# Red Hat Enterprise Linux 7 Kernel Crypto API Cryptographic Module

# version rhel7.20190718,

# version rhel7.20200812

# and

# version rhel7.20210526

## FIPS 140-2 Non-Proprietary Security Policy

**Version 1.4**

**Last update: 2021-10-04**

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260
Austin, TX 78759
www.atsec.com

# Table of Contents

# Introduction

This document is the non-proprietary Security Policy for the Red Hat Enterprise Linux 7 Kernel Crypto API Cryptographic Module version rhel7.20190718, version rhel7.20200812 and version rhel7.20210526. It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-2 (Federal Information Processing Standards Publication 140-2) for a Security Level 1 module.

# 1 Cryptographic Module Specification

## 1.1 Module Overview

The Red Hat Enterprise Linux 7 Kernel Crypto API Cryptographic Module (hereafter referred to as the “Module”) is a software only cryptographic module that provides general-purpose cryptographic services to the remainder of the Linux kernel. The Red Hat Enterprise Linux 7 Kernel Crypto API Cryptographic Module is software only, security level 1 cryptographic module, running on a multi-chip standalone platform.

The module is implemented as a set of shared libraries / binary files.

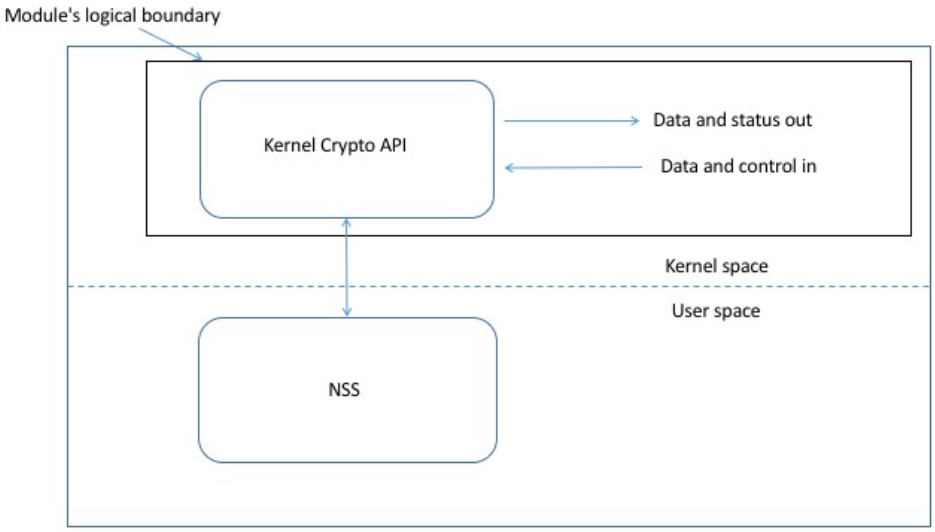


*Figure 1: Cryptographic Module Logical Boundary*

The module is aimed to run on a general purpose computer; the physical boundary is the surface of the case of the target platform, as shown in the diagram below:

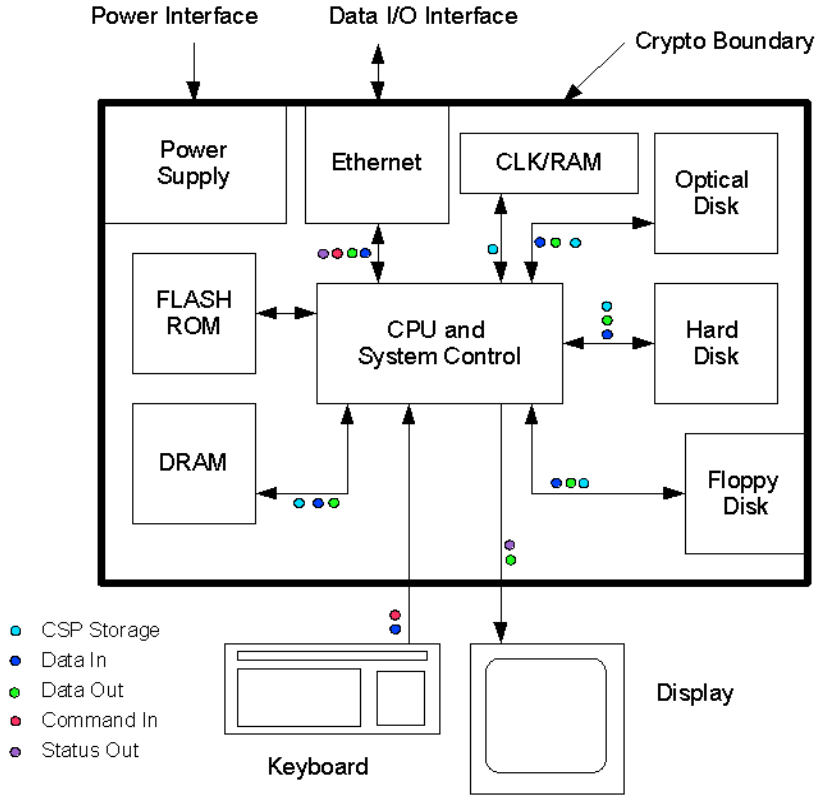


*Figure 2: Cryptographic Module Physical Boundary*

The following list of packages is required for the module to operate:

For Red Hat linux 7 Kernel Crypto API Cryptographic Module version rhel7.20190718 the following files are required:

- the kernel-3.10.0-1062.el7 package, which contains the binary files, integrity check HMAC files and Man Pages for the kernel
- the dracut-fips-033-564.el7 and the dracut-fips-aesni-033-564.el7 packages, which provide the configuration of the FIPS mode

For Red Hat linux 7 Kernel Crypto API Cryptographic Module version rhel7.20200812 the following files are required:

- the kernel-3.10.0-1127.19.1.el7 package, which contains the binary files, integrity check HMAC files and Man Pages for the kernel
- the dracut-fips-033-568.el7 and the dracut-fips-aesni-033-568.el7 packages, which provide the configuration of the FIPS mode

For Red Hat linux 7 Kernel Crypto API Cryptographic Module version rhel7.20210526 the following files are required:

- the kernel-3.10.0-1160.31.el7 package, which contains the binary files, integrity check HMAC files and Man Pages for the kernel
- the dracut-fips-033-572.el7and the dracut-fips-aesni-033-572.el7 packages, which provide the configuration of the FIPS mode

For Red Hat linux 7 Kernel Crypto API Cryptographic Module version rhel7.20190718, version rhel7.20200812 and version rhel7.20210526 the following file is required:

- the hmaccalc-0.9.13-4.el7 package.

The module is made of the following files:

- kernel loadable components /lib/modules/$(uname -r)/kernel/crypto/*.ko
- kernel loadable components /lib/modules/$(uname -r)/kernel/arch/x86/crypto/*.ko
- static kernel binary (vmlinuz): /boot/vmlinuz-$(uname -r)
- static kernel binary (vmlinuz) HMAC file: /boot/.vmlinuz-$(uname -r).hmac
- sha512hmac binary file for performing the integrity checks: usr/bin/sha512hmac
- sha512hmac binary HMAC file: /usr/lib64/hmaccalc/sha512hmac.hmac

The Red Hat linux 7 Kernel Crypto API Cryptographic Module version rhel7.20190718, version rhel7.20200812 and version rhel7.20210526 are bound to the Red Hat Enterprise Linux 7 NSS Cryptographic Module version rhel7.20190606 with FIPS 140-2 Certificate #3860 (hereafter referred to as the “NSS bound module” or “NSS module”) provides the HMAC-SHA-512 algorithm used by the sha512hmac binary file to verify the integrity of both the sha512hmac file and the vmlinuz (static kernel binary) file.

## 1.2 FIPS 140-2 validation

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

| FIPS 140-2 Section | | Security Level |
|---|---|---|
| 1 | Cryptographic Module Specification | 1 |
| 2 | Cryptographic Module Ports and Interfaces | 1 |
| 3 | Roles, Services and Authentication | 1 |
| 4 | Finite State Model | 1 |
| 5 | Physical Security | N/A |
| 6 | Operational Environment | 1 |
| 7 | Cryptographic Key Management | 1 |
| 8 | EMI/EMC | 1 |
| 9 | Self Tests | 1 |
| 10 | Design Assurance | 1 |
| 11 | Mitigation of Other Attacks | N/A |

*Table 1: Security Levels*

The module has been tested on the following platforms with the following configuration:

| Hardware Platform | Processor | Operating System | Tested | |
|---|---|---|---|---|
| | | | With PAA (AES-NI) | Without PAA (AES-NI) |
| Dell PowerEdge R630 | Intel(R) Xeon(R) E5 | Red Hat Enterprise Linux 7 | yes | yes |

*Table 2: Tested Platforms*

The physical boundary is the surface of the case of the target platform. The logical boundary is depicted in Figure 1.

The module also includes algorithm implementations using Processor Algorithm Acceleration (PAA) functions provided by the different processors supported, as shown in the following table:

| Processor | Processor Algorithm Acceleration (PAA) function | Algorithm |
|---|---|---|
| Intel Xeon E5 | AES-NI | AES |

*Table 3: PAA function implementations*

***Note:*** *Per [FIPS 140-2_IG] G.5, the Cryptographic Module Validation Program (CMVP) makes no statement as to the correct operation of the module or the security strengths of the generated keys when this module is ported and executed in an operational environment not listed on the validation certificate.*

## 1.3 Modes of Operations

The module supports two modes of operation: the FIPS approved and non-approved modes.

Section 9.1.1 describes the Secure Installation and Startup to correctly install and configure the module. The module turns to FIPS approved mode after correct initialization, successful completion of power-on self-tests.

Invoking a non-Approved algorithm or a non-Approved key size with an Approved algorithm as listed in Table 7 will result in the module implicitly entering the non-FIPS mode of operation.

The approved services available in FIPS mode can be found in section 3.2, Table 5.

The non-approved services not available in FIPS mode can be found in section 3.2, Table 7.

# 2 Cryptographic Module Ports and Interfaces

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

The logical interfaces are the application program interface (API) through which applications request services. The following table summarizes the four logical interfaces:

| Logical interfaces | Description | Physical ports mapping the logical interfaces |
|---|---|---|
| Command In | API function calls, kernel command line | Keyboard |
| Status Out | API return codes, kernel logs | Display |
| Data In | API input parameters | Keyboard |
| Data Out | API output parameters | Display |
| Power Input | PC Power Port | Physical Power Connector |

*Table 4: Ports and Logical Interfaces*

# 3 Roles, Services and Authentication

## 3.1 Roles

The module supports the following roles:

- **User role**: performs symmetric encryption/decryption, keyed hash, message digest, random number generation, show status
- **Crypto Officer role**: performs the module installation and configuration, module's initialization, self-tests, zeroization and signature verification

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

## 3.2 Services

The module supports services available to users in the available roles. All services are described in detail in the user documentation.

The following table shows the available services, the roles allowed, the Critical Security Parameters involved and how they are accessed in the FIPS mode. 'R' stands for Read permission, 'W' stands for write permission and 'EX' stands for executable permission of the module:

| Service | Algorithms | Note(s) / Mode(s) | CAVS Cert(s). | Role | CSPs | Access |
|---|---|---|---|---|---|---|
| Symmetric encryption/ decryption | Triple-DES | CBC, CTR, ECB | C1404 | User | 192 bits Triple-DES keys | R, EX |
| | AES | CBC, CCM, CTR, ECB, GCM, GMAC, XTS | C1395<br>C1398<br>C1399 | | 128, 192 and 256 bits AES keys<br>Note: XTS mode only with 128 and 256 bits keys | |
| | | CBC, CTR, ECB, GCM | C1396<br>C1397<br>C1401<br>C1402 | | | |
| | | CBC, CTR, ECB, GCM, XTS | C1400 | | | |
| Keyed hash (HMAC) | HMAC SHA-1, HMAC SHA-224, HMAC SHA-256, HMAC SHA-384, HMAC SHA-512 | BS < KS, KS = BS, KS > BS | C1405<br>C1406 | User | at least 112 bits HMAC keys | R, EX |
| | HMAC SHA-1, HMAC SHA-256, HMAC SHA-512 | | C1403<br>C1407 | | | |
| Message digest (SHS) | SHA-1, SHA-224, SHA-256, SHA-384 SHA-512 | N/A | C1405<br>C1406 | User | N/A | N/A |
| | SHA-1, SHA-256, SHA-512 | | C1403<br>C1407 | | | |
| Authenticated | Triple-DES CBC, AES | Encrypt-then-MAC | See AES | User | 128, 192 | R, EX |

| Service | Algorithms | Note(s) / Mode(s) | CAVS Cert(s). | Role | CSPs | Access |
|---|---|---|---|---|---|---|
| encryption | CBC mode and HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512 | cipher (authenc) used for IPsec | and HMAC certs | | and 256 bits AES keys, HMAC keys | |
| Random number generation (SP 800-90A DRBG) | CTR DRBG | With derivation function, with and without prediction resistance function using AES-128, AES-192 and AES-256 | C1395<br>C1398<br>C1399 | User | Entropy input string, V and Key | R, W, EX |
| | Hash DRBG | With derivation function, with and without prediction resistance function using SHA-1, SHA-256 and SHA-512 | C1403<br>C1407 | | Entropy input string, V, C values and Key | |
| | | With derivation function, with and without prediction resistance function using SHA-1, SHA-256, SHA-384 and SHA-512 | C1405<br>C1406 | | | |
| | HMAC DRBG | With and without prediction resistance function using SHA-1, SHA-256 and SHA-512 | C1403<br>C1407 | | Entropy input string, V and Key | |
| | | With and without prediction resistance function using SHA-1, SHA-256, SHA-384 and SHA-512 | C1405<br>C1406 | | | |
| Signature verification | RSA | 2048 and 3072 bits signature verification according to PKCS#1 v1.5, using SHA-1, SHA-256, SHA-512 | C1403<br>C1407 | Crypto Officer | N/A | N/A |
| | | 2048 and 3072 bits signature verification according to PKCS#1 v1.5, using SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 | C1405<br>C1406 | | | |
| Module initialization | N/A | N/A | N/A | Crypto officer | N/A | N/A |
| Self-tests | HMAC-SHA-512,<br>RSA Signature Verification | Integrity test of the kernel static binary performed by the sha512hmac binary provided by the bound NSS module. | NSS HMAC:<br>C1387<br>C1420 | Crypto officer | HMAC-SHA-512 key | R, EX |

| Service | Algorithms | Note(s) / Mode(s) | CAVS Cert(s). | Role | CSPs | Access |
|---|---|---|---|---|---|---|
| | | RSA signature verification performs the signature verification of the kernel loadable components | | | | |
| Show status | N/A | Via verbose mode, exit codes and kernel logs (dmesg) | N/A | User | N/A | N/A |
| Zeroize | N/A | N/A | N/A | Crypto officer | N/A | N/A |
| Installation and configuration | N/A | N/A | N/A | Crypto officer | N/A | N/A |

*Table 5: Available Cryptographic Module's Services in FIPS mode*

The following Key Establishment methods are claimed for this module:

KTS (AES Certs. #C1395, #C1396, #C1397, #C1398, #C1399, #C1400, #C1401 and #C1402, key establishment methodology provides between 128 and 256 bits of encryption strength)

KTS (AES Certs. #C1395, #C1396, #C1397, #C1398, #C1399, #C1400, #C1401 and #C1402 and HMAC Certs. #C1403, #C1405, #C1406 and #C1407, key establishment methodology provides between 128 and 256 bits of encryption strength)

KTS (Triple-DES Certs. #C1404 and HMAC Certs. #C1403, #C1405, #C1406 and #C1407 key wrapping, key establishment methodology provides 112 bits of encryption strength)

In the FIPS Approved mode the Module supports the following non-FIPS Approved but allowed algorithms and/or services, which can be used in the FIPS Approved mode of operation:

| Service | Algorithms | Note(s) / Mode(s) | Role | CSPs | Access |
|---|---|---|---|---|---|
| Non Deterministic Random Number Generation | NDRNG | N/A | User | seed | R |

*Table 6: Non-Approved but Allowed Service Details for the FIPS Approved mode*

In non-Approved mode the Module supports the following non-FIPS Approved algorithms, which shall not be used in the FIPS Approved mode of operation:

| Service | Algorithms | Note(s) / Mode(s) | Role | Keys | Access |
|---|---|---|---|---|---|
| Symmetric encryption/ decryption | AES | XTS with 192-bit keys | User | 192 bits AES keys | R, EX |
| | DES | ECB | | 56 bits DES keys | |
| Message digest | SHA-1 (multiple-buffer implementation) | N/A | User | N/A | N/A |
| Keyed hash | HMAC | Keys smaller than 112 bits | User | HMAC keys with size less than 112 bits | R, EX |
| Random number generation | ansi_cprng | N/A | User | seed | R, W, EX |
| Shared secret computation | Diffie-Hellman | Shared secret computation | User | Diffie-Hellman private keys (1536 bits and larger) | R, W, EX |

| Service | Algorithms | Note(s) / Mode(s) | Role | Keys | Access |
|---|---|---|---|---|---|
| | EC Diffie-Hellman | | | EC Diffie-hellman private keys (P-192 and P-256) | |

*Table 7: Service Details for the non-FIPS mode*

## 3.3 Authentication

The module is a Level 1 software-only cryptographic module and does not implement authentication. The role is implicitly assumed based on the service requested.

# 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 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 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 FIPS Approved 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 Generation

The module employs the Deterministic Random Bit Generator (DRBG) based on [SP800-90A] for the creation of random numbers.

The DRBG is initialized during module initialization. The module loads by default the DRBG using HMAC DRBG with SHA-512, with derivation function, without prediction resistance. The DRBG is seeded during initialization with a seed obtained from get_random_bytes( ) of length 3/2 times the DRBG strength. The NDRNG is provided by the Linux RNG and is located within the module's physical boundary but outside its logical boundary. The NDRNG provides 128 bits of entropy.

The module implements the health checks defined by SP 800-90A, section 11.3. The noise source  implements continuous tests for the NDRNG.

Here are listed the CSPs/keys details concerning storage, input, output, generation and zeroization:

| Type | Keys/CSPs | CSP Size/Modes | Key Generation | Key Storage | Key Entry/Output | Key Zeroization |
|---|---|---|---|---|---|---|
| Symmetric keys | AES | CBC, CCM, CTR, ECB, GCM, GMAC, XTS<br><br>128, 192, 256 bit keys.<br><br>Note: XTS mode only with 128 and 256 bit keys. | N/A | Protected kernel memory | API allows caller on the same GPC to supply key | Memory is automatically overwritten by zeroes when freeing the cipher handler |
| | Triple-DES | CBC, CTR, ECB<br><br>192 bits Triple-DES keys | N/A | Protected kernel memory | API allows caller on the same GPC to supply key | Memory is automatically overwritten by zeroes when freeing the cipher handler |
| DRBG SP800-90A entropy string | SP 800-90A DRBG seed and Entropy string for HMAC and CTR DRBG | According to SP 800-90A | The seed data obtained from get_random_bytes () | Module’s application memory | N/A | Automatic zeroization when seeding operation completes |
| | SP 800-90A DRBG seed and Entropy string for HASH DRBG | | | | | |
| SP 800- | SP 800- | According | Based on | Protected | N/A | Memory is automatically |

| | | | | | | |
|---|---|---|---|---|---|---|
| 90A DRBG internal state | 90A DRBG Seed and internal state values V, and K for HMAC and CTR DRBG<br>SP 800-90A DRBG internal state values V, and C for Hash DRBG | to SP 800-90A | entropy string as defined in SP 800-90A | kernel memory | | overwritten by zeroes when freeing the cipher handler |
| HMAC keys | HMAC keys | BS < KS, KS = BS, KS > BS<br><br>At least 112 bits HMAC keys | N/A | Protected kernel memory | HMAC key can be supplied by calling application | Memory is automatically overwritten by zeroes when freeing the cipher handler |

*Table 8: Keys/CSPs*

As defined in SP800-90A, the DRBG obtains the entropy string and nonce from the Linux kernel non-deterministic random number generator during:

a. initialization of a DRBG instance

b. after $2^{48}$ requests for random numbers

The module does not provide any key generation service or perform key generation for any of its Approved algorithms. Keys are passed in from calling application via API parameters.

*Caveat: The module generates random strings whose strengths are modified by available entropy.*

## 6.2 Key establishment / Key Transport

The module provides SP 800-38F compliant key wrapping using AES with GCM and CCM block chaining modes, as well as a combination of AES-CBC for encryption/decryption and HMAC for authentication. The module also provides SP 800-38F compliant key wrapping using a combination of Triple-DES-CBC for encryption/decryption and HMAC for authentication.

According to “Table 2: Comparable strengths” in [SP 800-57], the key sizes of AES provides the following security strength in FIPS mode of operation:

- AES: key wrapping provides between 128 and 256 bits of encryption strength.
- Triple-DES: key wrapping provides 112 bits of encryption strength.

## 6.3 Key / Critical Security Parameter (CSP) Access

An authorized application as user (the User role) has access to all key data generated during the operation of the module. Moreover, the module does not support the output of intermediate key generation values during the key generation process.

## 6.4 Key / CSP Storage

Symmetric 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 keys. The RSA public key used for signature verification of the kernel loadable components is stored outside of the module's boundary, in a keyring file in /proc/keys/.

## 6.5 Key / CSP Zeroization

When a calling kernel components calls the appropriate API function that operation overwrites memory with 0s and then frees that memory (please see the API document for full details).

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.

# 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

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 perform self-tests to ensure the integrity of the Module and the correctness of the cryptographic functionality at start up. In addition, the module performs conditional test for DRBG.

A failure of any of the self-tests panics the Module. The only recovery is to reboot. For persistent failures, you must reinstall the kernel. See section 9.1 for details.

No operator intervention is required during the running of the self-tests.

## 8.1 Power-Up Self-Tests

The module performs power-up self-tests at module initialization to ensure that the module is not corrupted and that the cryptographic algorithms work as expected. The self-tests are performed without any user intervention.

While the module is performing the power-up tests, services are not available and input or output is not possible: the module is single-threaded and will not return to the calling application until the self-tests are completed successfully.

### 8.1.1 Integrity Tests

The Module performs power-up self tests (at module initialization). Input, output, and cryptographic functions cannot be performed while the Module is in a self test or error state. The Module is single-threaded during the self tests and will stop the boot procedure, and therefore any subsequent operation before any other kernel component can request services from the Module.

The Crypto Officer with physical or logical access to the Module can run the POST (Power-On Self-Tests) on demand by power cycling the Module or by rebooting the operating system.

For Known Answer Test, HMAC SHA-512 provided by the NSS bound module is tested before the NSS module makes itself available to the sha512hmac application. In addition, if the Intel AES-NI support is present and the dracut-fips aesni RPM package (see section 1) is installed, the AES-NI implementation is self-tested with the same KAT vector as the other AES implementations.

An HMAC SHA-512 (provided by the NSS bound module) calculation is performed on the sha512hmac utility and static Linux kernel binary to verify their integrity. The Linux kernel crypto API kernel components, and any additional code components loaded into the Linux kernel are checked with the RSA signature verification implementation of the Linux kernel when loading them into the kernel to confirm their integrity.

NOTE: The fact that the kernel integrity check passed, which requires the loading of sha512hmac with the self tests implies a successful execution of the integrity and self tests of sha512hmac (the HMAC is stored in /usr/lib/hmaccalc/sha512hmac.hmac).

With respect to the integrity check of kernel loadable components providing the cryptographic functionality, the fact that the self test of these cryptographic components are displayed implies that the integrity checks of each kernel component passed successfully.

The table below summarizes the power-on self tests performed by the module, which includes the Integrity Test of the module itself as stated above and the Known Answer Test for each approved cryptographic algorithm.

| Algorithm | Test |
|---|---|
| AES (ECB, CBC, XTS, CTR, OFB, GCM, CCM) | KAT, encryption and decryption are tested separately |
| Triple-DES (CBC, CTR, ECB) | KAT, encryption and decryption are tested separately |
| RSA signature verification | Part of the integrity test (considered as a KAT) |

| | |
|---|---|
| DRBG (CTR, Hash, HMAC) | KAT |
| HMAC SHA-1, -224, -256, -384, -512 | KAT |
| SHA-1, -224, -256, -384, -512 | KAT |
| Integrity check | HMAC SHA-512 |

*Table 9: Module Self-Tests*

## 8.2 Conditional Tests

The module does not perform any conditional tests.

# 9 Guidance

## 9.1 Cryptographic Officer Guidance

To operate the Kernel Crypto API 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 runlevel 1 on RHEL. This refers to processes having access to the same cryptographic instance which RHEL ensures cannot happen by the memory management hardware.)

### 9.1.1 Secure Installation and Startup

Crypto Officers use the Installation instructions to install the Module in their environment.

The version of the RPM containing the FIPS validated module is stated in section 1.1 above. The integrity of the RPM is automatically verified during the installation and the Crypto Officer shall not install the RPM file if the RPM tool indicates an integrity error.

To bring the Module into FIPS approved mode, perform the following:

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
```

### 9.1.2 FIPS 140-2 and AES NI Support

According to the Kernel Crypto API FIPS 140-2 Security Policy, the Kernel Crypto API module supports the AES-NI Intel processor instruction set as an approved cipher. The AES-NI instruction set is used by the Module.

In case you configured a full disk encryption using AES, you *may* use the AES-NI support for a higher performance compared to the software-only implementation.

To utilize the AES-NI support, the mentioned Module must be loaded during boot time by installing a plugin.

Before you install the plugin, you MUST verify that your processor offers the AES-NI instruction set by calling the following command:

```
cat /proc/cpuinfo | grep aes
```

If the command returns a list of properties, including the “aes” string, your CPU provides the AES-NI instruction set. If the command returns nothing, AES-NI is not supported.

You MUST NOT install the following plugin if your CPU does not support AES-NI because the kernel will panic during boot.

The support for the AES-NI instruction set during boot time is enabled by installing the following plugin (make sure that the version of the plugin RPM matches the version of the installed RPMs!):

```
# install the dracut-fips-aesni package
yum install dracut-fips-aesni-*.noarch.rpm
# recreate the initramfs image
dracut -f
```

The changes come into effect during the next reboot.

# 9.2 User Guidance

CTR and RFC3686 mode must only be used for IPsec. It must not be used otherwise.

There are three implementations of AES: aes-generic, aesni-intel, and aes-x86_64 on x86_64 machines. The additional specific implementations of AES for the x86 architecture are disallowed and not available on the test platforms.

When using the Module, the user shall utilize the Linux Kernel Crypto API provided memory allocation mechanisms. In addition, the user shall not use the function copy_to_user() on any portion of the data structures used to communicate with the Linux Kernel Crypto API.

Only the cryptographic mechanisms provided with the Linux Kernel Crypto API are considered for use. The NSS bound module, although used, is only considered to support the integrity verification and is not intended for general-purpose use with respect to this Module.

## 9.2.1 XTS Usage

The XTS mode must only be used for the disk encryption functionality offered by dm-crypt.

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.

The XTS key check that key1 != key2 is done within the module, which is compliant with IG A.9

## 9.2.2 GCM Usage

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 module generates the IV internally randomly with an approved SP 800-90B DRBG, which is compliant with provision 2) of IG A.5.

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

## 9.2.3 Triple-DES Usage

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.

# 9.3 Handling Self Test Errors

Self test failure within the Kernel Crypto API module or the dm-crypt kernel component will panic the kernel and the operating system will not load.

Recover from this error by trying to reboot the system. If the failure continues, you must reinstall the software package being sure to follow all instructions. If you downloaded the software verify the package hash to confirm a proper download. Contact Red Hat if these steps do not resolve the problem.

The Kernel Crypto API module performs a power-on self test that includes an integrity check and known answer tests for the available cryptographic algorithms.

The kernel dumps self test success and failure messages into the kernel message ring buffer. Post boot, the messages are moved to /var/log/messages.

Use **dmesg** to read the contents of the kernel ring buffer. The format of the ringbuffer (**dmesg**) output is:

```
alg: self-tests for %s (%s) passed
```

Typical messages are similar to "alg: self-tests for xts(aes) (xts(aes-x86_64)) passed" for each algorithm/sub-algorithm type.

# Appendix A Glossary and Abbreviations

| | |
|---|---|
| AES | Advanced Encryption Standard |
| AES-NI | Advanced Encryption Standard New Instructions |
| CAVP | Cryptographic Algorithm Validation Program |
| CBC | Cipher Block Chaining |
| CCM | Counter with Cipher Block Chaining Message Authentication Code |
| CFB | Cipher Feedback |
| CMAC | Cipher-based Message Authentication Code |
| CMVP | Cryptographic Module Validation Program |
| CSP | Critical Security Parameter |
| CTR | Counter Mode |
| DES | Data Encryption Standard |
| DSA | Digital Signature Algorithm |
| DRBG | Deterministic Random Bit Generator |
| ECB | Electronic Code Book |
| ECC | Elliptic Curve Cryptography |
| FFC | Finite Field Cryptography |
| FIPS | Federal Information Processing Standards Publication |
| FSM | Finite State Model |
| GCM | Galois Counter Mode |
| HMAC | Hash Message Authentication Code |
| KAS | Key Agreement Schema |
| KAT | Known Answer Test |
| MAC | Message Authentication Code |
| NDF | No Derivation Function |
| NIST | National Institute of Science and Technology |
| NDRNG | Non-Deterministic Random Number Generator |
| OFB | Output Feedback |
| O/S | Operating System |
| PAA | Processor Algorithm Acceleration |
| PR | Prediction Resistance |
| PSS | Probabilistic Signature Scheme |
| RNG | Random Number Generator |
| RSA | Rivest, Shamir, Addleman |

| | |
|---|---|
| SHA | Secure Hash Algorithm |
| SHS | Secure Hash Standard |
| TDES | Triple DES |
| XTS | XEX-based Tweaked-codebook mode with ciphertext Stealing |

# Appendix B References


**FIPS180-4** **Secure Hash Standard (SHS)**
August 2015
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

**FIPS186-4** **Digital Signature Standard (DSS)**
July 2013
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

**FIPS197** **Advanced Encryption Standard**
November 2001
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

**FIPS198-1** **The Keyed Hash Message Authentication Code (HMAC)**
July 2008
http://csrc.nist.gov/publications/fips/fips198 1/FIPS-198 1_final.pdf

**RFC3394** **Advanced Encryption Standard (AES) Key Wrap Algorithm**
September 2002
http://www.ietf.org/rfc/rfc3394.txt

**RFC5649** **Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm**
September 2009
http://www.ietf.org/rfc/rfc5649.txt

**SP800-38A** **NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques**
December 2001
http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

**SP800-38B** **NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication**
May 2005
http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

**SP800-38C** **NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality**
May 2004
http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated July20_2007.pdf

**SP800-38D** **NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC**
November 2007
http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf

**SP800-38E** **NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices**
January 2010
http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf

**SP800-38F** **NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping**
December 2012
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf

| | |
|---|---|
| **SP800-56A Rev. 3** | **NIST Special Publication 800-56A Revision 3 - Recommendation for Pair Wise Key Establishment Schemes Using Discrete Logarithm Cryptography**<br>May 2013<br>https://csrc.nist.gov/publications/detail/sp/800-56a/rev-3/final |
| **SP800-56C Rev. 1** | **NIST Special Publication 800-67 Revision 1 - Recommendation for Key Derivation through Extraction-then-Expansion**<br>November 2011<br>https://csrc.nist.gov/publications/detail/sp/800-56c/rev-1/final |
| **SP800-67 Rev. 2** | **NIST Special Publication 800-67 Revision 2 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher**<br>November 2017<br>https://csrc.nist.gov/publications/detail/sp/800-67/rev-2/final |
| **SP800-90A Rev. 1** | **NIST Special Publication 800-90A Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators**<br>June 2015<br>https://csrc.nist.gov/publications/detail/sp/800-90a/rev-1/final |
| **SP800-90B** | **NIST Draft Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation**<br>January 2018<br>https://csrc.nist.gov/publications/detail/sp/800-90b/final |
| **SP800-108** | **NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions**<br>October 2009<br>https://csrc.nist.gov/publications/detail/sp/800-108/final |
| **SP800-131A** | **NIST Special Publication 800-131A - Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths**<br>March 2019<br>https://csrc.nist.gov/publications/detail/sp/800-131a/rev-2/final |