# Red Hat Enterprise Linux 8 OpenSSL Cryptographic Module

## version rhel8.20200305.1

# FIPS 140-2 Non-proprietary Security Policy

## version 1.3

## Last Update: 2021-03-02

# Table of Contents

# 1 Cryptographic Modules' Specifications

This document is the non-proprietary Security Policy for the Red Hat Enterprise Linux 8 OpenSSL Cryptographic Module version rhel8.20200305.1 and was prepared as part of the requirements for conformance to Federal Information Processing Standard (FIPS) 140-2, Level 1.

## 1.1 Description of the Module

The Red Hat Enterprise Linux 8 OpenSSL Cryptographic Module (hereafter referred to as the "Module") is a software libraries supporting FIPS 140-2 Approved cryptographic algorithms. The code base of the Module is formed in a combination of standard OpenSSL shared library, OpenSSL FIPS Object Module and development work by Red Hat. The Module provides 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 |
| Overall Level | 1 |

*Table 1: Security Level of the Module*

The Red Hat Enterprise Linux 8 OpenSSL Cryptographic Module has been tested on the following multi-chip standalone platforms:

| Manufacturer | Model | O/S & Ver. | Processor |
|---|---|---|---|
| Dell | PowerEdge R430 | Red Hat Enterprise Linux 8 | Intel(R) Xeon(R) E5 |

*Table 2: Test Platform*

The Module has been tested for the following configurations:

- 64-bit library, x86_64 with and without AES-NI enabled.

To operate the Module, 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 Module supports 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 Module verifies 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 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.

The Red Hat Enterprise Linux 8 OpenSSL Cryptographic Module supports the following FIPS 140-2 Approved algorithms in FIPS Approved mode:

| Algorithm | Validation Certificate | Standards/Usage | Keys/CSPs |
|---|---|---|---|
| AES | Certs. #A562, #A563, #A564, #A565, #A566, #A567, #A568, #A569 and #A570 | FIPS 197 (AES)<br><br>SP 800-38D (GCM)<br><br>Encryption and Decryption | AES keys 128 bits, 192 bits (except XTS-AES) and 256 bits |
| | Certs. #A549, #A550, #A551, #A552, #A554, #A555 and #A556 | FIPS 197 (AES)<br>SP 800-38A (ECB)<br><br>Encryption and Decryption | |
| | Certs. #A554, #A555 and #A556 | FIPS 197 (AES)<br>SP 800-38A (CBC, OFB, CFB1, CFB8, CFB128, CTR)<br><br>SP 800-38C (CCM)<br>SP 800-38E (XTS)<br>SP 800-38F (KW, KWP)<br><br>Encryption and Decryption | |
| | Certs. #A554, #A555 and #A556 | SP 800-38B (CMAC) | Mac length of 128 bits |
| Triple-DES | Certs. #A549, #A550, #A551, #A552 and #A557 | SP 800-67<br><br>SP 800-38A (ECB)<br><br>Encryption and Decryption | Triple-DES keys 168 bits |

| Algorithm | Validation Certificate | Standards/Usage | Keys/CSPs |
|---|---|---|---|
| Triple-DES | Cert. #A557 | SP 800-67<br><br>SP 800-38A (CBC, OFB, CFB1, CFB8, CFB64)<br>SP 800-38B (CMAC)<br><br>Encryption and Decryption | Triple-DES keys 168 bits |
| DSA | Certs. #A558, #A559, #A560 and #A561 | FIPS 186-4<br><br>Domain Parameters Generation and Verification, Key Generation, Signature Generation | DSA keys:<br>• L=2048, N=224<br>• L=2048, N=256<br>• L=3072, N=256 |
| | | FIPS 186-4<br><br>Signature Verification | DSA keys:<br>• L=1024, N=160<br>• L=2048, N=224<br>• L=2048, N=256<br>• L=3072, N=256<br><br>Note: 1024 bit DSA signature verification is legacy-use. |
| RSA | Certs[1]. #A558, #A559, #A560 and #A561 | FIPS 186-4 Appendix B.3.3<br><br>Key Generation | RSA keys<br>• 2048 bits<br>• 3072 bits<br>• 4096 bits |
| | | Signature Generation (PKCS#1 v1.5 and PSS)<br><br>SHA-1,SHA-224,SHA-256,SHA-384,SHA-512 | RSA keys:<br>• 2048 bits<br>• 3072 bits<br>• 4096 bits |
| | | Signature Verification (PKCS#1 v1.5 and PSS)<br><br>SHA-1,SHA-224,SHA-256,SHA-384,SHA-512 | RSA keys:<br>• 1024 bits<br>• 2048 bits<br>• 3072 bits<br>• 4096 bits<br><br>Note: 1024 bit RSA signature verification is legacy-use. |
| | | Signature Generation (ANSI X9.31) | RSA keys:<br>• 2048 bits |

1 Use of SHA-1 for signature generation is allowed in the context of TLS protocol per SP800-52.

| Algorithm | Validation Certificate | Standards/Usage | Keys/CSPs |
|---|---|---|---|
| | | SHA-1,SHA-256,SHA-384,SHA-512 | • 3072 bits<br>• 4096 bits |
| | | Signature Verification (ANSI X9.31)<br><br>SHA-1,SHA-256,SHA-384,SHA-512 | RSA keys:<br>• 1024 bits<br>• 2048 bits<br>• 3072 bits<br>• 4096 bits<br><br>Note: 1024 bit RSA signature verification is legacy-use. |
| ECDSA | Certs. #A558, #A559, #A560 and #A561 | FIPS 186-4<br><br>Key Pair Generation and Public Key Verification | ECDSA keys based on P-256, P-384, or P-521 curve |
| | | FIPS 186-4<br><br>Signature Generation<br><br>SHA-224,SHA-256,SHA-384,SHA-512 | ECDSA keys based on P-224, P-256, P-384, or P-521 curve |
| | | FIPS 186-4<br><br>Signature Verification<br><br>SHA-1,SHA-224,SHA-256,SHA-384,SHA-512 | |
| DRBG | Certs. #A554, #A555, #A556 and #A581 | SP 800-90A (CTR_DRBG)<br><br>Random Number Generation<br><br>AES-128,AES-192,AES-256 | Entropy input string, seed, C, V and Key |
| | Cert. #A581 | SP 800-90A (Hash_DRBG, HMAC_DRBG)<br><br>Random Number Generation<br><br>SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 | |

| Algorithm | Validation Certificate | Standards/Usage | Keys/CSPs |
|---|---|---|---|
| SHS | Certs. #A558, #A559, #A560 and #A561 | FIPS 180-4 (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512)<br><br>Hashing | N/A |
| SHA-3 | Certs. #A546, #A547 and #A548 | FIPS 202 (SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE-128, SHAKE-256)<br><br>Hashing | N/A |
| HMAC | Certs. #A558, #A559, #A560 and #A561 | FIPS 198-1 (HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512,<br><br>Message Integrity | At least 112 bits HMAC Key |
| | Certs. #A546, #A547 and #A548 | HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512)<br><br>Message Integrity | |
| SP 800-56A DLC primitive Diffie-Hellman (CVL) | CVL Cert. #A580 | SP 800-56A<br><br>Shared Secret Computation | Public key size 2048 bits or larger, and private key size 224 bits or 256 bits<br><br>shared secret |
| SP 800-56A DLC primitive EC Diffie-Hellman (CVL) | CVL Certs. #A558, #A559, #A560 and #A561 | | NIST curves P-224, P-256, P-384, P-521<br><br>shared secret |
| SP 800-56A DLC primitive EC Diffie-Hellman (CVL) | CVL Certs. #A558, #A559, #A560 and #A561 | SP 800-56A<br><br>EC CDH Component | 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 (CVL) | CVL Certs. #A558, #A559, #A560 and #A561 | SP800-135<br><br>Key Derivation in TLS | TLS Pre-Master Secret and Master Secret |

| Algorithm | Validation Certificate | Standards/Usage | Keys/CSPs |
|---|---|---|---|
| PBKDF | (vendor affirmed)[2] | SP 800-132 (SHA-1, SHA-224, SHA-256, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512) | PBKDF password<br>PBKDF Derived Key |
| SSH KDF | CVL Certs. #A549, #A550, #A551 and #A552 | SP 800-135 | SSH-KDF Derived Key |
| Single-step KDF | (KDA vendor affirmed) | SP 800-56C (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512) | Derived key |
| KTS | Certs. #A554, #A555, #A556, #A562, #A563, #A564, #A565, #A566, #A567, #A568, #A569 and #A570 | SP800-38F<br><br>AES KW, KWP<br>AES CCM<br>AES GCM | AES keys 128, 192, 256 bits |
| | Certs. #A554, #A555, #A556, #A558, #A559, #A560 and #A561 | SP800-38F<br><br>AES CBC and HMAC | AES keys 128, 192, 256 bits |
| | Certs. #A557, #A558, #A559, #A560 and #A561 | SP800-38F<br><br>Triple-DES CBC and HMAC | Triple-DES keys 168 bits<br><br>Note: with encryption strength of 112 bits |

*Table 3: Approved Algorithms*

The Module supports 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; key establishment methodology provides between 112 and 256 bits of encryption strength | RSA private key |
| Diffie-Hellman with public key size 2048 bits or larger and private key size 224 bits or 256 bits | key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength | Diffie-Hellman private key |
| EC Diffie-Hellman with key sizes according to P-256, P-384 and P-521 NIST curves | key agreement; key establishment methodology provides between 128 and 256 bits of encryption strength | EC Diffie-Hellman private key |
| MD5 | Message Digest used only in TLS | N/A |
| NDRNG | Seeding the module's DRBG | Internal state |

2 Even though the PBKDF has CAVP certificates (#A546, #A547, #A548, #A558, #A559, #A560 and #A561), the KAT is not implemented.

*Table 4: Non-Approved but allowed Algorithms*

Per FIPS 140-2 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 those Module are ported and executed in an operational environment not listed on the validation certificate.

The Module supports 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 Module 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 Table 3 | sign, verify, and key generation | RSA keys |
| Digitial Signature Generation (DSA, ECDSA and RSA) using SHA-1 | sign | DSA, ECDSA and RSA keys |
| 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 | Diffie-Hellman keys |
| ANSI X9.31 RNG (with AES-128 core) | random number generation | PRNG seed value and seed key 128 bits |
| AES-OCB | Authenticated Encryption/Decryption | Symmetric key |
| Camellia | Encryption/decryption | Symmetric key |
| CAST | Encryption/decryption | Symmetric key |
| DES | Encryption/decryption | Symmetric key |
| IDEA | Encryption/decryption | Symmetric key |
| KBKDF (Cert. #A553)[3] | Key derivation | Derived key |
| RFC-3961 KDF | Key derivation | Derived 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 5: Non-Approved Algorithms*

3 Even though the KBKDF has CAVP certificates, the KAT is not implemented.

## 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 8 OpenSSL Cryptographic Module 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-1.1.1c-15.el8.x86_64.rpm (64 bits) file contains the following files that are part of the module boundary:

- /usr/lib64/.libcrypto.so.1.1.1c.hmac
- /usr/lib64/.libssl.so.1.1.1c.hmac
- /usr/lib64/libcrypto.so.1.1.1c
- /usr/lib64/libssl.so.1.1.1c.

The OpenSSL RPM package of the Module includes 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 Module, and therefore they must not be used.

### 1.3.1 Hardware Block Diagram

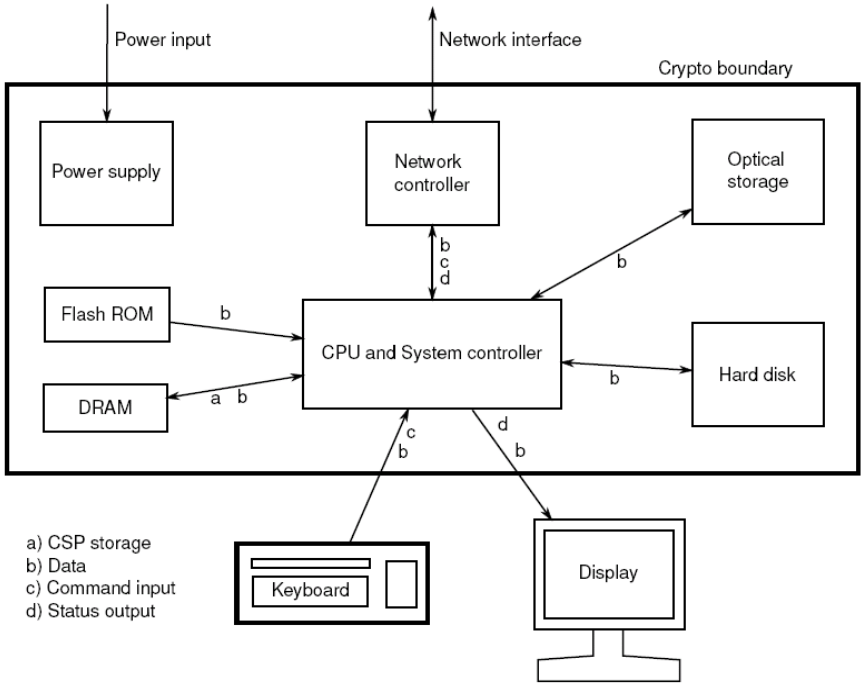


*Figure 1: Hardware Block Diagram*

### 1.3.2 Software Block Diagram

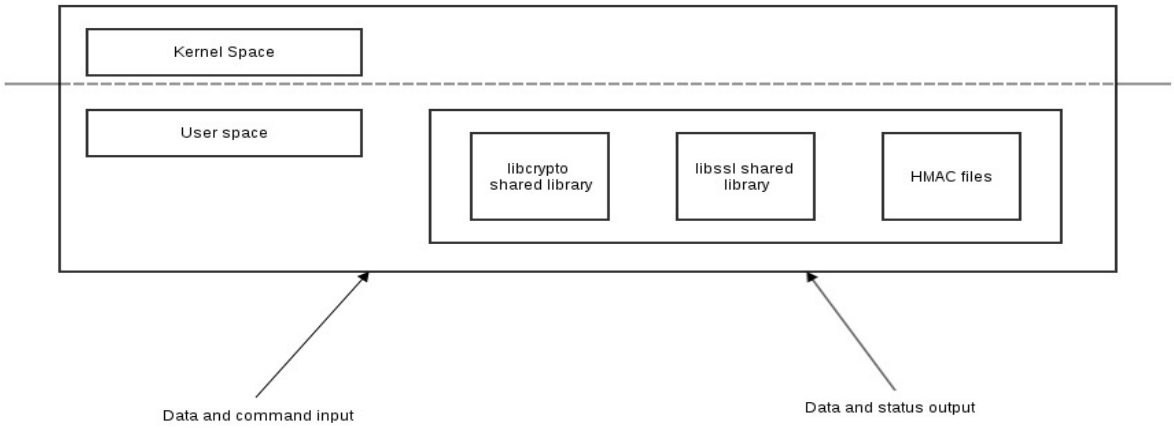


*Figure 2: Software Block Diagram*
*(the cryptographic boundary includes the HMAC integrity files)*

# 2 Cryptographic Modules' Ports and Interfaces

The physical ports of the Module 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 6: 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 Module. 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 Module is only done by the Crypto Officer.

## 3.2 Services

The Module supports 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 3 and Table 4 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/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/execute |
| TLS network protocol | User | AES or Triple-DES key, HMAC Key | read/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 |
| Shared secret computation | User | Diffiie-Hellman shared secret and EC Diffie-Hellman shared secret | read |
| RSA key wrapping | User | RSA, private key | read/execute |
| Certificate Management/ Handling | User | RSA, DSA or ECDSA private key parts of certificates | read/write/execute |
| Keyed Hash (HMAC) | User | HMAC Key | read/execute |
| Keyed Hash (CMAC) | User | CMAC key | read/execute |
| Message digest (SHS) | User | none | N/A |

| Service | Role | CSPs | Access |
|---|---|---|---|
| Random number generation (SP800-90A DRBG) | User | Entropy input string and seed (C, K and V values) | read/write/execute |
| Key Derivation (through PBKDF or SSH-KDF or Single-step KDF) | User | PBKDF password and derived key and SSH-KDF derived key and Single-step KDF derived key | read/write/execute |
| Show status | User | none | N/A |
| Module initialization | User | none | N/A |
| Self-test | User | none | N/A |
| Zeroize | User | All aforementioned CSPs | read/write/execute |
| Module installation | Crypto Officer | none | N/A |

*Table 7: 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/execute |
| Symmetric encryption/decryption using non-Approved algorithms | User | read/execute |
| Hash operation using non-Approved algorithms | User | read/execute |
| Digital signature generation and verification using non-Approved RSA and DSA private key sizes | User | read/execute |
| Digital signature generation using SHA-1 | User | read/execute |
| TLS connection using keys established by Diffie-Hellman with non-Approved key sizes | User | read/write/execute |
| TLS connection using keys established by RSA with key size less than 2048 bits | User | read/write/execute |
| Asymmetric key generation using non-Approved RSA and DSA key sizes | User | read/write/execute |
| Random number generation using ANSI X9.31 RNG | User | read/write/execute |
| Key derivation using SP 800-108 KBKDF | User | read/write |
| Key derivation using RFC-3961 | User | read/write |

*Table 8: Non-Approved Service Details*

Note:
The Module does 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 Module is comprised of software only and thus does not claim any physical security.

# 5 Operational Environment

## 5.1 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 CoreOS
- 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 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 section 1.2.

## 5.2 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 module, 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

## 6.1 Random Number and Key Generation

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

The seeding (and automatic reseeding) of the DRBG is done with getrandom().

The module performs the health tests for the SP800-90A DRBG as defined per section 11.3 of SP800-90A.

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

For generating RSA, DSA and ECDSA keys the module implements 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 module using the same DSA and ECDSA key generation compliant with [FIPS186-4] which is compliant with [SP800-56A].

The NDRNG provides 128 bits of entropy to the DRBG. Therefore, the following caveat applies:

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

## 6.2 Key Establishment

The module provides Diffie-Hellman and EC Diffie-Hellman key agreement In addition, the module offers AES key wrapping per [SP800-38F] (using GCM, CCM, AES-KW, AES-KWP and a combination of any approved block chaining modes with HMAC for authentication), Triple-DES key wrapping per [SP800-38F] (using a combination of any approved block chaining modes with HMAC for authentication), and RSA key wrapping (encapsulation) using public key encryption and private key decryption primitives as allowed by [FIPS140-2_IG] D.9.

AES, RSA, Diffie-Hellman and EC Diffie-Hellman) provide the following security strengths:

- AES: key wrapping provides between 128 and 256 bits of encryption strength.
- Triple-DES: key wrapping provides 112 bits of encryption strength.
- RSA: key wrapping provides between 112 and 256 bits of encryption strength.
- Diffie-Hellman: key agreement provides between 112 and 256 bits of encryption strength.
- EC Diffie-Hellman: key agreement provides between 128 and 256 bits of encryption strength.

## 6.3 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 Module. The following table summarizes the Critical Security Parameters (CSPs) that are used by the cryptographic services implemented in the module:

| Key/CSP | Generation | Storage | Zeroization |
|---|---|---|---|
| AES Symmetric Key | N/A(passed in as API input parameter) Alternatively, key can be established during a TLS handshake | RAM | EVP_CIPHER_CTX_cleanup() |
| Triple-DES Symmetric Key | | | |
| HMAC Key | | | HMAC_CTX_cleanup() |
| CMAC Key | N/A(passed in as API input parameter) | RAM | CMAC_CTX_cleanup() |
| RSA Private Key | Generated using FIPS 186-4 key generation method and the random value used in the key generation is generated using SP800-90A DRBG. | RAM | RSA_free() |
| DSA Private Key | | | DSA_free() |
| ECDSA Private Key | | | EC_KEY_free() |
| Diffie-Hellman Private Components | Generated as specified in SP800-56A and the random value used in the key generation is generated using SP800-90A DRBG. | RAM | DH_free() |
| EC Diffie-Hellman Private Components | | | EC_KEY_free() |
| Shared secret. | Generated during the Diffie-Hellman or EC Diffie-Hellman shared secret computation | RAM | DH_free(), EC_KEY_free() |
| SP 800-90A DRBG seed and entropy input ) | Obtained from NDRNG | RAM | FIPS_drbg_free() |
| SP 800-90A DRBG internal values (C, K, V values) | Derived from the seed and entropy input using SP800-90A mechanisms | | |
| TLS Pre-Master Secret and Master Secret | Established during the TLS handshake | RAM | SSL_free() and SSL_clear() |
| PBKDF Password | N/A | RAM | kdf_pbkdf2_free() |
| PBKDF Derived Key | SP800-132 PBKDF mechanisms | | kdf_pbkdf2_free() |
| SSH-KDF Derived Key | Derived SP800-135 SSH KDF mechanisms | RAM | kdf_sshkdf_free() |
| Single-step KDF derived key | SP 800-56C KDF mechanism | RAM | sskdf_free() |

*Table 9: Key Life Cycle*

## 6.4 Key/CSP Storage

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

## 6.5 Key/CSP Zeroization

The application that uses the Module 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.

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 R430
REGULATORY MODEL..... E28S
REGULATORY TYPE..... E28S001
EFFECTIVE DATE..... December 02, 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 Module performs self-tests to ensure the integrity of the Module, 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 9.2.6 for descriptions of possible self-test errors and recovery procedures.

## 8.1 Power-Up Tests

The Module performs 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 Module is in a self-test or error state because the Module is 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 module is loaded and cryptographic functions are available for use. If the power-up self-tests fail the module will enter the error state, subsequent calls to the Module will fail - thus no further cryptographic operations are possible.

| Algorithm | Test |
|---|---|
| AES (ECB, CCM, GCM, XTS modes) | 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 |
| SP 800-90-A DRBG | Health test per section 11.3 of SP 800-90A DRBG |
| HMAC-SHA-1, 224 -256, -384, -512 | KAT |
| SHA-1, -256, -512 | KAT |
| SHA3-256, SHA3-512, SHAKE-128, SHAKE-256 | KAT |
| CMAC (AES and Triple-DES) | KAT |
| Module integrity | HMAC-SHA-256 |

*Table 10: 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 |

*Table 11: Modules' Conditional Tests*

# 9 Guidance

## 9.1 Crypto Officer Guidance

The version of the RPM containing the FIPS validated Module is stated in section 1. The RPM package of the Module 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 Module 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.

### 9.1.1 FIPS module installation instructions

**Recommended method**

The system-wide cryptographic policies package (crypto-policies) contains a tool that completes the installation of cryptographic modules and enables self-checks in accordance with the requirements of Federal Information Processing Standard (FIPS) Publication 140-2. We call this step “FIPS enablement”. The tool named fips-mode-setup installs and enables or disables all the validated FIPS modules and it is the recommended method to install and configure a RHEL-8 system.

1. To switch the system to FIPS enablement in RHEL 8:

```
# fips-mode-setup --enable
Setting system policy to FIPS
FIPS mode will be enabled.
Please reboot the system for the setting to take effect.
```

2. Restart your system:

```
# reboot
```

3. After the restart, you can check the current state:

```
# fips-mode-setup --check
FIPS mode   is enabled.
```

Note: As a side effect of the enablement procedure the fips-mode-enable tool also changes the system-wide cryptographic policy level to a level named “FIPS”, this level helps applications by changing configuration defaults to approved algorithms.

**Manual method**

The recommended method automatically performs all the necessary steps.
The following steps can be done manually but are not recommended and are not required if the systems has been installed with the fips-mode-setup tool:

- create a file named /etc/system-fips, the contents of this file are never checked
- ensure to invoke the command ‘`fips-finish-install --complete`’ on the installed system.
- ensure that the kernel boot line is configured with the fips=1 parameter set
- Reboot the system

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

```
$ df |grep boot
```

| | | | | | |
|---|---|---|---|---|---|
| /dev/sda1 | 233191 | 30454 | 190296 | 14% | /boot |

The partition of the `/boot` file system is located on `/dev/sda1` in this example.

Therefore the parameter `boot=/dev/sda1` needs to be appended to the kernel command line in addition to the parameter `fips=1`.

# 9.2 User Guidance

To operate the Module in FIPS Approved mode, the user should use services and security functions listed in Table 7. Any use of non-approved services will put the module in the non-FIPS mode implicitly.

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

## 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-Hellman with keys smaller than 2048 bits must not be used any more. The TLS protocol cannot enforce the support of FIPS Approved Diffie-Hellman key sizes. To ensure full support for all TLS protocol versions, the TLS client implementation of the cryptographic module must accept Diffie-Hellman key sizes smaller than 2048 bits offered by the TLS server.
The TLS server implementation of the cryptographic Module allows 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)
```

Alternatively it is possible to use SSL_CTX_set_dh_auto(ctx, 1); function call that makes OpenSSL to use built-in 2048 bit parameters when the server RSA certificate is at least 2048 bits and 3072 bit DH parameters with RSA certificate of 3072 bits.

To comply with the FIPS 140-2 standard the requirement to not allow Diffie-Hellman key sizes smaller than 2048 bits must be met, to do this the Crypto Officer must ensure that:

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

Using DH parameters and keys smaller than 2048 bits will implicitly place the module into non-FIPS mode, as specified in section 1.2 of the Security Policy.

## 9.2.2 AES-XTS Guidance

The length of a single data unit encrypted or decrypted with the XTS-AES shall not exceed $2^{20}$ AES blocks that is 16MB of data per AES-XTS instance. An XTS instance is defined in section 4 of SP 800-38E.

The AES-XTS mode shall only be used for the cryptographic protection of data on storage devices. The AES-XTS shall not be used for other purposes, such as the encryption of data in transit.

### 9.2.3 AES-GCM IV

In case the module's power is lost and then restored, the key used for the AES GCM encryption or decryption shall be redistributed.

The nonce_explicit part of the IV does not exhaust the maximum number of possible values for a given session key. The design of the TLS protocol in this module implicitly ensures that the nonce_explicit, or counter portion of the IV will not exhaust all of its possible values.

The AES GCM IV generation is in compliance with the [RFC5288] and shall only be used for the TLS protocol version 1.2 to be compliant with [FIPS140-2_IG] IG A.5, provision 1 (“TLS protocol IV generation”); thus, the module is compliant with [SP800-52].

When a GCM IV is used for decryption, the responsibility for the IV generation lies with the party that performs the AES GCM encryption and therefore there is no restriction on the IV generation.

The module supports the TLS GCM ciphersuites from SP800-52 Rev1, section 3.3.1.

### 9.2.4 Triple-DES Keys

According to IG A.13, the same Triple-DES key shall not be used to encrypt more than $2^{16}$ 64-bit blocks of data. It is the user’s responsibility to make sure that the module complies with this requirement and that the module does not exceed this limit.

### 9.2.5 RSA and DSA Keys

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

RSA must be used with either 2048, 3072 or 4096-bit keys because larger key sizes have not been CAVP tested.

DSA must be used with either 2048 or 3072-bit keys because larger key sizes have not been CAVP tested.

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.

### 9.2.6 Handling Self-Test Errors

Any self-test error transitions the module into the error state. The application must be restarted to recover from these errors. Self-test errors include:

- Pairwise consistency test failure: failing a PCT for RSA, DSA or ECDSA
- Integrity test failure: failure to verify the integrity of the module and its shares libraries
- Known-Answer-Test failure: failure to pass a KAT for any algorithm.

These errors are reported through the regular ERR interface of the module 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, the module enters the error state. Any calls to a crypto function of the module returns an error with the error message: 'FATAL FIPS SELFTEST FAILURE' printed to stderr and the Module is terminated with the abort() call.

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

### 9.2.7 Key derivation using SP800-132 PBKDF

The module provides password-based key derivation (PBKDF), compliant with SP800-132. The module supports option 1a from section 5.4 of [SP800-132], in which the Master Key (MK) or a segment of it is used directly as the Data Protection Key (DPK).

In accordance to [SP800-132] and IG D.6, the following requirements shall be met.

Derived keys shall only be used in storage applications. The Master Key (MK) shall not be used for other purposes. The length of the MK or DPK shall be of 112 bits or more.

A portion of the salt, with a length of at least 128 bits, shall be generated randomly using the SP800-90A DRBG,

The iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users. The minimum value shall be 1000.

Passwords or passphrases, used as an input for the PBKDF, shall not be used as cryptographic keys.

The length of the password or passphrase shall be of at least 20 characters, and shall consist of lower-case, upper-case and numeric characters. The probability of guessing the value is estimated to be $1/62^{20} = 10^{-36}$, which is less than $2^{-112}$.

The calling application shall also observe the rest of the requirements and recommendations specified in [SP800-132].

# 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 |
| **HMAC** | Hash Message Authentication Code |
| **MAC** | Message Authentication Code |
| **NIST** | National Institute of Science and Technology |
| **OFB** | Output Feedback |
| **OS** | Operating System |
| **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, https://csrc.nist.gov/projects/cryptographic-module-validation-program/standards
[3] FIPS 140-2 Implementation Guidance, https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Module-Validation-Program/documents/fips140-2/FIPS1402IG.pdf
[4] FIPS 140-2 Derived Test Requirements,https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Module-Validation-Program/documents/fips140-2/FIPS1402DTR.pdf
[5] FIPS 197 Advanced Encryption Standard, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
[6] FIPS 180-4 Secure Hash Standard, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
[7] FIPS 198-1 The Keyed-Hash Message Authentication Code (HMAC), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.198-1.pdf
[8] FIPS 186-4 Digital Signature Standard (DSS), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
[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 2, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
[11] NIST SP 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38b.pdf
[12] NIST SP 800-38C, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf
[13] NIST SP 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf
[14] NIST SP 800-38E, Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf
[15] NIST SP 800-56A Revision 3, Recommendation for Pair-Wise Key Establishment Schemes using Discrete Logarithm Cryptography (Revised), https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf
[16] NIST SP 800-90A Revision 1, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

 