Red Hat Enterprise Linux 6.6 Kernel Crypto API Cryptographic Module v3.0

FIPS 140-2 Security Policy

Version 1.4

Last Update: 2016-02-23

Table of Contents

1. Cryptographic Module Specification

This document is the non-proprietary security policy for the Red Hat Enterprise Linux 6.6 Kernel Crypto API Cryptographic Module, and was prepared as part of the requirements for conformance to Federal Information Processing Standard (FIPS) 140-2, Level 1.

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

1.1. Description of the Module

The Red Hat Enterprise Linux 6.6 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 6.6 Kernel Crypto API Cryptographic Module is a software only, security level 1 cryptographic module, running on a multi-chip standalone platform.

The following table shows the overview of 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	N/A

Table 1: Security Level of the Module

The Module has been tested in the following configurations on all hardware systems listed below:

- 64-bit x86_64 without AES-NI support
- 64-bit x86_64 with AES-NI support

The Module has been tested on the following multi-chip standalone platforms:

Manufacturer	Model	O/S & Ver.
HP (Hewlett-Packard)	ProLiant DL380p Gen8	Red Hat Enterprise Linux 6.6 (Single User Mode)
IBM (International Business Machines)	System x3500 M4	Red Hat Enterprise Linux 6.6 (Single User Mode)

Table 2: Tested Platforms

The Module provides main services with the Linux kernel. To perform integrity verification, additional software is used as outlined in the following list.

The list of components and the cryptographic boundary of the composite module is defined as follows:

- The Linux Kernel with the version of the RPM file of 2.6.32-504.23.1.
- The module integrity check is performed by the Red Hat Enterprise Linux utility sha512hmac which uses the NSS library to obtain the cryptographic mechanisms. The name and version of the RPM file providing this utility is hmaccalc-0.9.12-2.el6.
- Network Security Service (NSS), a separate cryptographic module (Red Hat Enterprise Linux 6.6 NSS Module with FIPS certificate #2564) provides cryptographic algorithms and security functions for sha512hmac application, which performs the integrity validation of the Linux kernel binary. The sha512hmac application uses the NSS module in accordance with the NSS Security Policy. The CAVS certificates for NSS are listed in Table 5: NSS algorithms and CAVS certificates used for the integrity check.
- The configuration of the FIPS mode is provided by the dracut-fips package with the version of the RPM file of 004-356.el6_6.1. This RPM version is provided with accessibility to the Red Hat Network.

The module comprises the following files:

- kernel modules /lib/modules/$(uname -r)/kernel/crypto/*.ko
- kernel modules /lib/modules/$(uname -r)/kernel/arch/x86/crypto/*.ko
- static kernel binary /boot/vmlinuz-$(uname -r)
- NSS softoken FIPS 140-2 module providing the HMAC-SHA512 algorithm used by the sha512hmac binary file to verify the integrity of both the sha512hmac file and the vmlinuz (static kernel binary)
- sha512hmac binary file for performing the integrity checks

1.2. Description of the Approved Modes

The Module must always be configured as outlined in section 9. The Module can be operated in FIPS approved mode or non-FIPS approved mode. The Module's FIPS-mode is enforced by policy, I.e using FIPS approved algorithm implicitly puts the module into FIPS mode, if the configuration flag `/proc/sys/crypto/fips_enabled` equals 1. Only FIPS approved algorithms are available in FIPS mode. Using a non-FIPS approved algorithm results in putting the Module into non-FIPS mode.

The Module supports the following FIPS 140-2 approved algorithms:

Algorithm	Validation Certificate	Usage	Keys/CSPs
AES • ECB • CBC • CCM	Certs. #3145, #3146, #3148, #3149, #3151, #3152	encrypt/decrypt	AES keys 128, 192, 256 bits (stored in ephemeral memory and zeroized with memset)

Algorithm	Validation Certificate	Usage	Keys/CSPs
• GCM • CTR			
AES • ECB • CBC • GCM • CTR	Certs. #3147, #3150	encrypt/decrypt	AES keys 128, 192, 256 bits (stored in ephemeral memory and zeroized with memset)
AES • GCM	Certs. #3218, #3219	encrypt/decrypt	AES keys 128, 192, 256 bits (stored in ephemeral memory and zeroized with memset)
AES • XTS	Certs. #3145, #3146, #3147, #3148, #3149, #3150, #3151, #3152	encrypt/decrypt	AES keys 128, 256 bits (stored in ephemeral memory and zeroized with memset)
Triple-DES • ECB • CBC	Certs. #1797, #1798	encrypt/decrypt	168 bit Triple DES keys (stored in ephemeral memory and zeroized with memset)
FIPS 186-4 DSA2	Certs. #909, #910	signature verification	DSA 2048 bit public key (stored on non-volatile disk and is not zeroized by the module while not used; stored in ephemeral memory while processed and zeroized with memset)
SP 800-90A CTR_DRBG • AES-128 • AES-192 • AES-256	Certs. #639, #640, #641, #642, #643, #644, #645, #646	random number generation	Entropy input string, V and Key (stored in ephemeral memory and zeroized with memset)
SP 800-90A Hash_DRBG • SHA-1 • SHA-256 • SHA-384 • SHA-512	Certs. #645, #646	random number generation	Entropy input string, V and C (stored in ephemeral memory and zeroized with memset)
SP 800-90A HMAC_DRBG • SHA-1 • SHA-256 • SHA-384 • SHA-512	Certs. #645, #646	random number generation	Entropy input string, V and Key (stored in ephemeral memory and zeroized with memset)
C implementation of SHA-1 SHA-224 SHA-256	Certs. #2607, #2608	hashing	N/A

Algorithm	Validation Certificate	Usage	Keys/CSPs
SHA-384 SHA-512			
C implementation of HMAC-SHA1 HMAC-SHA224 HMAC-SHA256 HMAC-SHA384 HMAC-SHA512	Certs. #1985, #1986	message integrity	HMAC key (stored in ephemeral memory and zeroized with memset)

Table 3: Approved Algorithms

Caveats:

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

The Module supports the following FIPS 140-2 non-approved algorithms:

Algorithm	Usage	Keys/CSPs
DES	Encryption/decryption	DES key
Triple DES • CTR	Encryption/decryption	Triple DES key
AES • XTS	Encryption/decryption	192 bit AES key
SSSE3, AVX and AVX2 implementation of SHA-256 SHA-512	hashing	N/A
SSSE3, AVX and AVX2 implementation of HMAC SHA-256 HMAC SHA-512	Keyed hash	HMAC key

Table 4: Non-approved Algorithms

The Module provides multiple implementations of AES given in the following bullet list. All these implementations and the related algorithms have been CAVS tested.

- AES using AES-NI Intel instruction set when the aesni-intel kernel module is loaded (the kernel module can only be used if the underlying processor provides the AES-NI instruction set)
- AES using AES-NI Intel instruction set when the aesni-intel kernel module is loaded (the kernel module can only be used if the underlying processor provides the AES-NI instruction set) with C-version of block chaining modes
- AES implemented with assembler code when the aes-x86_64 kernel module is loaded
- AES implemented with C code when the aes-generic kernel module is loaded

Note, if more than one of the above listed kernel Modules are loaded, the respective implementation can be requested by using the following cipher mechanism strings with the initialization calls (such as crypto_alloc_blkcipher):

- aesni-intel kernel module: "__aes-aesni"
- aes-x86_64 kernel module: "aes-asm"
- aes-generic kernel module: "aes-generic"

The AES cipher can also be loaded by simply using the string "aes" with the initialization call. In this case, the AES implementation whose kernel module is loaded with the highest priority is used. The following priority exist:

1. aesni-intel
2. aes-x86_64
3. aes-generic

For example: If the kernel modules aesni-intel and aes-asm are loaded and the caller uses the initialization call (such as crypto_alloc_blkcipher) with the cipher string of "aes", the aesni-intel implementation is used. On the other hand, if only the kernel modules of aes-x86_64 and aes-generic are loaded, the cipher string of "aes" implies that the aes-x86_64 implementation is used.

The discussion about the naming and priorities of the AES implementation also applies when cipher strings are used that include the block chaining mode, such as "cbc(aes-asm)", "cbc(aes)", or "cbc(__aes-aesni)".

1.3. Cryptographic Boundary

The Red Hat Enterprise Linux 6.6 Kernel Crypto API Cryptographic Module physical boundary is defined by the surface of the case of the platform. The logical module boundary is depicted in the software block diagram and is embodied by the Linux kernel and the kernel modules implementing the ciphers in `/lib/modules/$(uname -r)/kernel/crypto` (please note that only the modules implementing the approved mechanisms are available at runtime) and `/lib/modules/$(uname -r)/kernel/arch/x86/crypto`.

The integrity check mechanism provided by the sha512hmac application and the NSS library are part of the cryptographic module but not depicted as they are only used during load time of the module.

Hereafter are referred the Red hat Enterprise Linux 6.6 NSS Module CAVS certificates and algorithms used for the integrity check of the Red Hat Enterprise Linux 6.6 Kernel Crypto API Cryptographic Module:

Algorithm	Usage	CAVS certificates
HMAC SHA-512	HMAC calculation for the integrity check	Certs. #1933, #1934, #1935 and #1936
DSA (signature verification)	Signature verification when kernel object modules are inserted in the kernel	Certs. #892, #893, #894 and #895

Table 5: NSS algorithms and CAVS certificates used for the integrity check

The Red Hat Enterprise Linux 6.6 NSS Module received the FIPS certificate #2564.

1.3.1. Hardware Block Diagram

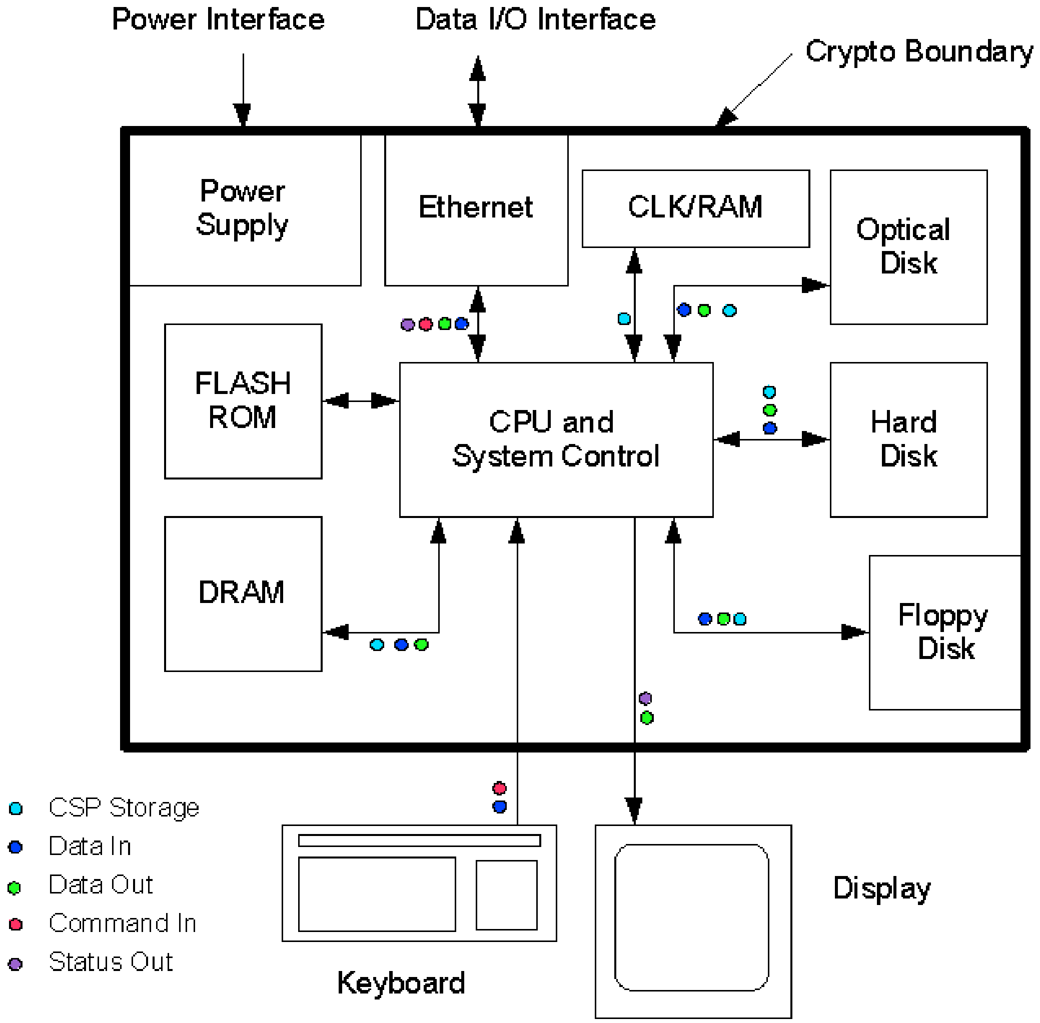

Figure 1: Hardware Block Diagram

This figure illustrates the various data, status and control paths through the Module. The physical boundary consists of the opaque enclosure surrounding the COTS PC system. The Module consists of standard integrated circuits, including processors, and memory. The Module does not include any security-relevant, semi- or custom integrated circuits or other active electronic circuit elements. The physical module includes power inputs and outputs, and internal power supplies. The cryptographic boundary contains only the security-relevant software elements that comprise the Module.

1.3.2. Software Block Diagram

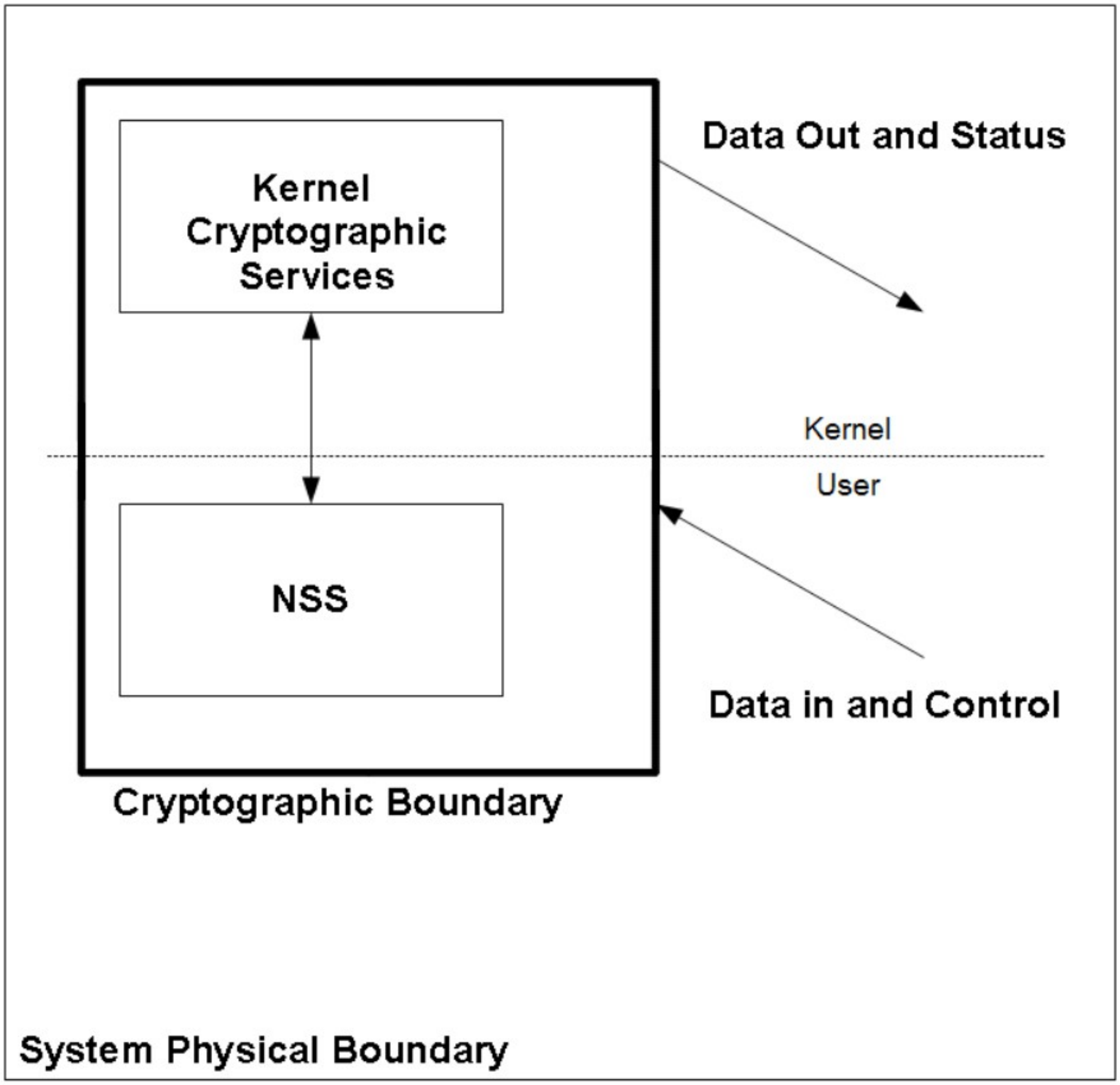

Figure 2: Software Block Diagram

1.4. Red Hat Enterprise Linux 6.6 Cryptographic Modules and FIPS 140-2 Certification

A set of kernel cryptographic libraries, services and user level cryptographic applications are certified at FIPS 140-2 level 1, providing a secure foundation for vendor use in developing dependent services, applications, and even purpose built appliances that may be FIPS 140-2 validated.

This section is informative to the reader to reference other cryptographic services of Red Hat Enterprise Linux. Only the software listed in section 1.2 is subject to the FIPS 140-2 validation.

The CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when supported if the specific operational environment is not listed on the validation certificate.

The certification is performed at FIPS 140-2 level 1, a software only certification that does not make any claims about the hardware enclosure. This allows vendors to develop their own higher level FIPS-validated modules by using services from any of these underlying cryptographic modules.

The following FIPS 140-2 validated cryptographic modules are included in the RHEL certification:

- Kernel Crypto API - a software only cryptographic module that provides general-purpose cryptographic services to the remainder of the Linux kernel
- Libgcrypt - supplies general cryptographic support for the Red Hat Enterprise Linux user space
- OpenSSL - a software library supporting cryptographic algorithms for general use by vendors
- OpenSSH-Server - supplies cryptographic support for the SSH protocol
- OpenSSH-Client - supplies cryptographic support for the SSH protocol
- NSS - open source C libraries designed to support cross-platform development of security-enabled applications

1.4.1. Platforms

The certification was performed on a 64-bit platform, which is capable of executing both 32 and 64-bit code concurrently. According to the Implementation Guidance for FIPS 140-2 document, IG G.5, this module will remain compliant if no source code changes (e.g., changes, additions, or deletions of code) are performed and the module is only recompiled and ported on a General Purpose Computer (GPC) operating in single-user mode.

However, the CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when ported and executed in an operational environment not listed on the validation certificate.

1.4.2. FIPS Approved Mode

Any vendor-provided FIPS 140-2 certified cryptographic modules and services that use the RHEL underlying services to provide cryptographic functionality must use the RHEL services in their approved mode. The approved mode ensures that FIPS required self-tests are executed and that ciphers are restricted to those that have been FIPS 140-2 validated by independent testing.

When invoking one of the non-FIPS approved ciphers which are still technically callable as listed above, the module implicitly transitions into non-FIPS approved mode. As all keys and Critical Security Parameters (CSP) handled by the module are ephemeral and there are no keys and CSPs shared between any functions, the Module transitions back into FIPS approved mode immediately when invoking one of the FIPS approved ciphers. A re-invocation of the self-tests or integrity tests is not required.

Even when using this non-FIPS approved mode, the module configuration ensures that the self-tests are always performed during initialization time of the module.

This section is given as guidance to developers to ensure the use cases conform with the initialization of the module. Although the Modules operates in FIPS approved mode when following the guidance, the initialization of the FIPS-validated module may consist of several phases which may be preempted by developers, but not administrators or users. The following explanation describes the initialization phase to ensure the FIPS-validated module initializes correctly.

If dm-crypt is used to encrypt a disk or partition, particularly when other modules may store cryptographic keys or other CSPs (Critical Security Parameters) there, it must be in FIPS approved mode as described in dm-crypt Security Policy.

If the kernel is in the FIPS approved mode, the remaining RHEL FIPS services (Crypto API, OpenSSL, Openswan, OpenSSH) start up in the FIPS approved mode by default. (Note that Openswan uses NSS for its cryptographic operations and NSS must explicitly be put into the FIPSapproved mode with the modutil command.)

The FIPS approved mode for a module becomes effective as soon as the module power on self tests complete successfully and the module loads into memory. Self tests and integrity tests triggered in RHEL at startup or on the first invocation of the crypto module are:

- kernel: during boot, before dm-crypt becomes available via dracut
- user space: before the user space module is available to the caller (i.e., for libraries during the initialization call, for applications: at load time)

See each module Security Policy for descriptions of self tests performed by each module and descriptions of how self test errors are reported for each module.

2. Cryptographic Module Ports and Interfaces

The physical ports of the Module are the same as the computer system on which it is executing.

The logical interface is the 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 actual API functions. The Status Output interface includes the return values of the API functions. The ports and interfaces are shown in the following tables.

FIPS Interface	Module's logical ports and interfaces
Data Input	API input parameters
Data Output	API output parameters
Control Input	API function calls
Status Output	API return codes
Power Input	Physical Power Connector

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. Installation of the Module is only done by the Crypto Officer. For User documentation, please refer to the official Linux kernel crypto API documentation.

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 Module's user documentation. The following table shows the services available to the various roles and the access to cryptographic keys and CSPs resulting from services.

Service	Algorithms	Role	CSPs	Generation	Storage	Zeroization	Access
Symmetric encryption/decryption	AES (ECB, CBC, CCM, GCM, CTR, XTS) Triple-DES (ECB, CBC)	User	symmetric key	Module's internal SP 800-90A DRBG	Volatile memory: the module does not support persistent storage of CSPs	Zeroized by a module's API function call when the cipher handler is released	read/write/execute
Keyed Hash (HMAC)	HMAC SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512	User	HMAC key	Module's internal SP 800-90A DRBG	Volatile memory: the module does not support persistent storage of CSPs	Zeroized by a module's API function call when the cipher handler is released	read/write/execute
Message digest (SHS)	SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512	User	none	none	none	none	read/write/execute
Random number generation	SP 800-90A DRBG (hash, HMAC and	User	SP800-90A DRBG: Entropy	Platforms non-deterministic	Volatile memory: the	Zeroized by a module's	read/write/execute

Service	Algorithms	Role	CSPs	Generation	Storage	Zeroization	Access
	AES-based)		input string, V, C and Key	random number generator	module does not support persistent storage of CSPs	API function call when the cipher handler is released	
Show status	none	User	none	none	none	none	execute
Module initialization	none	User	none	none	none	none	execute
Self-tests	All	User	none	none	none	none	read/execute
Zeroize	none	User	none	none	none	none	execute
Module installation	none	Crypto Officer	none	none	none	none	read/write

Table 7: Service Details for the FIPS mode

Service	Algorithms	Role	CSPs	Generation	Storage	Zeroization	Access
Symmetric encryption/decryption	AES (XTS with 192-bit keys) Triple-DES (CTR) DES	User	symmetric key	Module's internal SP 800-90A DRBG	Volatile memory: the module does not support persistent storage of CSPs	Zeroized by a module's API function call when the cipher handler is released	read/write/execute
Message digest (SHS)	SHA-256 and SHA-512 SSSE3, AVX and AVX2	User	none	none	none	none	read/write/execute
Show status	none	User	none	none	none	none	execute
Module initialization	none	User	none	none	none	none	execute
Self-tests	All	User	none	none	none	none	read/execute

Table 8: Service Details for the non-FIPS mode

More information about the services and their associated APIs can be found in official Linux kernel crypto API documentation.

The FIPS 140-2 module covers the static kernel binary and therefore provides a number of non-security services to callers within kernel space as well as user space. The following documents provide a description of these services:

- Linux system call API man pages provided in chapter 2 of the Linux man pages obtainable from https://github.com/mkerrisk/man-pages
- Linux kernel internals including interfaces between kernel components documented in the book ISBN-13: 978-0470343432
- Linux kernel driver development documentation covering the kernel interfaces available for device drivers: ISBN-13: 978-0596005900

The non-security services actually available will be based on the installed configuration.

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

This module will operate in a modifiable operational environment per the FIPS 140-2 definition.

5.1. Policy

The operating system shall be restricted to a single operator mode of operation (i.e., concurrent operators are explicitly excluded).

The kernel component that makes calls to the Module 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 not be used. In addition, other tracing mechanisms offered by the Linux environment, such as ftrace or kprobes (including systemtap) shall not be used.

6. Cryptographic Key Management

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.

CSPs	Generation	Storage	Zeroization	Access
Symmetric key (AES or Triple-DES)	Input through API call	Volatile memory: the module does not support persistent storage of CSPs	Zeroized by a module's API function call when the cipher handler is released	read/write/execute
HMAC key	Module's internal SP 800-90A DRBG	Volatile memory: the module does not support persistent storage of CSPs	Zeroized by a module's API function call when the cipher handler is released	read/write/execute
SP800-90A DRBG: Entropy input string, V, C and Key	Platform's non-deterministic random number generator	Volatile memory: the module does not support persistent storage of CSPs	Zeroized by a module's API function call when the cipher handler is released	read/write/execute

Table 9: CSPs management

6.1. Random Number Generation

The Module employs an SP800-90A compliant random number generator.

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

- initialization of a DRBG instance
- after 2^{48} requests for random numbers

6.2. Key/Critical Security Parameters (CSP) Authorized Access and Use by Role and Service/Function

Key usage is the responsibility of the calling program and is outside the scope of the module.

6.3. Key/CSP Storage

With respect to the Module, all keys are considered ephemeral. Any storage of keys or CSPs is the responsibility of the calling program and is outside the scope of the Module.

6.4. Key/CSP Zeroization

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

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

Product Name and Model: HP ProLiant DL380 Gen8
Regulatory Model Number: HSTNS-5163
Product Options: All
EMC: Class A

Product Name and Model: IBM System x3500 M4
Regulatory Model Number: 7383-AC1
Product Options: All
EMC: Class A

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, some functions require continuous verification of function, such as the Random Number Generator. All of these tests are listed and described in this section.

A failure of any of the power up self tests panics the Module. The only recovery is to reboot. For persistent failures, you must reinstall the kernel. See section 9.3 for details.

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

8.1. Power-Up Tests

The Module performs both power-up self tests (at module initialization) and continuous condition tests (during operation). 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 on demand by power cycling the Module or by rebooting the operating system.

For Known Answer Test, HMAC SHA-512 and DSA provided by the NSS library (Cert. #2564) are tested before the NSS library 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 9.1) is installed, the AES-NI implementation is self-tested with the same KAT vector as the other AES implementations.

For Software Integrity Test, a DSA (provided by the NSS library) calculation is performed by the NSS library to verify the integrity of the NSS library binary file. An HMAC SHA-512 (provided by the NSS library) calculation is performed on the sha512hmac utility and static Linux kernel binary to verify their integrity. The Linux kernel crypto API kernel modules, and any additional code modules loaded into the Linux kernel are checked with the DSA 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 modules providing the cryptographic functionality, the fact that the self test of these cryptographic modules are displayed implies that the integrity checks of each kernel module 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
DSA Signature Verification	Integrity Test
HMAC-SHA-512	Integrity Test
AES	KAT, encryption and decryption are tested separately
Triple-DES	KAT, encryption and decryption are tested separately

Algorithm	Test
SP 800-90A CTR DRBG	KAT
SP 800-90A Hash DRBG	KAT
SP 800-90A HMAC DRBG	KAT
HMAC-SHA-1, -224, -256, -384, -512	KAT
SHA-1, -224, -256, -384, -512	KAT

Table 10: Module Self-Tests

8.2. Conditional Tests

Algorithm	Test
DRBG	Continuous random number generator test

Table 11: Module Conditional Tests

9. Guidance

This section provides guidance for the Cryptographic Officer and the User to maintain proper use of the module per FIPS 140-2 requirements.

9.1. Cryptographic Officer Guidance

Additional kernel modules not provided with the RHEL FIPS-validated kernel RPM package must not be loaded as the kernel configuration is fixed in FIPS approved mode.

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 chapter 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 i386 and s390 are disallowed and not available on the test platforms.

The Linux Kernel Crypto API does implement DES, Triple-DES in CTR mode, and XTS using AES192 which are non-approved algorithms but shall not be used.

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 library, 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. SSSE3, AVX and AVX2 Implementation of SHA

The SSSE3, AVX and AVX2 implementation of SHA-256, and SHA-512 including the respective HMAC variants is not allowed in FIPS mode.

To ensure that these cipher implementations are not used, the following kernel modules must not be loaded:

- sha256-ssse3
- sha512-ssse3

This can be ensured by adding these modules into a modprobe black list as documented in modprobe(8).

9.2.2. XTS Usage

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

9.2.3. GCM Usage

The GCM block chaining mode must only be used in conjunction with the IPSEC stack of the Linux kernel due to the restrictions on the GCM IV generation mandated by SP800-38D.

9.3. Handling Self Test Errors

Self test failure within the Kernel Crypto API module or the dm-crypt kernel module 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. It also runs conditional tests on the DRBG whenever it is used and on asymmetric cryptographic keys whenever they are generated.

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.

10. Mitigation of Other Attacks

No other attacks are mitigated.

11. Glossary and Abbreviations

AES	Advanced Encryption Specification
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
ECB	Electronic Code Book
FSM	Finite State Model
HMAC	Hash Message Authentication Code
NIST	National Institute of Science and Technology
O/S	Operating System
RNG	Random Number Generator
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SSH	Secure Shell

Table 12: Abbreviations

12. References

[1] Kernel Cryptographic Services User Guide

[2] FIPS 140-2 Standard, http://csrc.nist.gov/groups/STM/cmvp/standards.html

[3] FIPS 140-2 Implementation Guidance, http://csrc.nist.gov/groups/STM/cmvp/standards.html

[4] FIPS 140-2 Derived Test Requirements,http://csrc.nist.gov/groups/STM/cmvp/standards.html

[5] FIPS 197 Advanced Encryption Standard, http://csrc.nist.gov/publications/PubsFIPS.html

[6] FIPS 180-4 Secure Hash Standard, http://csrc.nist.gov/publications/PubsFIPS.html

[7] FIPS 198-1 The Keyed-Hash Message Authentication Code (HMAC), http://csrc.nist.gov/publications/PubsFIPS.html

[8] FIPS 186-4 Digital Signature Standard (DSS), http://csrc.nist.gov/publications/PubsFIPS.html

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

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

[11] NIST SP 800-38C, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, http://csrc.nist.gov/publications/PubsFIPS.html

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

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

[14] NIST SP 800-56A, Recommendation for Pair-Wise Key Establishment Schemes using Discrete Logarithm Cryptography (Revised), http://csrc.nist.gov/publications/PubsFIPS.html

[15] NIST SP 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, http://csrc.nist.gov/publications/PubsFIPS.html