Red Hat Enterprise Linux OpenSSL Cryptographic Module v5.0

and

Red Hat Enterprise Linux OpenSSL Cryptographic Module v6.0

FIPS 140-2 Non-proprietary Security Policy

version 1.2

Last Update: 2018-02-19

Table of Contents

1. Cryptographic Modules' Specifications

This document is the non-proprietary Security Policy for both the Red Hat Enterprise Linux OpenSSL Cryptographic Module v5.0 and the Red Hat Enterprise Linux OpenSSL Cryptographic Module v6.0and was prepared as part of the requirements for conformance to Federal Information Processing Standard (FIPS) 140-2, Level 1.

1.1. Description of the Modules

The Red Hat Enterprise Linux OpenSSL Cryptographic Module v5.0 and the Red Hat Enterprise Linux OpenSSL Cryptographic Module v6.0 (hereafter referred to as the “Modules”) are software libraries supporting FIPS 140-2 Approved cryptographic algorithms. The code base of the Modules is formed in a combination of standard OpenSSL shared library, OpenSSL FIPS Object Module and development work by Red Hat. The Modules provide a C language application program interface (API) for use by other processes that require cryptographic functionality.

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

Security Component	FIPS 140-2 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	1

Table 1: Security Level of the Modules

The Red Hat Enterprise Linux OpenSSL Cryptographic Module v5.0has been tested on the following multi-chip standalone platforms:

Manufacturer	Model	O/S & Ver.	Processor
Dell	PowerEdge R630	Red Hat Enterprise Linux 7.4	Intel(R) Xeon(R) CPU E5-2640 v3

Table 2: Test Platform

The Red Hat Enterprise Linux OpenSSL Cryptographic Module v6.0has been tested on the following multi-chip standalone platforms:

Manufacturer	Model	O/S & Ver.	Processor
Dell	PowerEdge R630	Red Hat Enterprise Linux 7.5	Intel(R) Xeon(R) CPU E5-2640 v3

Table 3: Test Platform

The Module has been tested for the following configurations:
- 32-bit library, x86_64 with and without AES-NI enabled
- 64-bit library, x86_64 with and without AES-NI enabled
-

To operate the Modules, the operating system must be restricted to a single operator mode of operation. (This should not be confused with single user mode which is run level 1 on Red Hat Enterprise Linux (RHEL). This refers to processes having access to the same cryptographic instance which RHEL ensures this cannot happen by the memory management hardware.)

1.2. Description of the Approved Modes

The Modules support two modes of operation:
- in "FIPS mode" (the FIPS Approved mode of operation) only approved or allowed security functions with sufficient security strength can be used.
- in "non-FIPS mode" (the non-Approved mode of operation) non-approved security functions can also be used.

The Modules verify the integrity of the runtime executable using a HMAC-SHA-256 digest computed at build time. If the digests matched, the power-up self-test is then performed. The modules enter FIPS mode after power-up tests succeed. Once the modules iare operational, the mode of operation is implicitly assumed depending on the security function invoked and the security strength of the cryptographic keys.

The Red Hat Enterprise Linux OpenSSL Cryptographic Module v5.0 supports the following FIPS 140-2 Approved algorithms in FIPS Approved mode:

Algorithm	Validation Certificate	Standards/Usage	Keys/CSPs
AES	Certs. #4644, #4664, #4666, #4667, #4695, #4696, #4697, #4698, #4699, and #4700	FIPS 197 (AES) SP 800-38A SP 800-38C (CCM) SP 800-38D (GCM) SP 800-38E (XTS) SP 800-38F (Key Wrapping) Encryption and Decryption	AES keys 128 bits, 192 bits (except XTS-AES) and 256 bits
Triple-DES	Certs. #2471, #2481, #2483, and #2484	SP 800-67 Encryption and Decryption	Triple-DES keys 168 bits
DSA	Certs. #1228, #1235, #1237, and #1238,	FIPS 186-4	DSA keys: • L=2048, N=224

Algorithm	Validation Certificate	Standards/Usage	Keys/CSPs
		Domain Parameters Generation and Verification, Key Generation, Signature Generation	• L=2048, N=256 • L=3072, N=256 Note: 1024 bit DSA signature verification is legacy-use.
RSA Key Generation	Certs. #2535, #2544, #2546, and #2547	FIPS 186-4 Appendix B.3.3 Key Generation	RSA keys: • 1024 bits • 2048 bits • 3072 bits Note: 1024 bit RSA signature verification is legacy-use.
		Signature Generation and Verification (ANSI X9.31 and PKCS#1 v1.5)	
ECDSA	Certs. #1144, #1148, #1150, and #1151	FIPS 186-4 Key Pair Generation and Public Key Verification	ECDSA keys based on P-256, P-384, or P-521 curve
		FIPS 186-4 Signature Generation	
		FIPS 186-4 Signature Verification	
DRBG	Certs. #1567, #1576, #1578, #1579, #1593, #1594, #1595, #1596, #1597, and #1598	SP 800-90A Random Number Generation	Entropy input string, seed, V and Key
SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	Certs. #3807, #3821, #3823, #3824, #3842, #3843, #3844, #3845, #3846, and #3847	FIPS 180-4 Hashing	N/A
HMAC-SHA-1 HMAC-SHA-224 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512	Certs. #3076, #3088, #3090, #3091, #3107, #3108, #3109, #3110, #3111 and #3112	FIPS 198-1 Message Integrity	At least 112 bits HMAC Key
SP 800-56A DLC	Certs. #1298, #1312,	SP 800-56A	Public key size 2048

Algorithm	Validation Certificate	Standards/Usage	Keys/CSPs
primitive Diffie-Hellman	#1318 and #1320	Key Agreement and Establishment	bits or larger, and private key size 224 bits or 256 bits
SP 800-56A DLC primitive EC Diffie-Hellman			NIST curves P-256, P-384, P-521
SP 800-135 Section 4.2 Key Derivation in TLS v1.0, v1.1 and v1.2	Certs. #1299, #1313, #1319, #1321, #1338 and #1339	SP800-135 Key Derivation in TLS	None

Table 4: Approved Algorithms

The Red Hat Enterprise Linux OpenSSL Cryptographic Module v6.0supports the following FIPS 140-2 Approved algorithms in FIPS Approved mode:

Algorithm	Validation Certificate	Standards/Usage	Keys/CSPs
AES	Certs. #5203, #5204, #5205, #5207, #5208, #5209, #5210, #5211, #5212, and #5227	FIPS 197 (AES) SP 800-38A SP 800-38C (CCM) SP 800-38D (GCM) SP 800-38E (XTS) SP 800-38F (Key Wrapping) Encryption and Decryption	AES keys 128 bits, 192 bits (except XTS-AES) and 256 bits
Triple-DES	Certs. #2638, #2639, #2641,and #2642	SP 800-67 Encryption and Decryption	Triple-DES keys 168 bits
DSA	Certs. # #1346, #1347, #1349 and 1350	FIPS 186-4 Domain Parameters Generation and Verification, Key Generation, Signature Generation	DSA keys: • L=2048, N=224 • L=2048, N=256 • L=3072, N=256 Note: 1024 bit DSA signature verification is legacy-use.
RSA Key Generation	Certs. #2786, #2787, #2789 and #2792	FIPS 186-4 Appendix B.3.3 Key Generation	RSA keys: • 1024 bits • 2048 bits • 3072 bits Note: 1024 bit RSA signature verification is
		Signature Generation and Verification (ANSI	

Algorithm	Validation Certificate	Standards/Usage	Keys/CSPs
		X9.31 and PKCS#1 v1.5)	legacy-use.
ECDSA	Certs. ##1347, #1348, #1350 and #1353	FIPS 186-4 Key Pair Generation and Public Key Verification	ECDSA keys based on P-256, P-384, or P-521 curve
		FIPS 186-4 Signature Generation	
		FIPS 186-4 Signature Verification	
DRBG	Certs. #1975, #1976, #1977, #1979, #1980, #1981, #1982, #1983, #1984 and #1993	SP 800-90A Random Number Generation	Entropy input string, seed, V and Key
SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	Certs. #4193, #4194, #4195, #4197, #4198, #4199, #4200, #4201, #4202 and #4207	FIPS 180-4 Hashing	N/A
HMAC-SHA-1 HMAC-SHA-224 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512	Certs. #3445, #3446, #3447, #3449, #3450, #3451, #3452, #3453, #3454 and #3459	FIPS 198-1 Message Integrity	At least 112 bits HMAC Key
SP 800-56A DLC primitive Diffie-Hellman	Certs. #1687, #1689, #1693 and #1700	SP 800-56A Key Agreement and Establishment	Public key size 2048 bits or larger, and private key size 224 bits or 256 bits
SP 800-56A DLC primitive EC Diffie-Hellman			NIST curves P-256, P-384, P-521
SP 800-135 Section 4.2 Key Derivation in TLS v1.0, v1.1 and v1.2	Certs. #1688, #1690, #1691, #1692, #1694 and #1701	SP800-135 Key Derivation in TLS	None

Table 5: Approved Algorithms

The Modules support the following non-Approved algorithms but allowed in FIPS Approved mode:

Algorithm	Usage	Keys/CSPs
RSA (encrypt, decrypt) with key size equal or larger than 2048 bits	Key Wrapping	RSA private keys
Diffie-Hellman with public key size 2048 bits or larger and private key size 224 bits or 256 bits	Key Agreement	Diffie-Hellman private keys
EC Diffie-Hellman with key sizes according to P-256, P-384 and P-521 NIST curves	Key Agreement	EC Diffie-Hellman private keys
MD5	Message Digest used only in TLS	N/A

Table 6: Non-Approved but allowed Algorithms for both Modules

According to Table 2: “Comparable strengths” in NISP SP 800-57 Part1 (dated on March 8, 2007), the key sizes of RSA, Diffie-Hellman and EC Diffie-Hellman provides the following security strength for the corresponding key establishment method shown below:

1. *RSA (key wrapping; key establishment methodology provides 112 or 128 bits of encryption strength; non-compliant less than 112 bits of encryption strength)*
2. *Diffie-Hellman (key agreement; key establishment methodology provides 112 or 128 bits of encryption strength; non-compliant less than 112 bits of encryption strength)*
3. *EC Diffie-Hellman (key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength)*

However, the size alone does not determine the security strength of the RSA, Diffie-Hellman and EC Diffie-Hellman keys. Since the seed source for key generation is outside the logical boundary of the module, the following caveat is applicable:

The module generates cryptographic keys whose strengths are modified by available entropy

Per FIPS 140-2 IG G.5, the CMVP makes no statement as to the correct operation of the Modules or the security strengths of the generated keys when those Modules are ported and executed in an operational environment not listed on the validation certificate.

The Modules support the following non-FIPS 140-2 Approved algorithms, which shall not be used in the FIPS Approved mode. Any use of the non-Approved functions will cause the Modules to operate in the non-FIPS mode implicitly:

Algorithm	Usage	Keys/CSPs
RSA (encrypt, decrypt) with key size smaller than 2048 bits	key wrapping	RSA keys
RSA with key sizes not listed in	sign, verify, and key	RSA keys

Algorithm	Usage	Keys/CSPs
Table 3	generation	
DSA with key sizes not listed in Table 3	sign, verify, and key generation	DSA keys
Diffie-Hellman with key sizes not listed in Table 4	key agreement and establishment	Diffie-Hellman keys
ANSI X9.31 RNG (with AES-128 core)	random number generation	PRNG seed value and seed key 128 bits
Camellia	Encryption/decryption	Symmetric key
CAST	Encryption/decryption	Symmetric key
DES	Encryption/decryption	Symmetric key
IDEA	Encryption/decryption	Symmetric key
MD2	Hash function	N/A
MD4	Hash function	N/A
RC2	Encryption/decryption	Symmetric key
RC4	Encryption/decryption	Symmetric key
RC5	Encryption/decryption	Symmetric key
RIPEMD	Hash function	N/A
Whirlpool	Hash function	N/A

Table 7: Non-Approved Algorithms for both Modules

1.3. Cryptographic Boundary

The Modules' physical boundaries are the surface of the case of the platform (depicted in the hardware block diagram).

The Red Hat Enterprise Linux OpenSSL Cryptographic Module v5.0 logical cryptographic boundary is the shared library files and their integrity check HMAC files, which are delivered through Red Hat Package Manager (RPM) as listed below.

The openssl-libs-1.0.2k-8.el7.x86_64.rpm (64 bits) and openssl-libs-1.0.2k-8.el7.i686.rpm (32 bits) file contains the following files that are part of the module boundary:

- /usr/lib{64,}/.libcrypto.so.1.0.2k.hmac
- /usr/lib{64,}/.libssl.so.1.0.2k.hmac
- /usr/lib{64,}/libcrypto.so.1.0.2k
- /usr/lib{64,}/libssl.so.1.0.2k

The Red Hat Enterprise Linux OpenSSL Cryptographic Module v6.0 logical cryptographic boundary is the shared library files and their integrity check HMAC files, which are delivered through Red Hat Package Manager (RPM) as listed below.

The openssl-libs-1.0.2k-12.el7.x86_64.rpm (64 bits) and openssl-libs-1.0.2k-12.el7.i686.rpm (32 bits) file contains the following files that are part of the module boundary:

- /usr/lib{64,}/.libcrypto.so.1.0.2k.hmac
- /usr/lib{64,}/.libssl.so.1.0.2k.hmac
- /usr/lib{64,}/libcrypto.so.1.0.2k
- /usr/lib{64,}/libssl.so.1.0.2k

The OpenSSL RPM package of the Modules include the binary files, integrity check HMAC files, Man Pages and the OpenSSL Engines provided by the standard OpenSSL shared library. The OpenSSL Engines and their shared object files are not part of the Modules, and therefore they must not be used.

The Modules shall be installed and instantiated by the dracut-fips package with the RPM file version specified above. The dracut-fips RPM package is only used for the installation and instantiation of the Modules. This code is not active when the Modules are operational and do not provide any services to users interacting with the Modules. Therefore the dracut-fips RPM package is outside the Modules' logical boundary.

1.3.1. Hardware Block Diagram

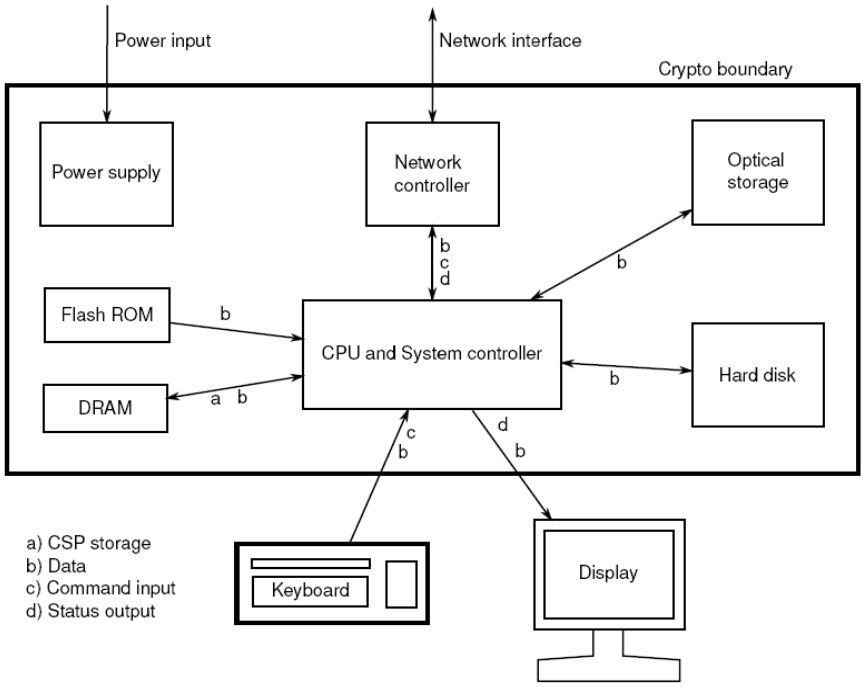

Figure 1: Hardware Block Diagram

1.3.2. Software Block Diagram

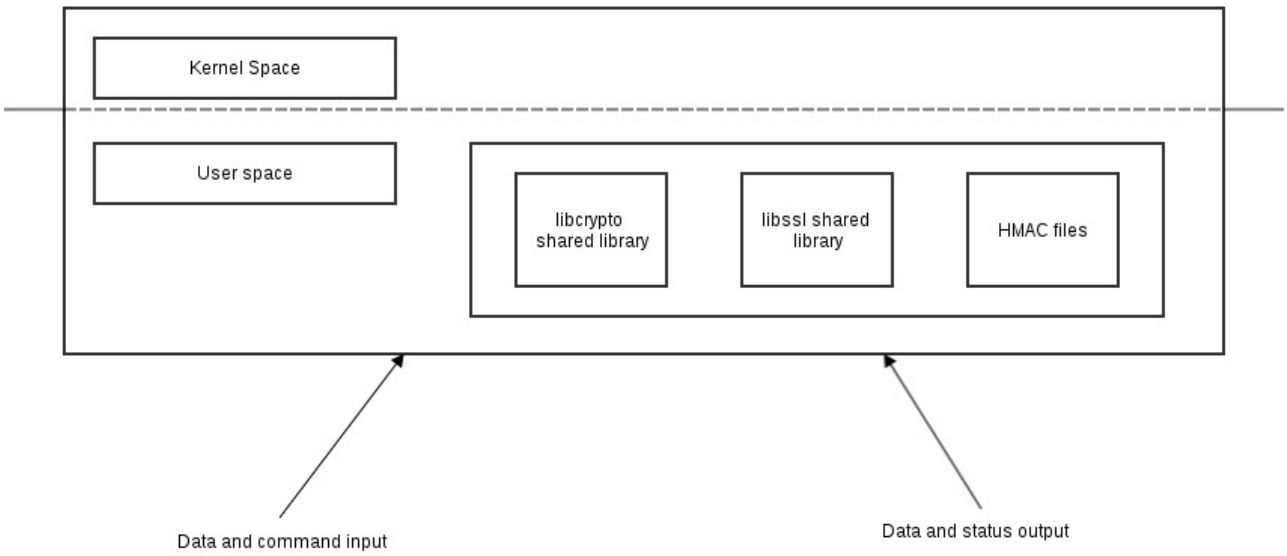

Figure 2: Software Block Diagram
(the cryptographic boundary includes the HMAC integrity files)

2. Cryptographic Modules' Ports and Interfaces

The physical ports of the Modules are the same as the computer system on which it executes. The logical interface is a C-language Application Program Interface (API).

The Data Input interface consists of the input parameters of the API functions. The Data Output interface consists of the output parameters of the API functions. The Control Input interface consists of the actual API functions. The Status Output interface includes the return values of the API functions. The ports and interfaces are shown in the following table.

FIPS Interface	Physical Port	Modules' Interfaces
Data Input	Ethernet ports	API input parameters, kernel I/O – network or files on filesystem
Data Output	Ethernet ports	API output parameters, kernel I/O – network or files on filesystem
Control Input	Keyboard, Serial port, Ethernet port, Network	API function calls, or configuration files on filesystem
Status Output	Serial port, Ethernet port, Network	API
Power Input	PC Power Supply Port	N/A

Table 8: Ports and Interfaces

3. Roles, Services and Authentication

This section defines the roles, services, and authentication mechanisms and methods with respect to the applicable FIPS 140-2 requirements.

3.1. Roles

There are two users of the Module:

- User
- Crypto Officer

The User and Crypto Officer roles are implicitly assumed by the entity accessing services implemented by the Modules. For User documentation, please refer to the man pages of ssl(3), crypto(3) as an entry into the Modules' API documentation for SSL/TLS and generic crypto support. Installation of the Modules is only done by the Crypto Officer.

3.2. Services

The Modules support services that are available to users in the various roles. All of the services are described in detail in the Modules' user documentation. The following tables show the services available to the various roles and the access to cryptographic keys and CSPs resulting from services.

The following table lists the Approved services available in FIPS Approved mode. Please refer to Table 4 and 5 for the Approval key size of each algorithm used in the services.

Service	Role	CSPs	Access
Symmetric encryption/decryption	User	AES and Triple-DES key	read/write/execute
Asymmetric key generation	User	RSA, DSA and ECDSA private key	read/write/execute
Digital signature generation and verification	User	RSA, DSA and ECDSA private key	read/write/execute
TLS network protocol	User	AES or Triple-DES key, HMAC Key	read/write/execute
TLS key agreement	User	AES or Triple-DES key, RSA, DSA or ECDSA private key, HMAC Key, Premaster Secret, Master Secret, Diffie-Hellman Private Components and EC Diffie-Hellman Private Components	read/write/execute
RSA key wrapping	User	RSA, private key	read/write/execute
Certificate Management/ Handling	User	RSA, DSA or ECDSA private key parts of certificates	read/write/execute
Keyed Hash (HMAC)	User	HMAC Key	read/write/execute
Keyed Hash (CMAC)	User	CMAC key	read/write/execute
Message digest (SHS)	User	none	read/write/execute

Service	Role	CSPs	Access
Random number generation (SP800-90A DRBG)	User	Entropy input string and seed	read/write/execute
Show status	User	none	execute
Module initialization	User	none	execute
Self-test	User	none	read/execute
Zeroize	User	All aforementioned CSPs	read/write/execute
Module installation	Crypto Officer	none	read/write

Table 9: Approved Service Details

The following table lists the non-Approved services available in non-FIPS mode. Please refer to Table 6 for the non-Approved key size of each algorithm.

Service	Role	Access
Asymmetric encryption/decryption using non-Approved RSA key size	User	read/write/execute
Symmetric encryption/decryption using non-Approved algorithms	User	read/write/execute
Hash operation using non-Approved algorithms	User	read/write/execute
Digital signature generation and verification using non-Approved RSA and DSA private key	User	read/write/execute
TLS connection using keys established by Diffie-Hellman with non-Approved key sizes	User	read/write/execute
Asymmetric key generation using non-Approved RSA and DSA key size	User	read/write/execute
Random number generation using ANSI X9.31 RNG	User	read/write/execute

Table 10: Non-Approved Service Details

Note:
The Modules do not share CSPs between an Approved mode of operation and a non-Approved mode of operation. All cryptographic keys used in the FIPS-Approved mode of operation must be generated in the FIPS-Approved mode or imported while running in the FIPS-Approved mode. If the DRBG is used for key generation for non-Approved services in non-FIPS mode, reseeding the DRBG before and after the key generation is mandatory.

More information about the services and their associated APIs can be found in the Man Pages included in the rpm packages. The evp(3) is the starting point of the Man Pages.

3.3. Operator Authentication

At security level 1, authentication is neither required nor employed. The role is implicitly assumed on entry.

3.4. Mechanism and Strength of Authentication

At security level 1, authentication is not required.

4. Physical Security

The Modules comprise of software only and thus does not claim any physical security.

5. Operational Environment

This Modules operate in a modifiable Operational Environment per the FIPS 140-2 definition.

5.3 Applicability

The Red Hat Enterprise Linux operating system is used as the basis of other products which include but are not limited to:

- Red Hat Enterprise Linux Atomic Host
- Red Hat Virtualization (RHV)
- Red Hat OpenStack Platform
- OpenShift Container Platform
- Red Hat Gluster Storage
- Red Hat Ceph Storage
- Red Hat CloudForms
- Red Hat Satellite.

Compliance is maintained for these products whenever the binary is found unchanged.

The modules operate in a modifiable operational environment per FIPS 140-2 level 1 specifications. The modules run on a commercially available general-purpose operating system executing on the hardware specified in section 1.2.

5.4. Policy

The operating system is restricted to a single operator (concurrent operators are explicitly excluded). The application that request cryptographic services is the single user of the modules, even when the application is serving multiple clients.

In the operational mode, the ptrace(2) system call, the debugger (gdb(1)), and strace(1) shall be not used.

6. Cryptographic Key Management

The application that uses the Modules is responsible for appropriate destruction and zeroization of the key material. The library provides functions for key allocation and destruction, which overwrites the memory that is occupied by the key information with “zeros” before it is deallocated.

6.1. Random Number and Key Generation

The Modules provide an SP800-90A-compliant Deterministic Random Bit Generator (DRBG) for creation of key components of asymmetric keys, and random number generation.

The /dev/urandom from the Operational Environment is used as a source of random numbers for DRBG seeds and entropy input string.

The Modules perform continuous self-tests on the output of SP800-90A DRBG to ensure that consecutive random numbers do not repeat.

The Key Generation methods implemented in the modules for Approved services in FIPS mode is compliant with [SP800-133].

For generating RSA, DSA and ECDSA keys the modules implement asymmetric key generation services compliant with [FIPS186-4]. A seed (i.e. the random value) used in asymmetric key generation is directly obtained from the [SP800-90A] DRBG.

The public and private key pairs used in the Diffie-Hellman and EC Diffie-Hellman KAS are generated internally by the modules using the same DSA and ECDSA key generation compliant with [FIPS186-4] which is compliant with [SP800-56A].

6.2. Key/Critical Security Parameter (CSP)

An authorized application as user (i.e., the User role) has access to all key data generated during the operation of the Modules. The following table summarizes the Critical Security Parameters (CSPs) that are used by the cryptographic services implemented in the modules:

Key/CSP	Generation	Storage	Zeroization
Symmetric Keys (AES, Triple-DES)	N/A(passed in as API input parameter) Alternatively, key can be established during a TLS handshake	RAM	EVP_CIPHER_CTX_cleanup()
MAC Keys (HMAC, CMAC)			HMAC_CTX_cleanup() CMAC_CTX_cleanup()
Asymmetric Private Keys (RSA, DSA, ECDSA)	Generated using FIPS 186-4 key generation method and the random value used in the key generation is generated using SP800-90A DRBG.	RAM	DSA_free(), RSA_free(), EC_GROUP_clear_free() and EC_POINT_clear_free()
Diffie-Hellman	Generated as specified in	RAM	DH_free()

Key/CSP	Generation	Storage	Zeroization
Private Components	SP800-56A and the random value used in the key generation is generated using SP800-90A DRBG.		
EC Diffie-Hellman Private Components			EC_GROUP_clear_free() and EC_POINT_clear_free()
SP 800-90A DRBG seed and entropy input	Obtained from NDRNG	RAM	FIPS_drbg_free()
TLS Pre-Master Secret and Master Secret	Established during the TLS handshake	RAM	SSL_free() and SSL_clear()

Table 11: Key Life Cycle

6.3. Key/CSP Storage

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

6.4. Key/CSP Zeroization

The memory occupied by keys is allocated by regular libc malloc/calloc() calls. The application is responsible for calling the appropriate destruction functions from the OpenSSL API. The destruction functions then overwrite the memory occupied by keys with pre-defined values and deallocates the memory with the free() call. In case of abnormal termination, or swap in/out of a physical memory page of a process, the keys in physical memory are overwritten by the Linux kernel before the physical memory is allocated to another process.

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

MARKETING NAME..... PowerEdge R630
REGULATORY MODEL..... E26S
REGULATORY TYPE..... E26S001
EFFECTIVE DATE..... September 03, 2014
EMC EMISSIONS CLASS..... Class A

7.1. Statement of compliance

This product has been determined to be compliant with the applicable standards, regulations, and directives for the countries where the product is marketed. The product is affixed with regulatory marking and text as necessary for the country/agency. Generally, Information Technology Equipment (ITE) product compliance is based on IEC and CISPR standards and their national equivalent such as Product Safety, IEC 60950-1 and European Norm EN 60950-1 or EMC, CISPR 22/CISPR 24 and EN 55022/55024. Dell products have been verified to comply with the EU RoHS Directive 2011/65/EU. Dell products do not contain any of the restricted substances in concentrations and applications not permitted by the RoHS Directive.

8. Self-Tests

FIPS 140-2 requires that the Modules perform self-tests to ensure the integrity of the Modules, and the correctness of the cryptographic functionality at start up. In addition, some functions require continuous verification of function, such as the Random Number Generator. All of these tests are listed and described in this section. No operator intervention is required during the running of the self-tests.

See section 9.3 for descriptions of possible self-test errors and recovery procedures.

8.1. Power-Up Tests

The Modules perform both power-up self-tests (at module initialization) and continuous conditional tests (during operation). The power-up self test start with the integrity test, where the FIPS_mode_set() function verifies the integrity of the runtime executable using a HMAC SHA-256 digest, which is computed at build time. If this computed HMAC SHA-256 digest matches the stored, known digest, then the rest of the power-up self-test (consisting of the algorithm-specific Pairwise Consistency and Known Answer Tests) is performed. Input, output, and cryptographic functions cannot be performed while the Modules are in a self-test or error state because the Modules are single-threaded and will not return to the calling application until the power-up self-tests are complete. After successful completion of the power-up tests, the modules are loaded and cryptographic functions are available for use. If the power-up self-tests fail, subsequent calls to the Modules will also fail - thus no further cryptographic operations are possible.

Algorithm	Test
AES	KAT, encryption and decryption are tested separately
Triple-DES	KAT, encryption and decryption are tested separately
DSA	Pairwise consistency test (PCT), sign and verify
RSA	KAT, signature generation and verification are tested separately
ECDSA	PCT, sign and verify
Diffie-Hellman	Primitive "Z" Computation KAT
EC Diffie-Hellman	Primitive "Z" Computation KAT
SP 800-90A CTR_DRBG	KAT
SP 800-90A Hash_DRBG	KAT
SP 800-90A DRBG_HMAC	KAT
HMAC-SHA-1, -244, -256, -384, -512	KAT
SHA-1, -224, -256, -384, -512	KAT
CMAC	KAT
Module integrity	HMAC-SHA-256

Table 12: Modules' Self-Tests

8.2. Conditional Tests

Algorithm	Test
DSA	PCT: signature generation and verification
ECDSA	PCT: signature generation and verification
RSA	PCT: signature generation and verification, encryption and decryption
DRBG SP800-90A	Continuous Random Number Generation Test

Table 13: Modules' Conditional Tests

9. Guidance

9.1. Crypto Officer Guidance

The version of the RPM containing the FIPS validated Modules is stated in section 1. The RPM package of the Modules can be installed by standard tools recommended for the installation of RPM packages on a Red Hat Enterprise Linux system (for example, yum, rpm, and the RHN remote management tool). The integrity of the RPM is automatically verified during the installation of the Modules and the Crypto Officer shall not install the RPM file if the RPM tool indicates an integrity error.

The OpenSSL static libraries libcrypto.a and libssl.a in openssl-static package are not approved to be used. The applications must be dynamically linked to run the OpenSSL.

The RPM package of the Modules can be installed by standard tools recommended for the installation of RPM packages on a Red Hat Enterprise Linux system (for example, yum, rpm, and the RHN remote management tool).For proper operation of the in-module integrity verification, the prelink has to be disabled.

1 Disable the prelink:
```
# sed -i 's/PRELINKING=yes/PRELINKING=no/g' /etc/sysconfig/prelink
```
2 Run following command to return binaries to a non-prelink state:
```
# /usr/sbin/prelink -ua
```

Crypto officer should perform the following for Modules initialization:

1. Install the dracut-fips package:
```
# yum install dracut-fips
```
2. Recreate the INITRAMFS image:
```
# dracut -f
```

After regenerating the initramfs, the Crypto Officer has to append the following string to the kernel command line by changing the setting in the boot loader:
```
fips=1
```

If /boot or /boot/efi resides on a separate partition, the kernel parameter boot=<partition of /boot or /boot/efi> must be supplied. The partition can be identified with the command
```
"df /boot"
```
or
```
"df /boot/efi"
```

respectively. For example:
```
$ df /boot
Filesystem        1K-blocks   Used       Available         Use%  Mounted on
/dev/sda1          233191     30454      190296            14%   /boot
```

The partition of /boot is located on /dev/sda1 in this example. Therefore, the following string needs to be appended to the kernel command line:
```
"boot=/dev/sda1"
```

Reboot to apply these settings.

The next step is to check the presence of the configuration file /proc/sys/crypto/fips_enabled and make sure it contains value 1.

0.0.1 The version of the RPM containing the validated Modules is the version listed in chapter 1. The integrity of the RPM is automatically verified during the installation of the Modules and the Crypto Officer shall not install the RPM file if the RPM tool indicates an integrity error.

9.2. User Guidance

To operate the Modules in FIPS Approved mode, the user should use services and security functions listed in Table 6. Any use of non-approved services will put the modules in the non-FIPS mode implicitly.

Interpretation of the return code is the responsibility of the host application.

ENGINE_register_*, ENGINE_set_default_* and FIPS_mode_set(0) function calls are prohibited.

9.2.1. TLS and Diffie-Hellman

The TLS protocol implementation provides both, the server and the client sides. As required by SP800-131A, Diffie-Helllman with keys smaller than 2048 bits must not be used any more. The TLS protocol lacks the support to negotiate the used Diffie-Hellman key sizes. To ensure full support for all TLS protocol versions, the TLS client implementation of the cryptographic Modules accepts Diffie-Hellman key sizes smaller than 2048 bits offered by the TLS server. The TLS server implementation of the cryptographic Modules allow the application to set the Diffie-Hellman key size. The server side must always set the DH parameters with the API call of

```
SSL_CTX_set_tmp_dh(ctx, dh)
```

For complying with the requirement to not allow Diffie-Hellman key sizes smaller than 2048 bits, the Crypto Officer must ensure that:

- in case the Modules are used as TLS server, the Diffie-Hellman parameters (dh argument) of the aforementioned API call must be 2048 bits or larger;
- in case the Modules are used as TLS client, the TLS server must be configured to only offer Diffie-Hellman keys of 2048 bits or larger.

9.2.2. AES XTS Guidance

The length of a single data unit encrypted with the XTS-AES shall not exceed 2^{20} AES blocks that is 16MB of data.

9.2.3. Random Number Generator

The OpenSSL API call of RAND_cleanup must not be used. This call will cleanup the internal DRBG state. This call also replaces the DRBG instance with the non-FIPS Approved SSLeay Deterministic Random Number Generator when using the RAND_* API calls.

9.2.4. AES GCM IV

In case the Modules' power is lost and then restored, the key used for the AES GCM

encryption/decryption shall be re-distributed.

9.2.5. Triple-DES Keys

According to IG A.13, the same Triple-DES key shall not be used to encrypt more than 2^{28} 64-bit blocks of data.

9.2.6. RSA and DSA Keys

The Modules allow the use of 1024 bit RSA and DSA keys for legacy purposes, including signature generation.

RSA and DSA must be used with either 2048 bit keys or 3072 bit keys because larger key sizes have not been CAVS tested. To comply with the requirements of FIPS 140-2, a user must therefore only use keys with 2048 bits or 3072 bits in FIPS Approved mode.

Application can enforce the key generation bit length restriction for RSA and DSA keys by setting the environment variable OPENSSL_ENFORCE_MODULUS_BITS. This environment variable ensures that 1024 bit keys cannot be generated.

9.2.7. Handling Self-Test Errors

The effects of self-test failures in the Modules differ depending on the type of self-test that failed.
Non-fatal self-test errors transition the Modules into an error state. The application must be restarted to recover from these errors. The non-fatal self-test errors are:

- FIPS_R_FINGERPRINT_DOES_NOT_MATCH - The integrity verification check failed
- FIPS_R_FIPS_SELFTEST_FAILED - a known answer test failed
- FIPS_R_SELFTEST_FAILED - a known answer test failed
- FIPS_R_TEST_FAILURE – a known answer test failed (RSA); pairwise consistency test failed (DSA)
- FIPS_R_PAIRWISE_TEST_FAILED – a pairwise consistency test during DSA or RSA key generation failed
- RAND_R_PRNG_STUCK – the random number generator generated two same consecutive 128 bit values

These errors are reported through the regular ERR interface of the Modules and can be queried by functions such as ERR_get_error(). See the OpenSSL manual page for the function description.

When a fatal error occurs (a self-test or conditional test has failed), the Modules enter an error state. Any calls to a crypto function of the Modules returns an error with the error message: 'FATAL FIPS SELFTEST FAILURE' printed to stderr and the Modules are terminated with the abort() call.

The only way to recover from a fatal error is to restart the Modules. If failures persist, the Modules must be reinstalled. If downloading the software, make sure to verify the package hash to confirm a proper download.

10. Mitigation of Other Attacks

RSA is vulnerable to timing attacks. In a setup where attackers can measure the time of RSA decryption or signature operations, blinding must be used to protect the RSA operation from that attack.

The API function of RSA_blinding_on turns blinding on for key rsa and generates a random blinding factor. The random number generator must be seeded prior to calling RSA_blinding_on.
Weak Triple-DES keys are detected as follows:

```
/* Weak and semi week keys as taken from
 * %A D.W. Davies
 * %A W.L. Price
 * %T Security for Computer Networks
 * %I John Wiley & Sons
 * %D 1984
 * Many thanks to smb@ulysses.att.com (Steven Bellovin) for the reference
 * (and actual cblock values).
 */
#define NUM_WEAK_KEY    16
static const DES_cblock weak_keys[NUM_WEAK_KEY]={
        /* weak keys */
        {0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01},
        {0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE},
        {0x1F,0x1F,0x1F,0x1F,0x0E,0x0E,0x0E,0x0E},
        {0xE0,0xE0,0xE0,0xE0,0xF1,0xF1,0xF1,0xF1},
        /* semi-weak keys */
        {0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE},
        {0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01},
        {0x1F,0xE0,0x1F,0xE0,0x0E,0xF1,0x0E,0xF1},
        {0xE0,0x1F,0xE0,0x1F,0xF1,0x0E,0xF1,0x0E},
        {0x01,0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1},
        {0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1,0x01},
        {0x1F,0xFE,0x1F,0xFE,0x0E,0xFE,0x0E,0xFE},
        {0xFE,0x1F,0xFE,0x1F,0xFE,0x0E,0xFE,0x0E},
        {0x01,0x1F,0x01,0x1F,0x01,0x0E,0x01,0x0E},
        {0x1F,0x01,0x1F,0x01,0x0E,0x01,0x0E,0x01},
        {0xE0,0xFE,0xE0,0xFE,0xF1,0xFE,0xF1,0xFE},
        {0xFE,0xE0,0xFE,0xE0,0xFE,0xF1,0xFE,0xF1}};
```

Please note that there is no weak key detection by default. The caller can explicitly set the DES_check_key to 1 or call DES_check_key_parity() and/or DES_is_weak_key() functions on its own.

11. Glossary and Abbreviations

AES	Advanced Encryption Specification
CAVP	Cryptographic Algorithm Validation Program
CBC	Cypher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Auhentication Code
CFB	Cypher Feedback
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
ECB	Electronic Code Book
FSM	Finite State Model
HMAC	Hash Message Authentication Code
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
NVLAP	National Voluntary Laboratory Accreditation Program
OFB	Output Feedback
OS	Operating System
PRNG	Pseudo Random Number Generator
RHEL	Red Hat Enterprise Linux
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard

12. References

[1] OpenSSL man pages where crypto(3) provides the introduction and link to all OpenSSL APIs regarding the cryptographic operation and ssl(3) to all OpenSSL APIs regarding the SSL/TLS protocol family
[2] FIPS 140-2 Standard, http://csrc.nist.gov/groups/STM/cmvp/standards.html
[3] FIPS 140-2 Implementation Guidance, http://csrc.nist.gov/groups/STM/cmvp/standards.html
[4] FIPS 140-2 Derived Test Requirements,http://csrc.nist.gov/groups/STM/cmvp/standards.html
[5] FIPS 197 Advanced Encryption Standard, http://csrc.nist.gov/publications/PubsFIPS.html
[6] FIPS 180-4 Secure Hash Standard, http://csrc.nist.gov/publications/PubsFIPS.html
[7] FIPS 198-1 The Keyed-Hash Message Authentication Code (HMAC), http://csrc.nist.gov/publications/PubsFIPS.html
[8] FIPS 186-4 Digital Signature Standard (DSS), http://csrc.nist.gov/publications/PubsFIPS.html
[9] ANSI X9.52:1998 Triple Data Encryption Algorithm Modes of Operation, http://webstore.ansi.org/FindStandards.aspx?Action=displaydept&DeptID=80&Acro=X9&DpName=X9,%20Inc.
[10] NIST SP 800-67 Revision 1, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, http://csrc.nist.gov/publications/PubsFIPS.html
[11] NIST SP 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, http://csrc.nist.gov/publications/PubsFIPS.html
[12] NIST SP 800-38C, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, http://csrc.nist.gov/publications/PubsFIPS.html
[13] NIST SP 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, http://csrc.nist.gov/publications/PubsFIPS.html
[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/PubsFIPS.html
[15] NIST SP 800-56A, Recommendation for Pair-Wise Key Establishment Schemes using Discrete Logarithm Cryptography (Revised), http://csrc.nist.gov/publications/PubsFIPS.html
[16] NIST SP 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, http://csrc.nist.gov/publications/PubsFIPS.html

