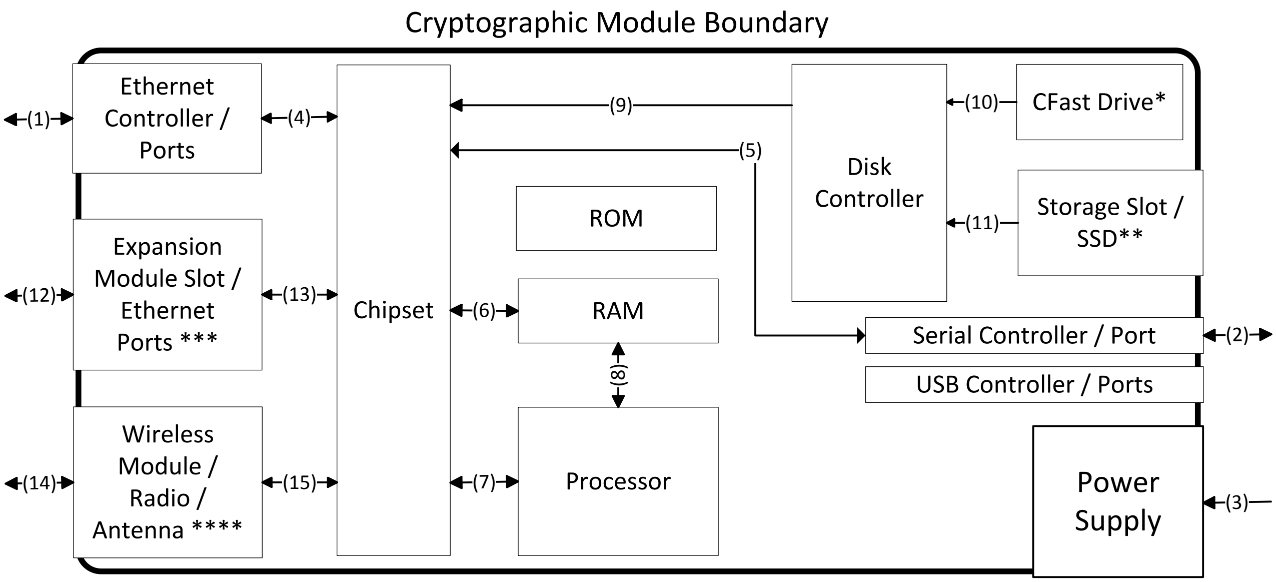


Figure 1: Software Block Diagram

2.3.2. Hardware Block Diagram

The physical boundary of the module is the enclosure of the appliance that the module is running on. The module was tested on seven separate appliances, all of which are general purpose computers. The hardware block diagram below depicts all test appliances and their internal components and ports (processor, SSD, USB, Ethernet, etc.).



1, 2, 4, 5, 6, 7, 8, 12, 13, 14 and 15: Data in, data out, control in, status out
3: Power in
9, 10 and 11: Control in

*) 321, 325, 2101, 2105, MIL-320, 1035, 1065
**) 3301, 3305, 6205, 1402, 3202, 3206, 5206
***) 325, 2101, 2105, 3301, 3305, 6205, 1035, 1065, 1402, 3202, 3206, 5206
****) 325, MIL-320

Figure 2: Hardware Block Diagram

3. Cryptographic Module Ports and Interfaces

FIPS Interface	Physical Ports	Logical Ports
Data Input	Ethernet ports, serial port, wireless radio	API input parameters
Data Output	Ethernet ports, serial port, wireless radio	API output parameters
Control Input	Ethernet ports, serial port, wireless radio	API function calls
Status Output	Ethernet ports, serial port, wireless radio	API return values
Power Input	PC power supply port	N/A

Table 3: Ports and Interfaces

4. Roles, Services, and Authentication

4.1. Roles

The module implements both a User and a Crypto Officer role. The module does not allow concurrent operators.

The User and Crypto Officer roles are implicitly assumed by the entity accessing services implemented by the module. No further authentication is required. The Crypto Officer can install and initialize the module.

4.2. Services

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
Symmetric Algorithms							
AES encryption and decryption	✓		128, 192, 256 bit keys	ECB, CBC, OFB, CFB128,	Yes Certs #2948, #2949, #2950, #2951, #2952, #2953, #2954, #2955, #4591, #4592, #4593, #4716	RWX	FIPS 197 ssh_cipher_allocate ssh_cipher_free ssh_cipher_get_block_length ssh_cipher_get_iv ssh_cipher_get_iv_length ssh_cipher_get_key_length ssh_cipher_get_max_key_length ssh_cipher_get_min_key_length ssh_cipher_get_supported ssh_cipher_has_fixed_key_length ssh_cipher_is_fips_approved ssh_cipher_name sh_cipher_set_iv ssh_cipher_supported ssh_cipher_transform ssh_cipher_transform_remaining ssh_cipher_transfor

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
							m_with_iv ssh_cipher_get_block_len
AES-GCM authenticated encryption and decryption	✓		128, 192, 256 bit keys	GCM	Yes Certs #2948, #2949, #2950, #2951, #2952, #2953, #2954, #2955, #4591, #4592, #4593, #4716	RWX	SP 800-38D ssh_cipher_allocate ssh_cipher_free ssh_cipher_get_block_length ssh_cipher_get_iv ssh_cipher_get_iv_length ssh_cipher_get_key_length ssh_cipher_get_max_key_length ssh_cipher_get_min_key_length ssh_cipher_get_supported ssh_cipher_has_fixed_key_length ssh_cipher_is_fips_approved ssh_cipher_name sh_cipher_set_iv ssh_cipher_supported ssh_cipher_transform ssh_cipher_transform_remaining ssh_cipher_transform_with_iv ssh_cipher_get_block_len ssh_cipher_is_auth_cipher ssh_cipher_auth_reset ssh_cipher_auth_update ssh_cipher_auth_fi

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
							nal ssh_cipher_auth_digest_length ssh_cipher_is_auth ssh_cipher_generate_iv_ctr ssh_cipher_auth_digest_len
Triple-DES encryption and decryption	✓		168 bit keys	ECB, CBC, OFB, CFB64	Yes Certs #1752, #1753, #1754, #1755, #1756, #1757, #2443, #2444, #2445	RWX	SP 800-67 ssh_cipher_allocate ssh_cipher_free ssh_cipher_get_block_length ssh_cipher_get_iv ssh_cipher_get_iv_length ssh_cipher_get_key_length ssh_cipher_get_max_key_length ssh_cipher_get_min_key_length ssh_cipher_get_supported ssh_cipher_has_fixed_key_length ssh_cipher_is_fips_approved ssh_cipher_name sh_cipher_set_iv ssh_cipher_supported ssh_cipher_transform ssh_cipher_transform_remaining ssh_cipher_transform_with_iv ssh_cipher_get_block_len
Asymmetric Algorithms							

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
DSA domain parameter generation	✓		L=2048, N=224; L=2048, N=256; L=3072, N=256		Yes Certs #878, #879, #880, #881, #882, #883, #1217, #1218, #1219	RWX	FIPS 186-4 ssh_private_key_generate
DSA key pair generation	✓		L=2048, N=224; L=2048, N=256; L=3072, N=256		Yes Certs #878, #879, #880, #881, #882, #883, #1217, #1218, #1219	RWX	FIPS 186-4 ssh_private_key_generate ssh_private_key_derive_public_key
DSA signature generation	✓		L=2048, N=224; L=2048, N=256; L=3072, N=256		Yes Certs #878, #879, #880, #881, #882, #883, #1217, #1218, #1219	RX	FIPS 186-4 ssh_private_key_sign ssh_private_key_sign_async ssh_private_key_sign_digest ssh_private_key_sign_digest_async ssh_private_key_max_signature_input_len ssh_private_key_max_signature_output_len ssh_private_key_derive_signature_hash ssh_proxy_key_rgf_sign

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
DSA signature verification	✓		L=1024, N=160; L=2048, N=224; L=2048, N=256; L=3072, N=256		Yes Certs #878, #879, #880, #881, #882, #883, #1217, #1218, #1219	RX	FIPS 186-4 ssh_public_key_verify_async ssh_public_key_verify_digest_async ssh_public_key_verify_signature ssh_public_key_verify_signature_with_digest ssh_public_key_derive_signature_hash ssh_proxy_key_rgf_verify
DSA public key validation	✓		1024, 2048, 3072 bits modulus size		N/A	RX	FIPS 186-4 ssh_public_key_validate
RSA key generation	✓		2048, 3072 modulus size. Public key value 65537.		Yes Certs #1549, #1550, #1551, #1552, #1553, #1554, #2502, #2503, #2504	RWX	FIPS 186-4 ssh_private_key_generate ssh_private_key_derive_public_key ssh_mp_fip186_ifc_aux_prime_create ssh_mp_fips186_ifc_prime_factor sg_mp_fip186_ifc_aux_prime_create
RSA signature generation based on PKCS#1 v1.5	✓		2048, 3072 bit modulus	SHA-224, SHA-256, SHA-384, SHA-512	Yes Certs #1549, #1550, #1551, #1552, #1553, #1554, #2502, #2503, #2504	RX	FIPS 186-4 ssh_private_key_sign ssh_private_key_sign_async ssh_private_key_sign_digest ssh_private_key_sign_digest_async ssh_private_key_max_signature_input_len

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
							ssh_private_key_max_signature_output_len ssh_private_key_derive_signature_hash ssh_proxy_key_rgf_sign
RSA signature verification based on PKCS#1 v1.5	✓		1024, 2048, 3072 bit modulus	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	Yes Certs #1549, #1550, #1551, #1552, #1553, #1554, #2502, #2503, #2504	RX	FIPS 186-4 ssh_public_key_verify_async ssh_public_key_verify_digest_async ssh_public_key_verify_signature ssh_public_key_verify_signature_with_digest ssh_public_key_derive_signature_hash ssh_proxy_key_rgf_verify
RSA signature generation based on PSS (probabilistic signature scheme)	✓		2048, 3072 bit modulus	SHA-224, SHA-256, SHA-384, SHA-512	Yes Certs #1549, #1550, #1551, #1552, #1553, #1554, #2502, #2503, #2504	RX	FIPS 186-4 ssh_private_key_sign ssh_private_key_sign_async ssh_private_key_sign_digest ssh_private_key_sign_digest_async ssh_private_key_max_signature_input_len ssh_private_key_max_signature_output_len ssh_private_key_derive_signature_hash ssh_proxy_key_rgf_sign

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
RSA signature verification based on PSS (probabilistic signature scheme)	✓		1024, 2048, 3072 bit modulus	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	Yes Certs #1549, #1550, #1551, #1552, #1553, #1554, #2502, #2503, #2504	RX	FIPS 186-4 ssh_public_key_verify_async ssh_public_key_verify_digest_async ssh_public_key_verify_signature ssh_public_key_verify_signature_with_digest ssh_public_key_derive_signature_hash ssh_proxy_key_rgf_verify
RSA public key validation	✓		1024, 2048, 3072 bit modulus		N/A	RX	FIPS 186-4 ssh_public_key_validate
ECDSA key pair generation	✓		224, 256, 384, 521 bit prime modulus		Yes Certs #537, #538, #539, #540, #541, #542, #1124, #1125, #1126	RWX	FIPS 186-4 ssh_private_key_generate ssh_private_key_derive_public_key
ECDSA signature generation	✓		224, 256, 384, 521 bit prime modulus		Yes Certs #537, #538, #539, #540, #541, #542, #1124, #1125, #1126	RX	FIPS 186-4 ssh_private_key_sign ssh_private_key_sign_async ssh_private_key_sign_digest ssh_private_key_sign_digest_async ssh_private_key_max_signature_input_len ssh_private_key_max_signature_outp

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
							ut_len ssh_proxy_key_rgf_sign
ECDSA signature verification	✓		192, 224, 256, 384, 521 bit prime modulus		Yes Certs #537, #538, #539, #540, #541, #542, #1124, #1125, #1126	RX	FIPS 186-4 ssh_public_key_verify_async ssh_public_key_verify_digest_async ssh_public_key_verify_signature ssh_public_key_verify_signature_with_digest ssh_public_key_derive_signature_hash ssh_proxy_key_rgf_verify
ECDSA public key validation	✓		192, 224, 256, 384, 521 bit prime modulus		Yes Certs #537, #538, #539, #540, #541, #542, #1124, #1125, #1126	RX	FIPS 186-4 ssh_public_key_validate
Asymmetric key management	✓		Private keys		N/A	RW	ssh_private_key_copy ssh_private_key_free ssh_private_key_get_info ssh_private_key_is_fips_approved ssh_private_key_name ssh_private_key_precompute ssh_private_key_select_scheme

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
							ssh_public_key_copy ssh_public_key_create_proxy ssh_public_key_free ssh_public_key_get_info ssh_public_key_get_predefined_groups ssh_public_key_get_supported ssh_public_key_is_fips_approved ssh_public_key_name ssh_public_key_precompute
Hash Functions							
SHA-1	✓			N/A	Yes Certs #2482, #2483, #2484, #2485, #2486, #2487, #3765, #3766, #3767	RX	FIPS 180-4 ssh_hash_allocate ssh_hash_asn1_oid ssh_hash_asn1_oid_compare ssh_hash_asn1_oid_generate ssh_hash_compare_result ssh_hash_compare_start ssh_hash_digest_length ssh_hash_final ssh_hash_free ssh_hash_get_supported ssh_hash_input_block_size ssh_hash_is_fips_approved

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
							ssh_hash_name ssh_hash_reset ssh_hash_supported ssh_hash_update ssh_hash_of_buffer ssh_sha_transform ssh_sha_permuted_transform
SHA-224 SHA-256 SHA-384 SHA-512	✓			N/A	Yes Certs #2482, #2483, #2484, #2485, #2486, #2487, #3765, #3766, #3767	RX	FIPS 180-4 ssh_hash_allocate ssh_hash_asn1_oid ssh_hash_asn1_oid_compare ssh_hash_asn1_oid_generate ssh_hash_compare_result ssh_hash_compare_start ssh_hash_digest_length ssh_hash_final ssh_hash_free ssh_hash_get_supported ssh_hash_input_block_size ssh_hash_is_fips_approved ssh_hash_name ssh_hash_reset ssh_hash_supported ssh_hash_update ssh_hash_of_buffer
Message Authentication Codes (MACs)							
HMAC-SHA-1	✓		At least	N/A	Yes	RWX	FIPS 198-1

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
HMAC-SHA-224 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512			112 bits HMAC key		Certs #1869, #1870, #1871, #1872, #1873, #1874, #3039, #3040, #3041		ssh_mac_allocate ssh_mac_final ssh_mac_free ssh_mac_get_block_length ssh_mac_get_max_key_length ssh_mac_get_min_key_length ssh_mac_get_supported ssh_mac_is_fips_approved ssh_mac_length ssh_mac_name ssh_mac_reset ssh_mac_supported ssh_mac_update
Random Number Generation							
DRBG	✓		Seed with 256-bit entropy, Entropy input string with 256-bit entropy	AES 256 ECB	Yes Certs #549, #550, #551, #552, #553, #554, #555, #556, #1532, #1533, #1534, #1605	RWX	SP 800-90A ssh_random_add_noise ssh_random_get_byte ssh_random_get_uint32 ssh_random_stir ssh_random_get_supported ssh_random_supported ssh_random_is_fips_approved ssh_random_allocate ssh_random_free ssh_random_name ssh_random_add_entropy ssh_random_add_li

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
							ght_noise ssh_mprz_aux_mod_random ssh_mprz_aux_mod_random_entropy
Key Agreement							
Diffie-Hellman	✓		Diffie-Hellman secret, shared secret		Yes Certs #344, #346, #348, #350, #352, #354, #1260, #1261, #1262	RWX	SP 800-56A ssh_pk_group_copy ssh_pk_group_count_randomizers ssh_pk_group_dh_agree ssh_pk_group_dh_agree_async ssh_pk_group_dh_agree_max_output_length ssh_pk_group_dh_return_randomizer ssh_pk_group_dh_secret_free ssh_pk_group_dh_setup ssh_pk_group_dh_setup_async ssh_pk_group_dh_setup_max_output_length ssh_pk_group_free ssh_pk_group_generate ssh_pk_group_generate_randomizer ssh_pk_group_get_info
EC Diffie-Hellman	✓		EC Diffie-Hellman secret, shared secret		Yes Certs #344, #345, #346, #347, #348,	RWX	SP 800-56A ssh_pk_group_copy ssh_pk_group_count_randomizers ssh_pk_group_dh_agree

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
					#349, #350, #351, #352, #353, #354, #355, #1260, #1261, #1262		ssh_pk_group_dh_agree_async ssh_pk_group_dh_agree_max_output_length ssh_pk_group_dh_return_randomizer ssh_pk_group_dh_secret_free ssh_pk_group_dh_setup ssh_pk_group_dh_setup_async ssh_pk_group_dh_setup_max_output_length ssh_pk_group_free ssh_pk_group_generate ssh_pk_group_generate_randomizer ssh_pk_group_get_info ssh_pk_group_precompute ssh_pk_group_select_scheme ssh_dh_group_create_proxy
Key Entry and Output							

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
DSA key entry	✓		DSA private key and public key		N/A	W	ssh_pk_import ssh_private_key_define ssh_private_key_import ssh_public_key_define ssh_public_key_import
DSA key output	✓		DSA private key and public key		N/A	R	ssh_pk_export ssh_private_key_export
RSA key entry	✓		RSA private key and public key		N/A	W	ssh_pk_import ssh_private_key_define ssh_private_key_import ssh_public_key_define ssh_public_key_import
RSA key output	✓		RSA private key and public key		N/A	R	ssh_pk_export ssh_private_key_export
ECDSA key entry	✓		ECDSA private key and public key		N/A	W	ssh_pk_import ssh_private_key_define ssh_private_key_import ssh_public_key_define ssh_public_key_import
ECDSA key output	✓		ECDSA private key and public key		N/A	R	ssh_pk_export ssh_private_key_export

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
Diffie-Hellman key entry	✓		Diffie-Hellman private key and public key		N/A	W	ssh_pk_import ssh_pk_group_import ssh_pk_group_import_randomizers
Diffie-Hellman key output	✓		Diffie-Hellman private key and public key		N/A	R	ssh_pk_export ssh_pk_group_export ssh_pk_group_export_randomizers
EC Diffie-Hellman key entry	✓		EC Diffie-Hellman private key and public key		N/A	W	ssh_pk_import ssh_pk_group_import ssh_pk_group_import_randomizers
EC Diffie-Hellman key output	✓		EC Diffie-Hellman private key and public key		N/A	R	ssh_pk_export ssh_pk_group_export ssh_pk_group_export_randomizers
Management							
Installation		✓	N/A	N/A	N/A	N/A	Please refer to section 11.3 “Cryptographic Officer Guidance” for secure installation of the module.

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
Initialization		✓	N/A	N/A	N/A	RX	ssh_crypto_library_initialize ssh_crypto_library_register_noise_request ssh_crypto_library_register_progress_func ssh_pk_provider_register sg_crypto_register_error_callback ssh_random_noise_polling_init ssh_drbg_instantiate sg_drbg_enable_continuous_test ssh_drbg_reseed ssh_drbg_generate ssh_drbg_uninstantiate
Mode management		✓	N/A	N/A	N/A	RX	ssh_crypto_get_certification_mode ssh_crypto_set_certification_mode
Uninitialization		✓	N/A	N/A	N/A	RX	ssh_crypto_free ssh_crypto_library_uninitialize ssh_crypto_library_unregister_noise_request ssh_random_noise_polling_uninit
External crypto registration		✓	N/A	N/A	N/A	RX	The external crypto registration is not supported on the tested Forcepoint platforms. The functions below return SG_CRYPTO_REGISTER_NOT_SUPPORT

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
							ED. sg_cipher_external_register sg_cipher_external_unregister sg_hash_external_register sg_hash_external_unregister sg_mac_external_register sg_mac_external_unregister sg_ciphermac_external_register sg_ciphermac_external_unregister
Status							
Query status	✓	✓	N/A	N/A	N/A	RX	ssh_crypto_library_get_status ssh_crypto_library_get_version ssh_crypto_status_message
Self-tests							
Perform self-tests	✓	✓	N/A	N/A	N/A	RX	ssh_crypto_library_self_tests
Other services							
Compression	✓		N/A	N/A	N/A	RX	ssh_compress_allocate ssh_compress_free ssh_compress_get_supported ssh_compress_is_none ssh_compress_sync_levels ssh_compress_buffer

Service	Roles		CSP	Modes	FIPS Approved? Cert # (if applicable)	Access	Notes/API Function
	User	CO					
Auxiliary services	✓		N/A	N/A	N/A	RX	ssh_aux_pkcs1_pad ssh_aux_pkcs1_unpad ssh_aux_pkcs1_wrap_and_pad ssh_cipher_alias_get_native ssh_cipher_alias_get_supported ssh_cipher_alias_supported ssh_ecp_set_param

Table 4: FIPS-Approved Services

Note – The 3305 and 6205 platforms share the same processor family and operating system. The CAVS certificates for 3305 will therefore apply to both 3305 and 6205 platforms.

Use of this Non-Approved service listed below will cause the Module to operate in the Non-Approved mode implicitly.

Service	Roles	Modes	Access	CSP	Notes	API Function
AES Key Wrapping	User	KW, KWP	RXW	AES key	CAVS tested on 4 platforms: 325-C2,2105, 3305,6205 #4591, #4592, #4593, #4716	sg_aes_key_unwrap_kek_with_padding sg_aes_key_unwrap_with_padding sg_aes_key_wrap_kek_with_padding sg_aes_key_wrap_with_padding ssh_aes_key_unwrap ssh_aes_key_unwrap_kek ssh_aes_key_wrap ssh_aes_key_wrap_kek

Table 5: Non-Approved Services

4.3. Operator Authentication

There is no operator authentication; assumption of role is implicit by action.

4.4. Mechanism and Strength of Authentication

No authentication is required at Security Level 1; authentication is implicit by assumption of the role.

5. Finite State Machine

The following diagram represents the states and transitions of the cryptographic module.

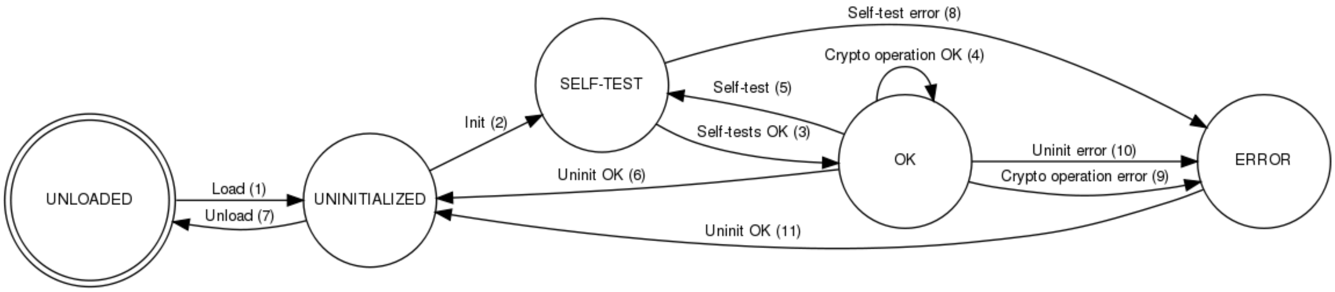


Figure 3: Cryptographic Module Finite State Machine

The state model contains the following states:

- UNLOADED: The start state of the cryptographic module is UNLOADED. The module is in this state until the shared library is loaded and linked to the application. Cryptographic operations are not available while in this state.
- UNINITIALIZED: The module is in the UNINITIALIZED state after it has been loaded but not yet initialized, or it has been successfully uninitialized. Cryptographic operations are not available while in this state.
- SELF-TEST: The module performs power-up self-tests during initialization or on-demand. Cryptographic operations are not available while in this state.
- OK: The module enters the operational mode in the “OK” state after successfully passing the power-up self-tests. The cryptographic services are available in this state.
- ERROR: The module enters this state after a self-test, a cryptographic operation or uninitialization has failed. An error indicator is output by the module.

The state transitions are as follows:

1. The shared library is loaded and linked dynamically to the application.
2. The cryptographic module is initialized using the ssh_crypto_library_initialize function. The function is called automatically when the shared library is loaded.
3. The self-tests succeed.
4. A cryptographic operation is performed successfully.
5. On-demand self-tests are performed using the ssh_crypto_library_self_tests function.
6. The cryptographic module is uninitialized using the ssh_crypto_library_uninitialize function.
7. The shared library is unloaded.
8. Power-up self-tests fail.
9. A conditional test fails during a cryptographic operation.
10. The module uninitialization fails because cryptographic objects are still referenced.
11. Cryptographic objects are no longer in use and the module uninitialization succeeds. This transition also occurs automatically when the power-up self-tests fail during the module initialization.

6. Physical Security

The cryptographic module is tested on the Forcepoint MIL-320, 5206, 3206, 3202, 1402, 1065 and 1035 appliances that consist of production-grade components with standard passivation and a production-grade enclosure.

7. Operational Environment

This module will operate in a modifiable operational environment per the FIPS 140-2 definition. The module operates on the Forcepoint NGFW Debian GNU/Linux based hardened operating system that is set in the FIPS compatible mode of operation. Login to the operating system is disabled and only the preinstalled Forcepoint application is running on the system. Therefore the operational environment is considered non-modifiable. The application that uses the cryptographic module is also the single user of the module.

8. Cryptographic Key Management

Keys are established externally. CSPs can be accessed only using the API. The operating system protects the memory and the address space of the process from unauthorized access.

Name	Auth Role	Generation	Type	Output	Storage	Zeroization
HMAC key for module integrity check	User, CO	Manufacturer	128 bits HMAC key	N/A	In module binary	Zeroization is not required per FIPS IG 7.4
AES symmetric keys	User	External, electronic entry	Symmetric key	N/A	Plaintext in memory	API call, power off
Triple-DES symmetric keys	User	External, electronic entry	Symmetric key	N/A	Plaintext in memory	API call, power off
DSA private key	User	DSA key generation using DRBG, externally using DSA key entry	Private key	Encrypted, plaintext	Plaintext in memory	API call, power off
RSA private key	User	RSA key generation using DRBG, externally using RSA key entry	Private key	Encrypted, plaintext	Plaintext in memory	API call, power off
ECDSA private key	User	ECDSA key generation using DRBG, externally using ECDSA key entry	Private key	Encrypted, plaintext	Plaintext in memory	API call, power off
HMAC key	User	External, electronic entry	HMAC key	N/A	Plaintext in memory	API call, power off
DRBG entropy input	User	External, electronic entry	Entropy input	N/A	Plaintext in memory	API call, power off
DRBG seed	User	/dev/random	Seed	N/A	Plaintext in memory	API call, power off
Diffie-Hellman secret	User	DSA key generation using DRBG	Private key	N/A	Plaintext in memory	API call, power off
Diffie-Hellman	User	Generated through Diffie-	Symmetric	Plaintext	Plaintext in	API call, power off

Name	Auth Role	Generation	Type	Output	Storage	Zeroization
shared secret		Hellman protocol	key		memory	
EC Diffie-Hellman secret	User	ECDSA key generation using DRBG,	Private key	N/A	Plaintext in memory	API call, power off
EC Diffie-Hellman shared secret	User	Generated through Diffie-Hellman protocol	Symmetric key	Plaintext	Plaintext in memory	API call, power off

Table 6: Key Management

8.1. Random Number Generation

The cryptographic module implements an AES block cipher based DRBG with derivation function according to SP 800-90A. The module obtains the seed and the entropy input string by default from /dev/random. The entropy source can be changed by setting the new source either in the /etc/qscrypto.entropysource file or in the LIBQSCRYPTO_ENTROPY_SOURCE environment variable. The seed and the entropy input string are both 256 bytes long. Their security strength is 256 bits, i.e., 1 bit per byte is assumed. In the operational environment, /dev/random is used as the entropy source. The Linux kernel has been patched to contain the CPU Jitter Random Number Generator [19].

8.2. Key/CSP Generation

DSA key pairs are generated using random bits from DRBG according to FIPS 186-4 Appendix B.1.1.

RSA key pairs are generated using probable primes with conditions using auxiliary probable primes and random bits from the DRBG according to FIPS 186-4 Appendix B.3.6.

ECDSA key pairs are generated using extra random bits from the DRBG according to FIPS 186-4 Appendix B.4.1.

Diffie-Hellman and EC Diffie-Hellman secrets and public values are generated using random bits from the DRBG.

8.3. Key/CSP Establishment

The cryptographic module supports Diffie-Hellman primitives for key agreement using ephemeral keys:

- FFC DH dhEphem, C(2, 0, FFC DH) using 2048-bit group
- ECC CDH Ephemeral Unified Model, C(2, 0, ECC CDH) using p-224, p-256, p-384, and p-521 curves

CAVEAT 1: Diffie-Hellman key agreement; key establishment methodology provides 112 bits of encryption strength;

CAVEAT 2: EC Diffie-Hellman key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength.

8.4. Key Entry and Output

The cryptographic module supports electronic entry of symmetric keys and HMAC keys. The application using the cryptographic module can import secret keys to the module in plaintext within the physical boundary.

Private keys can be exported in plaintext to the application using the module within the physical boundary.

There is no output of intermediate key generation values from the module at any point in time. The module does not support manual entry of keys.

8.5. Key Storage

The keys and CSPs are stored in plaintext in memory. The module does not provide persistent storage of keys.

8.6. Zeroization Procedure

The stored keys and CSPs are zeroized when the application calls the appropriate API function: ssh_cipher_free, ssh_mac_free, ssh_private_key_free, ssh_pk_group_free and ssh_crypto_library_uninitialize. Intermediate key material is zeroized automatically by the module when no longer needed. All keys and CSPs can be zeroized by powering off the module and performing a system restore operation by the operational environment.

9. Self-Tests

9.1. Power-Up Tests

The power-up self-tests are executed automatically when the cryptographic module is loaded. The ssh_crypto_library_initialize() function returns 0 (SSH_CRYPTO_OK) when the power-up self-tests are successfully completed.

If the power-up self-tests fail, the cryptographic module outputs an error message and enters an error state. No further operations are allowed when the module is in an error state. The cryptographic module causes the process termination with a non-zero exit status when the power-up self-tests have failed. The computer will need to be restarted in order for the cryptographic module to enter to an operational state.

Self-tests are performed on-demand when the user calls the ssh_crypto_library_self_tests() function.

Algorithm	**Test**
AES	Known Answer Test (KAT), encryption and decryption are tested separately
Triple-DES	KAT, encryption and decryption are tested separately
DSA	Pair-wise consistency test (PCT) for DSA key pair generation
RSA	KAT for signature generation and verification tested separately, PCT for RSA key pair generation
ECDSA	KAT for signature generation, PCT for ECDSA key pair generation
SHS	KAT for SHA-1, SHA-256 and SHA-512
HMAC	KAT for HMAC-SHA-1, HMAC-SHA-256 and HMAC-SHA-512
DRBG	KAT
Diffie-Hellman	KAT, PCT
EC Diffie-Hellman	KAT, PCT

Table 7: Power-Up Tests

The following are the error messages related to self-test failure:

Reason For Failure	**Error Message**
Failure of AES/Triple-DES KAT	Cipher algorithm test failed during self-test
Failure of RSA/DSA/Diffie-Hellman KAT or PCT	Public key algorithm test failed during self-test
Failure of ECDSA/EC-Diffie-Hellman KAT or PCT	Unknown error code (exit code 160)
Failure of SHS KAT	Hash algorithm test failed during self-test
Failure of HMAC KAT	Mac algorithm test failed during self-test
Failure of integrity test	The checksum of the library is incorrect. Integrity has been compromised

Table 8: Error Messages Related to Self-Test Failure

It is the applications responsibility to reboot the appliance to recover the module from the error state. The library will not cause the rebooting of the appliance.

9.2. Integrity Check

The cryptographic module uses the HMAC-SHA-256 message authentication code of the module binary for the integrity tests. The module reads the module binary file, computes the HMAC-SHA-256 MAC of the file content and compares it to the known correct MAC that is input to the module when it is loaded.

9.3. Conditional Tests

Algorithm	Test
DSA	Pair-wise consistency test
RSA	Pair-wise consistency test
ECDSA	Pair-wise consistency test
DRBG	Continuous test

Table 9: Conditional Tests

The following are the error messages related to conditional test failure:

Reason For Failure	Error Message
Failure of DSA pair-wise consistency test	One of the following (%d is error code): Private key consistency test failed: %d Public key consistency test failed: %d DH group consistency test failed: %d and Cryptographic Library error occurred (1)
Failure of RSA pair-wise consistency test	One of the following (%d is error code): Private key consistency test failed: %d Public key consistency test failed: %d and Cryptographic Library error occurred (1)
Failure of ECDSA pair-wise consistency test	One of the following (%d is error code): Private key consistency test failed: %d Public key consistency test failed: %d DH group consistency test failed: %d and Cryptographic Library error occurred (1)
Failure of DRBG continuous test	Continuous DRBG test failed Cryptographic Library error occurred (0)

Table 10: Error Messages Related to Conditional Test Failure

10. Design Assurance

10.1. Configuration Management

Git and SharePoint are used for configuration management of the cryptographic module.

10.2. Delivery and Operation

The cryptographic module is never released as source code. It is delivered as part of the Forcepoint NGFW software (formerly Stonesoft Security Engine). The FIPS 140-2-compatible Forcepoint NGFW software image is downloaded from the Forcepoint website. The Forcepoint NGFW software is also preinstalled on Forcepoint NGFW appliances (see Table 2: Tested Platforms). Product information for the appliances is available at the Forcepoint website: https://www.forcepoint.com/product/network-security/forcepoint-ngfw

10.2.1. Downloading a FIPS 140-2-compatible engine version

A FIPS 140-2-compatible version of the Forcepoint NGFW software is downloaded as follows:

1. Go to the Forcepoint NGFW Downloads page at https://support.forcepoint.com/Downloads.
2. Enter the Proof-of-License (POL) or Proof-of-Serial (POS) code in the License Identification field and click **Submit**.
3. Click **Forcepoint NGFW downloads**. The Forcepoint NGFW Downloads page opens.
4. Download the .zip installation file.
5. Verify the SHA checksum. The correct checksum is shown on the download page.

10.3. Cryptographic Officer Guidance

10.3.1. Installation

The cryptographic module is delivered as part of the Forcepoint NGFW software. To run the cryptographic module on a Forcepoint NGFW appliance, the NGFW software is set to a FIPS 140-2-compatible operating mode.

10.3.1.1 Upgrading appliances to the FIPS 140-2-compatible engine version

Forcepoint NGFW appliances are delivered with the most recent engine software preinstalled. The engine software must be upgraded to the FIPS 140-2-compatible engine version before entering FIPS-compatible operating mode. This is necessary even if the same version was installed previously, because the file system checksum is stored during the upgrade process.

To upgrade to the FIPS-compatible engine version:

1. Save the FIPS 140-2-compatible engine upgrade zip file in the root directory of a USB memory stick. Note – The engine upgrade zip file must be in the root directory of the media.
2. Boot up the appliance. The Engine Configuration Wizard starts.
3. In the NGFW Initial Configuration Wizard, select Firewall/VPN for the role.
4. Select **Upgrade**. The Select Source Media dialog opens.
5. Select **USB Memory**. The upgrade starts.

6. Select **OK**. The engine reboots and the Engine Configuration Wizard starts with the engine image verification dialog shown. Select **Calculate**. The file system checksum is calculated and displayed below the checksum from the engine image zip file.
7. Verify that the calculated checksum is identical to the checksum from the zip file.
8. Select **OK**. The upgrade starts and the engine reboots.
9. Check the engine version to make sure that the certified version is loaded.
10. Select kernel in FIPS mode after reboot.

Continue as instructed in **Configuring the engine**, below.

10.3.1.2 Configuring the engine

To configure the engine:

1. Start the Engine Configuration Wizard as instructed in the **Configuring the Engine in the Engine Configuration Wizard** section of the *Forcepoint NGFW Installation Guide*.
2. Configure the Operating System settings as instructed in the **Configuring the Operating System Settings** section of the *Forcepoint NGFW Installation Guide*. Select **Restricted FIPS-compatible operating mode**. The SSH daemon and root password options are automatically disabled in the Engine Configuration Wizard.
3. Configure the network interfaces according to your environment as instructed in the **Configuring the Network Interfaces** section of the *Forcepoint NGFW Installation Guide*.
4. Contact the Management Server as instructed in the **Contacting the Management Server** section of the *Forcepoint NGFW Installation Guide*. Enter node IP address manually is selected by default and other IP address options are disabled when FIPS-compatible operating mode is enabled.

The engine restarts.

10.3.1.3 Verifying activation of FIPS 140-2-compatible operating mode

Restricted FIPS-compatible operating mode must be enabled during the initial configuration of the appliance. The following steps describe how to verify that FIPS 140-2-compatible operating mode has been activated.

To verify activation of FIPS 140-2-compatible operating mode:

1. Verify that the following messages are displayed on the console when the engine restarts:

```
FIPS: rootfs integrity check OK
```

(displayed after the root file system integrity test has been executed successfully)

```
FIPS power-up tests succeeded
```

(displayed after the FIPS 140-2 power-up tests have been executed successfully)

2. Continue as instructed in the **After Successful Management Server Contact** section of the *Forcepoint NGFW Installation Guide*.

Note – If the engine does not enter the FIPS 140-2-compatible operating mode even though it is configured to do so, or if the power-up tests fail (a power-up test error message is displayed or the success message is not displayed), the appliance must be reset to factory settings and reinstalled as instructed in **Recovering from a FIPS 140-2 self-test failure.**

10.3.1.4 Resetting the appliance to factory settings

Resetting the appliance to factory settings is not part of the normal installation procedure. There is no need to reset the appliance to factory settings before starting to use it for the first time. These instructions can be used to reset the appliance to factory settings when necessary, such as when initial configuration has been completed without enabling the Restricted FIPS 140-2-compatible operating mode, during use, or when the appliance is being removed from use.

To reset the appliance to factory settings:

1. Reboot the appliance and select **System restore options** from the boot menu. Forcepoint NGFW System Restore starts.
2. Enter 2 for **Advanced data removal options**.
3. Enter one of the following options:
   - 1 for **1 pass overwrite**
   - 8 for a **Custom** number of overwrite passes

If you selected **Custom**, enter the number of overwrite passes. A larger number of overwrites is more secure, but it may take a considerable amount of time depending on the appliance storage capacity.

10.3.1.5 Recovering from a FIPS 140-2 self-test failure

If the FIPS 140-2 power-up self-tests fail, or the engine does not enter FIPS 140-2-compatible operating mode, the appliance must be reset to factory settings and reinstalled according to these instructions. Begin by **Resetting the appliance to factory settings**.

To recover from a FIPS 140-2 self-test failure:

1. Reset the appliance to factory settings as instructed in **Resetting the appliance to factory settings**.
2. Repeat the engine version upgrade as instructed in **Upgrading appliances to the FIPS 140-2-compatible engine version**.
3. Configure the firewall engine and enable FIPS 140-2-compatible operating mode as instructed in **Configuring the engine**.
4. Verify that FIPS-compatible operating mode is activated as instructed in **Verifying activation of FIPS 140-2-compatible operating mode**.

10.3.2. Entropy Source

The cryptographic module uses `/dev/random` as the default entropy source. The entropy source can be changed by setting the new source either in the /etc/qscrypto.entropysource file or in the LIBQSCRYPTO_ENTROPY_SOURCE environment variable.

`/dev/random` is always used as the entropy source for the cryptographic module when the Forcepoint NGFW software is in FIPS-compatible operating mode.

10.3.3. Initialization

The cryptographic module is initialized using the ssh_crypto_library_initialize() function before any cryptographic functionality is available. In order for the integrity check to succeed, the known HMAC-SHA-256 MAC needs to be available either in:

`/etc/checksums.fips` file

or

`LIBQSCRYPTO_CHECKSUM` environment variable

The `/etc/checksums.fips` file is provided with the Forcepoint NGFW software.

10.4. User Guidance

10.4.1. AES GCM

In case the module's power is lost and then restored, the key used for the AES GCM encryption/decryption shall be re-distributed.

10.4.2. Zeroization

When a cryptographic key is no longer used, the key must be zeroized and freed using the ssh_cipher_free, ssh_mac_free and ssh_private_key_free functions for symmetric key encryption/decryption, message authentication and public key cryptography, respectively.

10.4.3. Key Export

Private keys must not be exported unencrypted outside the physical module boundary from the application using the cryptographic module.

11. Mitigation of Other Attacks

No other attacks are mitigated.

12. Glossary and Abbreviations

AES	Advanced Encryption Specification
API	Application Programming Interface
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CFB	Cipher Feedback
CMT	Cryptographic Module Testing
CMVP	Cryptographic Module Validation Program
CO	Cryptographic Officer
CSP	Critical Security Parameter
CTR	Counter
CVT	Component Verification Testing
DES	Data Encryption Standard
DH	Diffie-Hellman
DSA	Digital Signature Algorithm
ECB	Electronic Codebook
ECDH	EC Diffie-Hellman
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standards
FSM	Finite State Model
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
KAT	Known Answer Test
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
NVLAP	National Voluntary Laboratory Accreditation Program
OFB	Output Feedback
O/S	Operating System
PCT	Pair-wise Consistency Test

RNG Random Number Generator

RSA Rivest, Shamir, Addleman

SHA Secure Hash Algorithm

SHS Secure Hash Standard

UI User Interface

13. References

[1] FIPS 140-2 Standard, http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[2] FIPS 140-2 Implementation Guidance, http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf

[3] FIPS 140-2 Derived Test Requirements, http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402DTR.pdf

[4] FIPS 197, Advanced Encryption Standard (AES), http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[5] FIPS 180-4 Secure Hash Standard, http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

[6] FIPS 198-1 The Keyed-Hash Message Authentication Code (HMAC), http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

[7] FIPS 186-2, Digital Signature Standard, http://csrc.nist.gov/publications/fips/archive/fips186-2/fips186-2.pdf

[8] FIPS 186-4 Digital Signature Standard (DSS), http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

[9] ANS X9.31 Appendix A.2.4, Random Number Generator, http://csrc.nist.gov/groups/STM/cavp/documents/rng/931rngext.pdf

[10] NIST SP 800-67 Revision 1, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf

[11] NIST SP 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

[12] NIST SP 800-38C, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf

[13] NIST SP 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf

[14] NIST SP 800-38E, Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices, http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf

[15] NIST SP 800-56A, Recommendation for Pair-Wise Key Establishment Schemes using Discrete Logarithm Cryptography (Revised), http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf

[16] NIST SP 800-56B, Recommendation for Pair-Wise Establishment Schemes Using Integer Factorization Cryptography, http://csrc.nist.gov/publications/nistpubs/800-56B/sp800-56B.pdf

[17] NIST SP 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf

[18] NIST SP 800-131A Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf

[19] CPU Time Jitter Based Non-Physical True Random Number Generator, http://www.chronox.de/jent/doc/CPU-Jitter-NPTRNG.pdf