Red Hat Enterprise Linux - OpenSSL Module v2.0

FIPS 140-2 Security Policy

version 1.2

Last Update: 2012-07-19

Contents

1 Cryptographic Module Specification

This document is the non-proprietary security policy for the OpenSSL FIPS Object Module, and was prepared as part of the requirements for conformance to Federal Information Processing Standard (FIPS) 140-2, Level 1.

1.1 Description of the Module

The OpenSSL FIPS Object Module (hereafter referred to as the “Module”) is a software library supporting FIPS 140-2 -approved cryptographic algorithms. For the purposes of the FIPS 140-2 level 1 validation, the OpenSSL FIPS Object Module is Red Hat Enterprise Linux OpenSSL Module v2.0. All components of the Module will be in the OpenSSL RPM version 1.0.0-20.el6.

This Module provides a C language application program interface (API) for use by other processes that require cryptographic functionality.

For FIPS 140-2 purposes, the Module is classified as a multi-chip standalone module. The Module's logical cryptographic boundary is the shared library files and their integrity check HMAC files are: .libcrypto.so.1.0.0.hmac, .libcrypto.so.10.hmac, .libssl.so.1.0.0.hmac, .libssl.so.10.hmac, libcrypto.so.1.0.0, libcrypto.so.10, libssl.so.1.0.0, libssl.so.10. For x86_64 they are in the /usr/lib64 directory, for x86_32 (i386) they are in the /usr/lib directory. Please note that even though the AES-NI support is implemented as an OpenSSL Engine, the code is compiled as part of the libcrypto.so file, instead of an independent shared object. For dracut-fips RPM package version 004-256.el6_2.1:

/usr/share/dracut/modules.d/01fips/fips-boot.sh
/usr/share/dracut/modules.d/01fips/fips-noboot.sh
/usr/share/dracut/modules.d/01fips/fips.sh
/usr/share/dracut/modules.d/01fips/install
/usr/share/dracut/modules.d/01fips/installkernel

The Module's physical cryptographic boundary is the enclosure of the computer system on which it is executing.

The FIPS_mode_set() function verifies the integrity of the runtime executable using a HMAC SHA-256 digest computed at build time. If the digests match, the power-up self test is then performed. If the power-up self test is successful, FIPS_mode_set() sets the FIPS_mode flag to TRUE and the Module is in FIPS mode.

Security Component	FIPS 140-2 Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self Tests	1
Design Assurance	1
Mitigation of Other Attacks	1

Table 1: Security level of the Module

The Module has been tested in the following configurations:

- 32 bit x86_64
- 64 bit x86_64

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

Manufacturer	Model	O/S & Ver.
HP	Proliant DL585	Red Hat Enterprise Linux 6.2 (In Single User Mode)
IBM	HS22	Red Hat Enterprise Linux 6.2 (In Single User Mode)

Table 2: Tested platforms

1.2 Description of the Approved Mode

If the file `/proc/sys/crypto/fips_enabled' exists and contains a numeric of '1', the Module is put into FIPS approved mode at initialization time.

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

Algorithm	Validation Certificate	Usage	Keys/CSPs
AES	Certs. #1888, #1887, #1889, #1895, #1893, and #1894	encrypt/decrypt	AES keys 128, 192, 256 bits
Triple-DES	Certs. #1226, #1227, #1231, and #1232	encrypt/decrypt	Triple-DES keys 168 bits
DSA2	Certs. #592, #593, #597, and #598	sign, verify, and key generation	DSA keys 1024, 2048, 3072 bits
ANSI X9.31PRNG	Certs. #989, #990, #994, and #995	random number generation	PRNG seed value and seed key 128 bits
SHA-1	Certs. #1658, #1659, #1663, and #1664	hashing	N/A
SHA-224	Certs. #1658, #1659, #1663, and #1664	hashing	N/A
SHA-256	Certs. #1658, #1659, #1663, and #1664	hashing	N/A
SHA-384	Certs. #1658, #1659, #1663, and #1664	hashing	N/A
SHA-512	Certs. #1658, #1659, #1663, and #1664	hashing	N/A
HMAC-SHA-1	Certs. #1129, #1130, #1134, and #1135	message integrity	HMAC Key
HMAC-SHA224	Certs. #1129, #1130, #1134, and #1135	message integrity	HMAC Key
HMAC-SHA256	Certs. #1129, #1130, #1134, and #1135	message integrity	HMAC Key

Algorithm	Validation Certificate	Usage	Keys/CSPs
HMAC-SHA384	Certs. #1129, #1130, #1134, and #1135	message integrity	HMAC Key
HMAC-SHA512	Certs. #1129, #1130, #1134, and #1135	message integrity	HMAC Key
RSA (X9.31, PKCS #1.5, PSS)	Certs. #964, #965, #969, and #970	sign, verify, and key generation	RSA keys 1024 to 16384 bits

Table 3: Approved algorithms

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

Algorithm	Validation Certificate	Usage	Keys/CSPs
Diffie-Hellman	N/A, see caveat below	key agreement and establishment	None
RSA (encrypt, decrypt)	N/A, see caveat below	key wrapping	RSA keys 1024 to 16384 bits
MD5	NA, see caveat below	message digest	N/A

Table 4. Non-Approved algorithms

CAVEATS:

1) Diffie-Hellman (key agreement; key establishment methodology provides between 80 and 160 bits of encryption strength)

2)RSA (key wrapping; key establishment methodology provides between 80 and 160 bits of encryption strength)

3) MD5 for use in TLS only

4) The strength of AES encryption keys is between 128 and 160 bits

5) The module generates cryptographic keys whose strengths are modified by available entropy – 160 bits

1.3 Cryptographic Boundary

The Module's physical boundary is the surface of the case of the platform (depicted in the hardware block diagram). The Module's logical boundary is depicted in the software block diagram.

1.3.1 Hardware Block Diagram

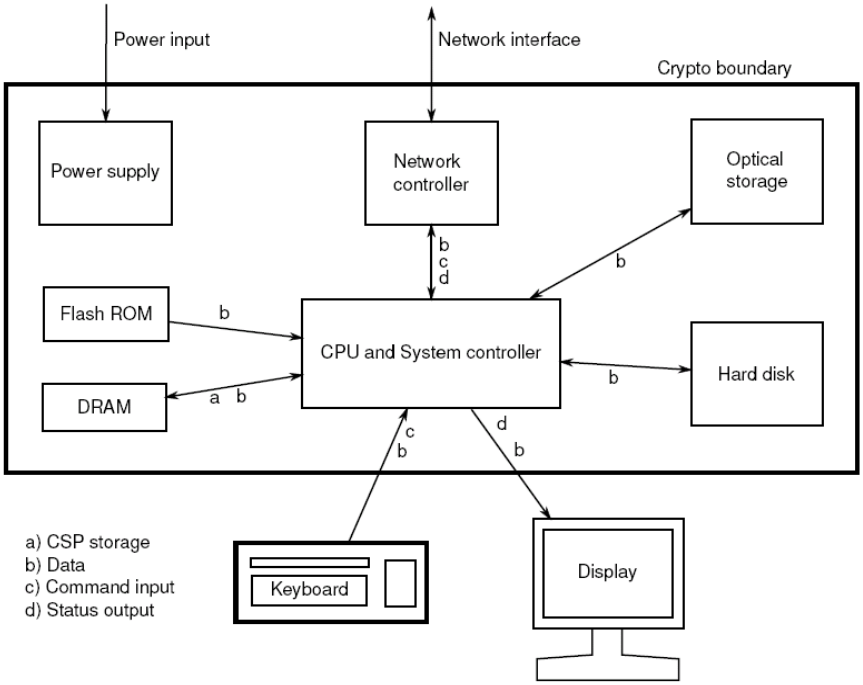

Figure 1. Hardware Block Diagram

1.3.2 Software Block Diagram

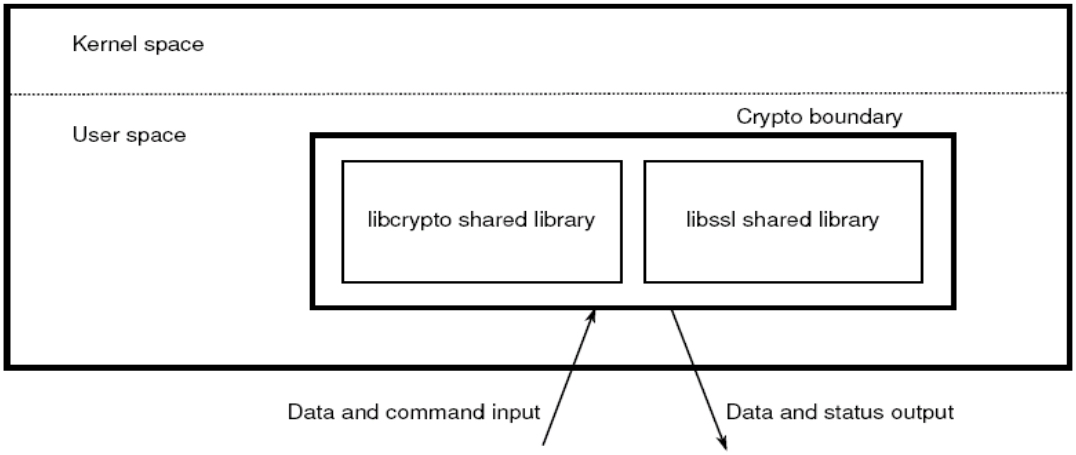

Figure 2. Software Block Diagram

1.4 Red Hat Enterprise Linux 6.2 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 certified.

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 helps vendors more easily develop their own higher-level FIPS 140-2 certified modules by using services from any of these underlying cryptographic modules.

The following FIPS 140-2 certified cryptographic modules are included in the RHEL6.2 certification:

- Kernel Crypto API - a software-only cryptographic module that provides general purpose cryptographic services to the remainder of the Linux kernel
- Disk Volume Encryption - provides disk management and transparent partial or full disk encryption Partial disk encryption encrypts only one or more partitions, leaving at least one partition as plaintext
- 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
- Openswan - provides the IKE protocol version 1 and version 2 key agreement services required for IPSec

1.4.1 Platforms

The certification was performed on a 64-bit system, which are capable of executing both 32 and 64-bit code concurrently. Vendors can "transfer" the FIPS 140-2 certificate to the other similar platforms, provided the binary is only recompiled without changing the code. This is called a vendor assertion.

1.4.2 FIPS Approved Mode

Any vendor-provided FIPS certified cryptographic modules and services that use the RHEL6.2 underlying services to provide cryptographic functionality must use the RHEL6.2 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 certified by independent testing.

The kernel services (Kernel Crypto API) must be in FIPS approved mode as many services rely on these underlying services for their cryptographic capabilities. See the Kernel Crypto API Security Policy for instructions.

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 approved mode, the remaining RHEL6.2 FIPS services (libgcrypt, OpenSSL, Openswan, OpenSSH) start up in the approved mode by default. (Note that Openswan uses NSS for its cryptographic operations and NSS must explicitly be put into the approved mode with the modutil command.)

The 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 RHEL6.2 at startup or on the first invocation of the crypto module:

- kernel - during boot, before dm-crypt becomes available via initrd
- 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 a C-language application program interface (API).

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

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

Table 5: 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. Install is only done by the Crypto Officer. For User documentation, please refer to the man pages of ssl(3), crypto(3) as an entry into the Module's API documentation for SSL/TLS and generic crypto support.

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	Role	Algorithms and CSPs	Access
Symmetric encryption/decryption	User, Crypto Officer	symmetric key AES, TDES	read/write/execute
Digital signature	User, Crypto Officer	asymmetric private key RSA, DSA	read/write/execute
Symmetric key generation	User, Crypto Officer	symmetric key AES, TDES	read/write/execute
TLS	User, Crypto Officer	symmetric key AES, TDES asymmetric public/private key RSA, HMAC Key	read/write/execute
TLS key agreement	User, Crypto Officer	symmetric key AES, TDES asymmetric public/private key RSA, HMAC Key, Premaster Secret, Master Secret and DH Secret.	read/write/execute
Certificate Management/ Handling	User, Crypto Officer	Certificates	read/write/execute
Asymmetric key generation	User, Crypto Officer	asymmetric private key	read/write/execute

Service	Role	Algorithms and CSPs	Access
		RSA, DSA	
Keyed Hash (HMAC)	User, Crypto Officer	HMAC Key, HMAC SHA-1, HMAC SHA-224, HMAC SHA-256, HMAC SHA-384, HMAC SHA-512	read/write/execute
Message digest (SHS)	User, Crypto Officer	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	read/write/execute
Random number generation	User, Crypto Officer	PRNG Seed and Seed Key	read/write/execute
Show status	User, Crypto Officer	none	execute
Module initialization	User, Crypto Officer	none	execute
Self test	User, Crypto Officer	HMAC SHA-256 key	read/execute
Zeroize	User, Crypto Officer	symmetric key, asymmetric key, HMAC key, Seed and Seed key	read/write/execute

Table 6: Service details

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 operates in a modifiable operational environment per the FIPS 140-2 definition.

5.1 Policy

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

The application that makes calls to the cryptographic Module is the single user of the cryptographic 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. In addition, other tracing mechanisms offered by the Linux environment, such as ftrace or 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.

6.1 Random Number Generation

The Module employs an ANSI X9.31-compliant random number generator for creation of asymmetric and symmetric keys.

The Module provides /dev/urandom as a source of random numbers for RNG seeds. The Module initializes this pseudo device at system startup.

The Module performs continual tests on the random numbers it uses, to ensure that the seed and seed key input to the Approved RNG do not have the same value. The Module also performs continual tests on the output of the approved RNG to ensure that consecutive random numbers do not repeat.

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

An authorized application as user (the User role) has access to all key data generated during the operation of the Module.

6.3 Key/CSP Storage

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

6.4 Key/CSP Zeroization

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

7 Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

Product Name and Model: HP ProLiant Server DL585 Series
Regulatory Model Number: HSTNS-1025
Product Options: All
EMC: Class A

Product Name and Model: IBM BladeCenter HS22 Series
Regulatory Model Number: 09-EMCRTP-0008
Product Options: All
EMC: Class A
1

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.

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

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

8.1 Power-Up Tests

The 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 because the Module is single-threaded and will not return to the calling application until the power-up self tests are complete. If the power-up self tests fail, subsequent calls to the Module will also fail - thus no further cryptographic operations are possible.

Algorithm	Test
AES	KAT
Triple-DES	KAT
DSA	pairwise consistency test, sign/verify
RSA	KAT
PRNG	KAT
HMAC-SHA-1	KAT
HMAC-SHA-224	KAT
HMAC-SHA-256	KAT
HMAC-SHA-384	KAT
HMAC-SHA-512	KAT
SHA-1	KAT
SHA-224	Tested as part of HMAC SHA-224
SHA-256	KAT
SHA-384	Tested as part of HMAC SHA-384
SHA-512	KAT
Module integrity	HMAC-SHA-256

Table 7: Module self tests

8.2 Conditional Tests

Algorithm	Test
DSA	Key generation

Algorithm	Test
RSA	Key generation
PRNG	Continuous test

Table 8: Module conditional tests

9 Guidance

9.1 Crypto Officer Guidance

The RPM package of the Module can be installed by standard tools recommended for the installation of RPM packages on a Red Hat Enterprise Linux system (for example, yum, rpm, and the RHN remote management tool).

For proper operation of the in-module integrity verification, the prelink has to be disabled. This can be done by setting PRELINKING=no in the /etc/sysconfig/prelink configuration file. If the libraries were already prelinked, the prelink should be undone on all the system files using the 'prelink -u -a' command.

Operators must first invoke OPENSSL_init(void) before using the Module.

Only the cipher types listed in section 1.2 are allowed to be used.

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

1. Install the dracut-fips package:

   ```
   # yum install dracut-fips
   ```

2. Recreate the INITRAMFS image:

   ```
   # dracut -f
   ```

After regenerating the initrd, the crypto officer has to append the following string to the kernel command line by changing the setting in the boot loader:

```
fips=1
```

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

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

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

"boot=/dev/sda1"

Reboot to apply these settings.

Because FIPS 140-2 has certain restrictions on the use of cryptography which are not always wanted, the Module needs to be put into FIPS approved mode explicitly. Two alternative mechanisms are provided to switch the Module into this mode:

- If the file /proc/sys/crypto/fips_enabled exists and contains a numeric value other than `0`, the Module is put into FIPS approved mode at initialization time. This is the mechanism recommended for ordinary use, activated by using the fips=1 option in the boot loader, as described above.
- If the application requests FIPS approved mode using the FIPS_mode_set() function call while passing a 1 as its parameter. This must be done prior to any initialization.

If an application that uses the OpenSSL module for its cryptography is put into a chroot environment, the crypto officer must ensure one of the above methods is available to the module from within the chroot environment to ensure entry into FIPS mode. Failure to do so will not allow the application to properly enter FIPS mode.

Once the OpenSSL module has been put into FIPS mode, it is not possible to switch back to standard mode without terminating the process first.

The version of the RPM containing the validated Module is version 1.0.0-20.el6. 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 operate the Module, the operating system must be restricted to a single operator mode of operation. (This should not be confused with single user mode which is runlevel 1 on RHEL. This refers to processes having access to the same cryptographic instance which RHEL ensures this cannot happen by the memory management hardware.)

9.2 User Guidance

The OpenSSL module must be operated in FIPS approved mode to ensure that FIPS 140-2 validated cryptographic algorithms and security functions are used.

Either of the following two methods may be used to initialize the Module in Approved mode:

- Explicitly invoke the Module in the approved mode by calling the FIPS_mode_set() function, which returns a “1” for success or a “0” for failure.
- Implicitly initialize the Module in the FIPS approved mode by calling OpenSSL_add_all_algorithms() and/or SSL_library_init() functions. These functions query the file /proc/sys/crypto/fips_enabled. If the file contains 1, the Module implicitly calls FIPS_mode_set(1) which ensures that the Module will operate in the FIPS approved mode. The application can query whether the FIPS approved mode is active by calling FIPS_mode() and it can query whether an integrity check or KAT self test failed by calling FIPS_selftest_failed(). See Appendix A for code examples.

Interpretation of the return code is the responsibility of the host application. Prior to invocation, the Module is uninitialized in non-FIPS mode by default.

ENGINE_register_* and ENGINE_set_default_* function calls are prohibited while in the Approved mode. Furthermore, once the Approved mode is entered, it must not be exited, which prohibits calls to FIPS_mode_set(0).

See Appendix A for skeleton code that illustrates how to invoke the OpenSSL Module.

9.3 Handling Self Test Errors

The effects of self test failures in the OpenSSL Module differ depending on the type of self test that failed.

The FIPS_mode_set() function verifies the integrity of the runtime executable using a HMAC SHA-256 digest, which is computed at build time. If this computed HMAC SHA-256 digest matches the stored, known digest, then the power-up self test (consisting of the algorithm-specific Pairwise Consistency and Known Answer Tests) is performed.

Non-fatal self test errors transition the Module into an error state. The application must be restarted to recover from these errors. The non-fatal self test errors are:

FIPS_R_FINGERPRINT_DOES_NOT_MATCH - The integrity verification check failed
FIPS_R_FIPS_SELFTEST_FAILED - a known answer test failed
FIPS_R_SELFTEST_FAILED - a known answer test failed
FIPS_R_TEST_FAILURE – a known answer test failed (RSA); pairwise consistency test failed (DSA)
FIPS_R_PAIRWISE_TEST_FAILED – a pairwise consistency test during DSA or RSA key generation failed
FIPS_R_FIPS_MODE_ALREADY_SET - the application initializes the FIPS mode when it is already initialized
RAND_R_PRNG_STUCK – the random number generator generated two same consecutive 128 bit values

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

A fatal error occurs only when the Module is in the error state (a self test has failed) and the application calls a crypto function of the Module that cannot return an error in normal circumstances (void return functions). The error message: 'FATAL FIPS SELFTEST FAILURE' is printed to stderr and the application is terminated with the abort() call.

The only way to recover from a fatal error is to restart the application. If failures persist, you must reinstall the Module. If you downloaded the software, verify the package hash to confirm a proper download.

10 Mitigation of Other Attacks

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

The API function of RSA_blinding_on turns blinding on for key rsa and generates a random blinding factor. The random number generator must be seeded prior to calling RSA_blinding_on.

Weak Triple-DES keys are detected as follows:

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

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

11 Glossary and Abbreviations

AES	Advanced Encryption Specification
CAVP	Cryptographic Algorithm Validation Program
CBC	Cypher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cypher Feedback
CMT	Cryptographic Module Testing
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CVT	Component Verification Testing
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
ECB	Electronic Code Book
FSM	Finite State Model
HMAC	Hash Message Authentication Code
LDAP	Lightweight Directory Application Protocol
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
NVLAP	National Voluntary Laboratory Accreditation Program
OFB	Output Feedback
O/S	Operating System
PRNG	Pseudo Random Number Generator
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SDK	Software Development Kit
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SLA	Service Level Agreement
SOF	Strength of Function
SSH	Secure Shell
TDES	Triple DES
UI	User Interface

12 References

[1] OpenSSL man pages where crypto(3) provides the introduction and link to all OpenSSL APIs regarding the cryptographic operation and ssl(3) to all OpenSSL APIs regarding the SSL/TLS protocol family

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

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

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

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

[6] FIPS 180-3 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 186-3 Digital Signature Standard (DSS), http://csrc.nist.gov/publications/PubsFIPS.html

[9] ANSI X9.52:1998 Triple Data Encryption Algorithm Modes of Operation, http://webstore.ansi.org/FindStandards.aspx?Action=displaydept&DeptID=80&Acro=X9&DpName=X9,%20Inc.

Appendix A: Code for Invoking the OpenSSL Module

The following code snippets demonstrate how to initialize OpenSSL and library FIPS mode features:

```
/* Initialize the library */
OpenSSL_add_all_algorithms();

/* Optionally initialize also the SSL library */
/* SSL_library_init(); */

/* Optionally call this code to force the FIPS mode regardless
of the system-wide settings */
/* if (!FIPS_mode()) {
if (!FIPS_mode_set(1)) {
/* FIPS_mode_set() failed */
}
/* } */

/* To query, whether the FIPS mode is active */
if (FIPS_mode()) {
/* Active */
} else {
/* Inactive */
}

/* To query, whether some of the FIPS selftests failed and the module is in
the error state */
if (FIPS_selftest_failed() {
/* Failed, error state */
} else {
/* FIPS mode is either inactive or FIPS selftests succeeded */
}
```