Red Hat

Red Hat Enterprise Linux 8 Kernel Crypto API Cryptographic Module

version rhel8.20190926

FIPS 140-2 Non-Proprietary Security Policy

Version 1.3

Last update: 2021-01-08

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 8 Kernel Crypto API Cryptographic Module version rhel8.20190926. 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 8 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 8 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.

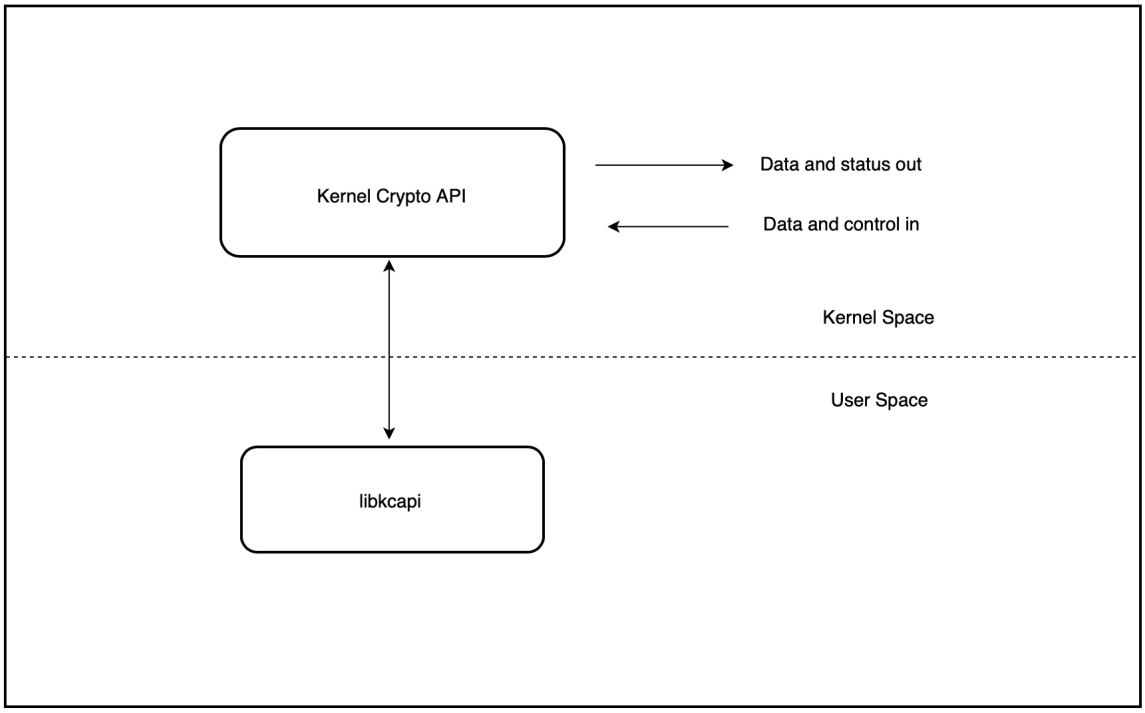

Figure 1: Cryptographic Module Logical Boundary

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:

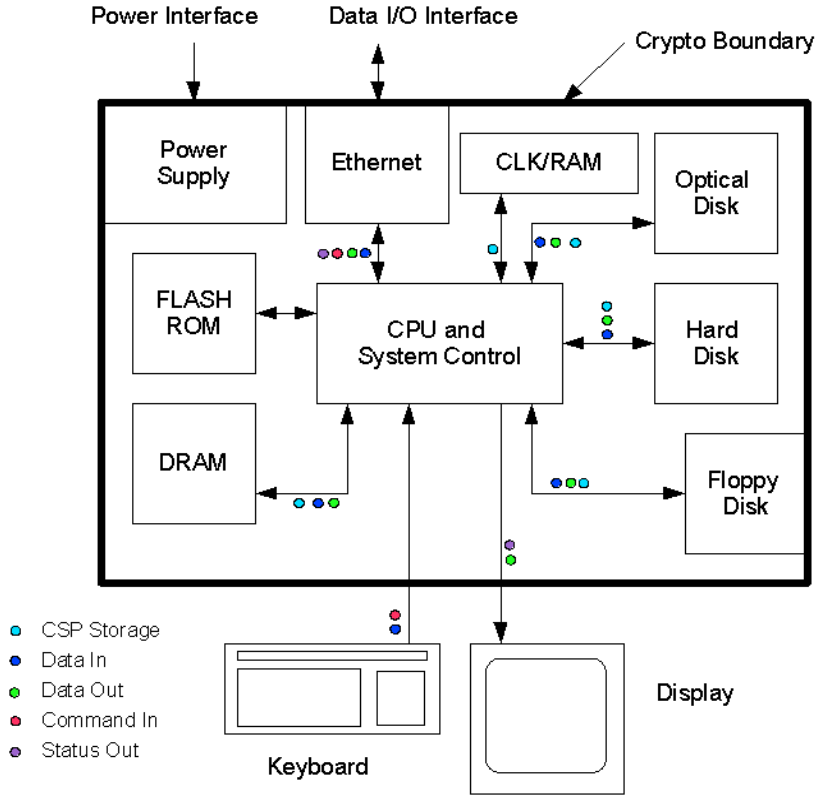

Figure 2: Cryptographic Module Physical Boundary

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

- the kernel-4.18.0-147.el8 package, which contains the binary files, integrity check HMAC files and Man Pages for the kernel
- the libkcapi-hmaccalc-1.1.1-16_1.el8.x86_64 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 kernel 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 R430	Intel(R) Xeon(R) E5	Red Hat Enterprise Linux 8	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

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 critical security parameters (CSPs) used or stored in approved mode are not used in non-approved mode and vice versa.

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 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,	A142 A180 A183	User	168 bits Triple-DES keys	R, EX
		CMAC	A180 A183			
		CFB64	A127 A181			
	AES	CBC, CCM, CMAC, CTR, ECB, GCM (external IV, decryption only), GMAC, XTS	A141 A180 A183		128, 192 and 256 bits AES keys Note: XTS mode only with 128 and 256 bits keys	
		ECB	A128 A178			
		ECB, GCM (internal IV, encryption/decryption)	A123 A140 A171 A174			
		ECB, GCM (external IV, decryption only)	A124 A126 A139 A179			

Service	Algorithms	Note(s) / Mode(s)	CAVS Cert(s).	Role	CSPs	Access
		CFB128	A125 A127 A181			
		CBC, CTR, ECB, GCM (external IV, decryption only), XTS	A172			
Keyed hash (HMAC)	HMAC SHA-1, HMAC SHA-224, HMAC SHA-256, HMAC SHA-384, HMAC SHA-512	BS < KS, KS = BS, KS > BS	A128 A134 A176 A177 A178 A180	User	at least 112 bits HMAC keys	R, EX
	HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384 HMAC-SHA3-512		A182			
Message digest (SHS)	SHA-1, SHA-224, SHA-256, SHA-384 SHA-512	N/A	A128 A134 A176 A177 A178 A180	User	N/A	N/A
	SHA3-224, SHA3-256, SHA3-384 SHA3-512		A182			
Authenticated encryption (KTS)	AES-CBC and HMAC-SHA-1, SHA-224 SHA-256, SHA-384 and SHA-512	CBC and HMAC used with encrypt-then-MAC cipher (authenc) used for IPsec	See AES, Triple-DES and HMAC certs	User	128, 192 and 256 bits AES keys, HMAC keys 168 bits Triple-DES keys	R, EX
	Triple-DES-CBC and HMAC-SHA-1, SHA-224 SHA-256, SHA-384 and SHA-512					
	AES	GCM and CCM				
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	A123 A124 A126 A128 A139 A140 A141 A171 A172	User	Entropy input string, seed, V, C values and Key (K)	R, W, EX

Service	Algorithms	Note(s) / Mode(s)	CAVS Cert(s).	Role	CSPs	Access
			A174 A178 A179 A180 A183			
	Hash DRBG	With derivation function, with and without prediction resistance function using SHA-1, SHA-256, SHA_384 and SHA-512	A123 A124 A126 A128 A134			
	HMAC DRBG	With and without prediction resistance function using SHA-1, SHA-256, SHA-384 and SHA-512	A139 A140 A141 A142 A171 A172 A174 A176 A177 A178 A179 A180 A183			
Signature verification	RSA	2048, 3072 and 4096 bits signature verification according to PKCS#1 v1.5, using SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	A134 A176 A177 A180	User	N/A	N/A
Signature Generation	RSA	According to PKC#1 v1.5 SHA-224, SHA-256, SHA-384, SHA-512	A134 A176 A177 A180	User	2048, 3072, 4096-bit RSA private key	R, EX
Component Public Key Validation and Shared Secret Computation	ECDH	P-256 with SHA-256, SHA-384, SHA-512	CVL A178	User	P-256-based ECDH private key Shared Secret	R, W, EX
	DH	N/A SHA-224, SHA-256	CVL A128	User	2048-bit DH private key Shared secret	R, W, EX

Service	Algorithms	Note(s) / Mode(s)	CAVS Cert(s).	Role	CSPs	Access
Module initialization	N/A	N/A	N/A	Crypto officer	N/A	N/A
Self-tests	HMAC-SHA-512, RSA Signature Verification	Integrity test of the kernel static binary performed by the sha512hmac binary RSA signature verification performs the signature verification of the kernel loadable components		Crypto officer	N/A	R, EX
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	All CSPs	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 module claims SP 800-38F compliant key wrapping with the following modes (using any available implementation specified in Table 5):

- AES-GCM
- AES-CCM
- AES-CBC with HMAC-SHA1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384 and HMAC-SHA-512
- Triple-DES-CBC with HMAC-SHA1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384 and HMAC-SHA-512.

Therefore, the following caveats apply:

KTS (AES Certs. #A123, #A124, #A126, #A139, #A140, #A141, #A171, #A172, #A174, #A179, #A180 and #A183; key establishment methodology provides between 128 and 256 bits of encryption strength)

KTS (AES Certs. #A141, #A172, #A180 and #A183 and HMAC Certs. #A128, #A134, #A176, #A177, #A178 and #A180; key establishment methodology provides between 128 and 256 bits of encryption strength)

KTS (Triple-DES Certs. #A142, #A180 and #A183 and HMAC Certs. #A128, #A134, #A176, #A177, #A178 and #A180; 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	N/A	N/A

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	Key(s)	Access
Symmetric encryption/ decryption	AES	XTS with 192-bit keys	User	192 bits AES keys	R, EX
		GCM encryption with external IV		128, 192, 256-bit AES keys	
	DES	ECB		56 bits DES keys	
Message digest	SHA-1 (multiple-buffer implementation)	N/A	User	N/A	R, EX
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
Signature Generation	RSA	Using SHA-1	User	RSA private key	R, EX
Key generation	ECDSA	P-192, P-256 The PCT is not implemented.	User	ECDSA private key	R, W, EX
Shared secret computation	Diffie-Hellman	Shared secret computation	User	Diffie-Hellman private keys (smaller than 2048 bits and keys larger than 2048 bits)	R, W, EX
	EC Diffie-Hellman			EC Diffie-hellman private keys (P-192)	

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 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 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 a SP 800-90A DRBG as a random number generator for the creation of random numbers. In addition, the module provides a Random Number Generation service to applications.

The DRBG supports the Hash_DRBG, HMAC_DRBG and CTR_DRBG mechanisms. The module uses a Non-Deterministic Random Number Generator (NDRNG) as the entropy source. The NDRNG is based on the Linux RNG and the CPU time jitter RNG, both within the module's logical boundary. The NDRNG provides at least 128 bits of entropy to the DRBG during initialization (seed) and reseeding (reseed).

“The module generates random strings whose strengths are modified by available entropy”

The module performs conditional self-tests on the output of NDRNG to ensure that consecutive random numbers do not repeat, and performs DRBG health tests as defined in section 11.3 of [SP800-90A].

6.2 Key Generation

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

Type	Keys/CSPs	Key Generation	Key Storage	Key Entry/Output	Key Zeroization
Symmetric keys	AES	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	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 Entropy string	The seed data obtained from getrandom()	Module’s application memory	N/A	Automatic zeroization when seeding operation completes
SP 800-90A DRBG C, V and K values (internal state)	SP 800-90A DRBG seed and internal state values V, C and K	Based on entropy string as defined in SP 800-90A	Protected kernel memory	N/A	Memory is automatically overwritten by zeroes when freeing the cipher handler
HMAC keys	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
Diffie-Hellman Private Components	shared secret	Computed using SP 800-56A	Protected kernel memory	secret is passed out of module via API output parameters	Memory is automatically overwritten by zeroes when

	DH private key	N/A		key is passed out of module via API output parameters	freeing the cipher handler
EC Diffie-Hellman Private Components	shared secret	Computed using SP 800-56A	Protected kernel memory	secret is passed out of module via API output parameters	Memory is automatically overwritten by zeroes when freeing the cipher handler
	ECDH private key	N/A		key is passed out of module via API output parameters	
RSA private key	Private key	N/A	Protected kernel memory	key is passed in to module via API input parameters	Memory is automatically overwritten by zeroes when freeing the cipher handler
RSA public key	public key	N/A	Protected kernel memory	key is passed in to module via API input parameters	Memory is automatically overwritten by zeroes when freeing the cipher handler
Diffie-Hellman public key	public key	N/A	Protected kernel memory	key is passed in to module via API input parameters	Memory is automatically overwritten by zeroes when freeing the cipher handler
EC Diffie-Hellman public key	public key	N/A	Protected kernel memory	key is passed in to module via API input parameters	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.

6.3 Key establishment / Key Transport

The module supports Diffie-Hellman and EC Diffie-Hellman shared secret primitive computation:

- Diffie-Hellman: shared secret computation provides 112 bits of encryption strength.
- EC Diffie-Hellman: shared secret computation provides 128 bits of encryption strength.

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.

6.4 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.

6.5 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/. The Diffie-Hellman and EC Diffie-Hellman public keys are stored in protected kernel memory.

6.6 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.

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).

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

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.

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.

An HMAC SHA-512 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	KAT, encryption and decryption are tested separately
Triple-DES	KAT, encryption and decryption are tested separately
RSA signature generation	KAT
RSA signature verification	KAT (also tested as part of the integrity test)
DRBG (CTR, Hash, HMAC)	KAT
DRBG	Health test per section 11.3 of SP 800-90A DRBG

HMAC SHA-1, -224, -256, -384, -512	KAT
HMAC SHA3-224, -256, -384, -512	KAT
SHA-1, -224, -256, -384, -512	KAT
SHA3-224, -256, -384, -512	KAT
Diffie-Hellman Z primitive with 2048 bits	KAT
EC Diffie-Hellman Z primitive with P-256	KAT
Integrity check	HMAC SHA-512

Table 9: Module Self-Tests

8.2 Conditional Tests

The module performs the CRGNT on the NDRNG.

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.

9.1.2 FIPS module installation instructions:

9.1.2.1 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.

9.1.2.2 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

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.

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.

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
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
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard

XTS	XEX-based Tweaked-codebook mode with ciphertext Stealing

Appendix B References

FIPS180-4 **Secure Hash Standard (SHS)**
August 2015
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

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

FIPS197 **Advanced Encryption Standard**
November 2001
ttps://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

FIPS198-1 **The Keyed Hash Message Authentication Code (HMAC)**
July 2008
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.198-1.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
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf

SP800-38B **NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication**
May 2005
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.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**
July 2007
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf

SP800-38D **NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC**
November 2007
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.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
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf

SP800-38F **NIST Special Publication 800-38F - Recommendation for Block**

Cipher Modes of Operation: Methods for Key Wrapping
December 2012
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf

SP800-56A **NIST Special Publication 800-56A Revision 3 - Recommendation for Pair Wise Key Establishment Schemes Using Discrete Logarithm Cryptography**
April 2018
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf

SP800-56C **NIST Special Publication 800-56A Revision 1 - Recommendation for Key Derivation in Key-Establishment Schemes**
April 2018
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr1.pdf

SP800-67 **NIST Special Publication 800-67 Revision 2 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher**
November 2017
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-67r2.pdf

SP800-90A **NIST Special Publication 800-90A Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators**
June 2015
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

SP800-90B **NIST Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation**
January 2018
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf

SP800-108 **NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions**
October 2009
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-108.pdf

SP800-131A **NIST Special Publication 800-131A Revision 2 - Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths**
March 2019
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf