Red Hat Enterprise Linux OpenSSH Client Cryptographic Module version 4.0

FIPS 140-2 Non-Proprietary Security Policy

Version 1.2

Last update: 2016-06-13

Prepared by:
atsec information security corporation
9130 Jollyville Road, Suite 260
Austin, TX 78759
www.atsec.com

Table of Contents

1 Introduction

This document is the non-proprietary Security Policy for the Red Hat Enterprise Linux OpenSSH Client Cryptographic Module version 4.0. 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.

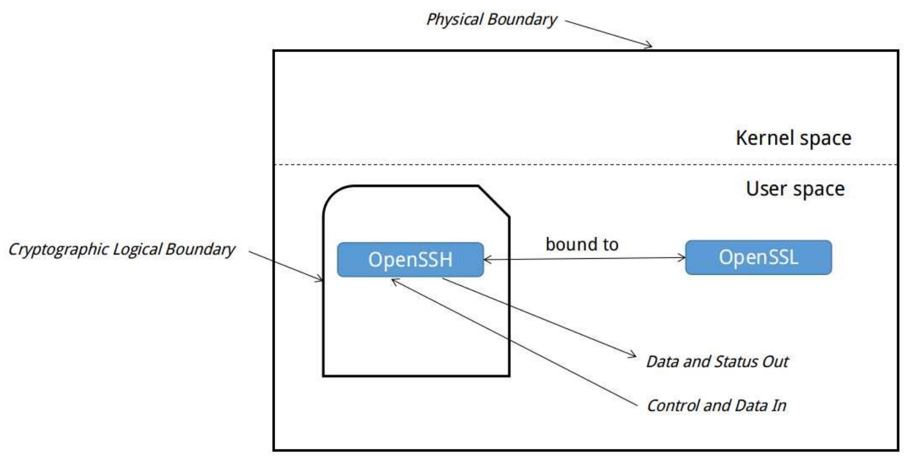
2 Cryptographic Module Specification

2.1 Module Overview

The Red Hat Enterprise Linux OpenSSH Client Cryptographic Module version 4.0 (hereafter referred to as “the module”) is a software library implementing the cryptographic support for the SSH protocol in the Red Hat Enterprise Linux user space.

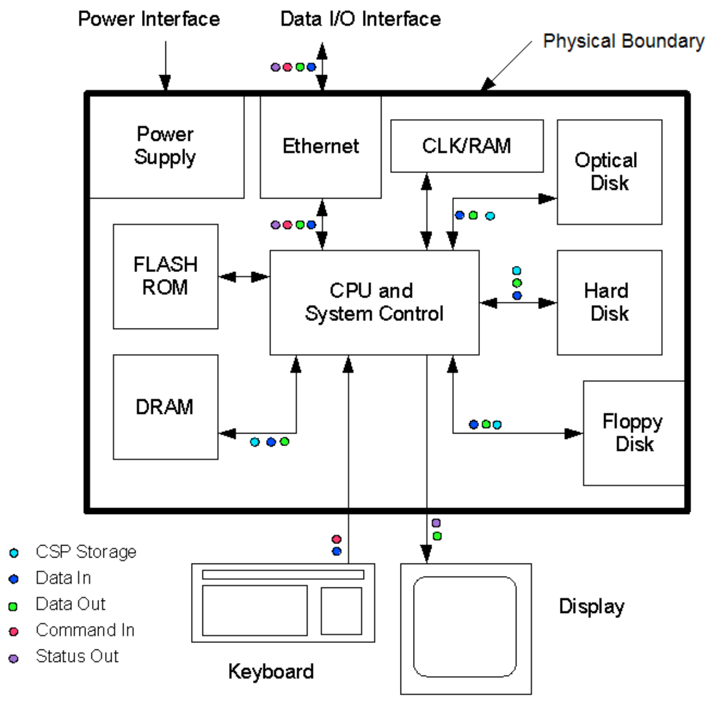
The module is implemented as a set of binary files.

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 module will use the Red Hat Enterprise Linux 7.1 OpenSSL Module (FIPS 140-2 Certificate #2441) as a bound module which provides the underlying cryptographic algorithms necessary for establishing and maintaining the SSH session. In addition the integrity check uses the cryptographic services provided by the Red Hat Enterprise Linux 7.1 OpenSSL Module as used by the utility application of fipscheck using the HMAC-SHA-256 algorithm.

This requires a copy of a Cert. #2441 validated version of the Red Hat Enterprise Linux 7.1 OpenSSL Module to be installed on the system for the current module to operate.

The cryptographic module combines a vertical stack of Linux components intended to limit the external interface each separate component may provide. The following software need to be installed for the module to operate:

- Red Hat Enterprise Linux OpenSSH Client Cryptographic Module with the version of the OpenSSH Client RPM file 6.6.1p1-12.el7_1
- The bound module of OpenSSL with FIPS 140-2 Certificate #2441
- The contents of the fipscheck RPM package (version 1.4.1-5.el7)
- The contents of the fipscheck-lib RPM package (version 1.4.1-5.el7).

The OpenSSH client RPM package of the Module includes the binary files, integrity check HMAC files and Man Pages. Any application other than the OpenSSH client application delivered with the aforementioned OpenSSH RPM packet is not part of the Module The FIPS certificate for this module will not be valid if any other application than the OpenSSH client application is used.

The files comprising the module are the following:

- /usr/sbin/ssh
- /usr/bin/fipscheck
- /usr/lib64/fipscheck/ssh.hmac
- /usr/lib64/fipscheck/fipscheck.so.1.2.1.hmac
- /usr/lib64/fipscheck/fipscheck.hmac
- /usr/lib64/fipscheck/libfipscheck.so.1.2.1.hmac
- /usr/lib64/libfipscheck.so.1.2.1

2.2 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:

Constructor	**Hardware**	**Processor**	**Operating System**	**Tested**		
				With AES-NI	**Without AES-NI**	**With CPACF**
HP	Proliant DL380p Gen8	Intel® Xeon® E5-2600 v3 product family	Red Hat Enterprise Linux 7.1	yes	yes	n/a
IBM	POWER8 Little Endian 8286-41A	POWER8E	Red Hat Enterprise Linux 7.1	n/a	n/a	n/a
IBM	z13	IBM/S390	Red Hat Enterprise Linux 7.1	n/a	n/a	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: Cryptographic Module Logical Boundary.

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 x86	AES-NI	AES

Table 3: PAA function implementations

2.3 Modes of Operations

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

The module turns to the FIPS approved mode after initialization and power-on self-tests succeed.

The mode of operation in which the module is operating can be determined by invoking a non FIPS-approved service: if the module is in the FIPS-mode, the service will fail. The user of the module can also check the flags `/proc/sys/crypto/fips_enabled`: if the value in this file is non-zero, the module will start in FIPS mode.

The Module verifies the integrity of the runtime executable using a HMAC-SHA-256 digest operation and compares the value with the build time pre-computed value. If the digests match, the power-up self-tests are then performed. If the power-up self-tests are successful, the `FIPS_mode` flag is set to TRUE and the Module is in the FIPS Approved mode.

The services available in FIPS mode can be found in section 4.2, Table 5.

3 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	Invocation of the ssh command on the command line or via the configuration file /etc/ssh/ssh_config and ~./.ssh/config Environment variable SSH_USE_STRONG_RNG	Keyboard, Ethernet port
Status Out	Status messages returned after the command execution	Display, Ethernet port
Data In	Input parameters of the ssh command on the command line with configuration file ~/.ssh/known_hosts, /etc/ssh/ssh_known_hosts, key files ~/.ssh/id_ecdsa*, ~/.ssh/id_rsa*, data via SSHv2 channel, data via local or remote port-forwarding port, data via TUN device	Keyboard, Ethernet port
Data Out	Output data returned by the ssh command	Display, Ethernet port

Table 4: Ports and Logical Interfaces

4 Roles, Services and Authentication

4.1 Roles

The module supports the following roles:

- **User role**: performs Key Derivation Function, Establish & Maintain SSH Session, Close SSH Session and Show Status
- **Crypto Officer role**: performs module installation and configuration, perform the selftests and terminate SSH Application

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

4.2 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 (CSPs) 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	Algo(s).	Note(s) / Mode(s)	CAVS Cert(s).	Role	CSPs	Access
Key Derivation	SP 800-135 Key Derivation Function (KDF) in the SSH protocol version 2	N/A	CVL Certs. #700, #701 and #702	User	Diffie-Hellman or EC Diffie-Hellman shared secret, derived keys	R, W, EX
Establish & Maintain SSH Session	SP 800-135 Key Derivation Function in the SSH protocol version 2	N/A	CVL Certs. #700, #701 and #702	User	RSA or ECDSA client private key Derived session encryption keys and derived data authentication (HMAC) keys	R, W, EX
Close SSH Session	N/A	Zeroize	N/A	User	Derived session encryption key and derived data authentication (HMAC) keys Shared secret	W
Terminate SSH Application	N/A	Zeroize	N/A	Crypto officer	Derived session encryption key and data authentication (HMAC) keys Shared secret	W
Self-Tests	HMAC-SHA-256 (uses the cryptographic services provided by the the bound OpenSSL	Integrity test invoked by restarting the module	N/A	Crypto officer	HMAC integrity key	R, EX

Service	Algo(s).	Note(s) / Mode(s)	CAVS Cert(s).	Role	CSPs	Access
	module)					
Show Status	N/A	Via verbose mode and exit codes	N/A	User	N/A	R, EX
Configure SSH Client	N/A	N/A	N/A	Crypto officer	N/A	R, EX
Installation	N/A	N/A	N/A	Crypto officer	N/A	R, EX

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

Note: The SSH protocol has not been reviewed or tested by the CAVP and CMVP.

Only the SP 800-135 Key Derivation Function has been validated by CAVP.

4.3 Authentication

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

5 Physical Security

The module is comprised of software only and thus does not claim any physical security.

6 Operational Environment

6.1 Applicability

The module operates in a modifiable operational environment per FIPS 140-2 Security Level 1 specifications. The module runs on a commercially available general-purpose operating system executing on the hardware specified in section 2.2.

6.2 Policy

The operating system is restricted to a single operator (concurrent operators are explicitly excluded). The application that requests 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.

7 Cryptographic Key Management

7.1 Random Number Generation

The module does not implement any random number generator nor provides key generation. The module only provides key derivation through the implementation of the SP 800-135 KDF.

The module calls the bound OpenSSL module to obtain the shared secret which will be used during the SSHv2 protocol initial handshake. The module derives keys from this shared secret through the SP 800-135 KDF implementation. When the module requests encryption/decryption services provided by the OpenSSL bound module, the resulting derived symmetric key (i.e. the output of the SP 800-135 KDF) will be passed to the OpenSSL bound module via API parameters.

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
Session Encryption Keys	Shared secret	N/A	Module's memory	Entry via API parameters Output: N/A	Zeroized by the ssh application
	Derived keys	N/A (Derived from the shared secret through the SP 800-135 KDF)	Module's memory	Entry: N/A Output via API parameters	Zeroized by the ssh application
Client Private Keys	RSA private keys	N/A	Module's memory	Via API parameters	Zeroized by the ssh application
	ECDSA private keys	N/A	Module's memory	Via API parameters	Zeroized by the ssh application
Software Integrity Key	HMAC Key	N/A	Module's binary file	N/A	N/A

Table 6: Keys/CSPs

7.2 Key / CSP Storage

Public and private keys are provided to the module by the calling process, and are destroyed when released. The module persistently stores public keys that are associated with a client username via the use of the file ~/.ssh/authorized_keys which is stored for each user individually in its home directory. This file is however not part of the module.

7.3 Key / CSP Zeroization

The destruction functions overwrite the memory occupied by keys with pre-defined values and deallocates the memory with the free() call. In case of abnormal termination, or swap in/out of a physical memory page of a process, the keys in physical memory are overwritten before the physical memory is allocated to another process.

8 EMI/EMC

Product Name and Model: HP ProLiant DL380p Gen8
Regulatory Model Number: HSTNS-5163
Product Options: All
EMC: Class A

Product Name and Model: IBM z13
Product Options: All
EMC: Class A

Product Name and Model: IBM Power8 Little Endian 8286-41A
Product Options: All
EMC: Class A

The HP Proliant DL380p Gen8, IBM z13 and IBM Power8 Little Endian 8286-41A test platforms have “been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.”

9 Self-Tests

9.1 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 automatically triggered without any user intervention.

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

9.1.1 Integrity Tests

The integrity check is performed by the fipscheck application using the HMAC-SHA-256 algorithm implemented by the bound Red Hat Enterprise Linux 7.1 OpenSSL Module.

When the OpenSSH module starts, it triggers the power-on self-tests, including the software integrity test. The software integrity test, using the HMAC-SHA-256 algorithm, constitutes a known answer test for the HMAC-SHA-256 algorithm.

The user space integrity verification is performed as follows: the OpenSSH Client application links with the library libfipscheck.so which is intended to execute fipscheck to verify the integrity of the OpenSSH Client application file using the HMAC-SHA-256 algorithm. Upon calling the FIPSCHECK_verify() function provided with libfipscheck.so, fipscheck is loaded and executed, and the following steps are performed:

1. OpenSSL, loaded by fipscheck, performs the integrity check of the OpenSSL library files using the HMAC-SHA-256 algorithm
2. fipscheck performs the integrity check of its application file using the HMAC-SHA-256 algorithm provided by the OpenSSL Module
3. fipscheck automatically verifies the integrity of libfipscheck.so before processing requests of calling applications
4. The fipscheck application performs the integrity check of the OpenSSH Client application file. The fipscheck computes the HMAC-SHA-256 checksum of that and compares the computed value with the value stored inside the /usr/lib64/fipscheck/<applicationfilename>.hmac checksum file. The fipscheck application returns the appropriate exit value based on the comparison result: zero if the checksum is OK, an error code otherwise (which brings the OpenSSH Module into the error state). The libfipscheck.so library reports the result to the OpenSSH Client application.

If any of those steps fail, an error code is returned and the OpenSSH Module enters the error state.

9.1.2 Cryptographic algorithm tests

The power-up self tests for the SP 800-135 KDF is covered by the SHS Known-Answer-Tests performed by the bound Red Hat Enterprise Linux 7.1 OpenSSL Module.

10 Guidance

The following guidance items are to be used for assistance in maintaining the module's validated status while in use.

10.1 Crypto Officer Guidance

The version of the RPMs containing the FIPS validated Module is stated in section 2.1 above. The Red Hat Enterprise Linux 7.1 OpenSSL Module referenced in section 2.1 must be installed according to its Security Policy.

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.

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

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"
```

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: 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 an application that uses the 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 Approved mode. Failure to do so will not allow the application to properly enter FIPS Approved mode.

Once the Module has been put into FIPS Approved 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 the version listed in chapter 2.1. The integrity of the RPM is automatically verified during the installation of the Module and the Crypto Officer shall not install the RPM file if the RPM tool indicates an integrity error.

10.1.1 Configuration Changes and FIPS Approved Mode

Use care whenever making configuration changes that could potentially prevent access to the /proc/sys/crypto/fips_enabled flag (fips=1) in the file/proc. If the module does not detect this flag during initialization, it does not enable the FIPS approved mode. For example, executing the module inside a chroot without the proc file system available, would instantiate the module in non-approved mode.

All user space modules depend on this file for transitioning into FIPS approved mode.

10.1.2 OpenSSH and NSS

The OpenSSH client is able to store the RSA and ECDSA host / user keys when using the UseNSS configuration option. When NSS is used to manage RSA / DSA keys, the signature generation and verification of the host or user keys is performed by NSS as well.

The module of OpenSSH is defined to be bound to OpenSSL only as outlined in chapter 1. The crypto officer can only use the UseNSS configuration option when the FIPS 140-2 validated NSS library is installed and used by OpenSSH.

10.1.3 OpenSSH Configuration

The user must not use DSA keys for performing key-based authentication as OpenSSH only allows DSA keys with 1024 bit size which are disallowed as per SP800-131A.

The user must not accept DSA host keys potentially offered during the first contact of an SSH server as OpenSSH only allows DSA keys with 1024 bit size which are disallowed as per SP800- 131A.

When re-generating RSA host keys, the crypto officer should generate RSA keys with a size of 2048 bit or higher according to [SP800-131A]. The crypto officer should inform the user base to not use RSA keys with key sizes smaller than 2048 bits.

In FIPS 140-2 mode, the module enforces the following restrictions. When these restrictions are violated by configuration options or command line options, the module will not be in the FIPS mode of oepration:

- SSH protocol version 1 is not allowed
- GSSAPI is not allowed
- Only the following ciphers are allowed:
  - aes128-ctr
  - aes192-ctr
  - aes256-ctr
  - aes128-cbc
  - aes192-cbc
  - aes256-cbc
  - 3des-cbc
  - rijndael-cbc@lysator.liu.se

Only the following message authentication codes are allowed:

- hmac-sha1
- hmac-sha2-256
- hmac-sha2-512

- hmac-sha1-etm@openssh.com
- hmac-sha2-256-etm@openssh.com
- hmac-sha2-512-etm@openssh.com

10.2 User Guidance

See the ssh(1) man page for general usage documentation of the ssh client.

When connecting to a previously unknown server, the user will be prompted to verify a fingerprint of the server's public key. This must be done by consulting a trusted source.

10.2.1 Handling Self-Test Errors

OpenSSL's self tests failures may prevent OpenSSH from operating. See the Guidance section in the OpenSSL Security Policy for instructions on handling OpenSSL self test failures.

The OpenSSH self test consists of the software integrity test. If the integrity test fails, OpenSSH enters an error state. The only recovery from this type of failure is to reinstall the OpenSSH module. If you downloaded the software, verify the package hash to confirm a proper download.

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
CMT	Cryptographic Module Testing
CMVP	Cryptographic Module Validation Program
CPACF	Central Processor Assist for Cryptographic Functions
CSP	Critical Security Parameter
CTR	Counter Mode
CVT	Component Verification Testing
DES	Data Encryption Standard
DFT	Derivation Function Test
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
ICM	Integer Counter Mode
KAS	Key Agreement Schema
KAT	Known Answer Test
MAC	Message Authentication Code
NDF	No Derivation Function
NIST	National Institute of Science and Technology
NDRNG	Non-Deterministic Random Number Generator
OFB	Output Feedback
O/S	Operating System

PAA	Processor Algorithm Acceleration
PR	Prediction Resistance
PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SSH	Secure Shell
TDES	Triple DES
UI	User Interface
XTS	XEX-based Tweaked-codebook mode with ciphertext Stealing

Appendix B References

FIPS180-4	**Secure Hash Standard (SHS)** March 2012 http://csrc.nist.gov/publications/fips/fips180-4/fips 180-4.pdf
FIPS186-4	**Digital Signature Standard (DSS)** July 2013 http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
FIPS197	**Advanced Encryption Standard** November 2001 http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS198-1	**The Keyed Hash Message Authentication Code (HMAC)** July 2008 http://csrc.nist.gov/publications/fips/fips198 1/FIPS-198 1_final.pdf
PKCS#1	**Public Key Cryptography Standards (PKCS) #1: RSA Cryptography** Specifications Version 2.1 February 2003 http://www.ietf.org/rfc/rfc3447.txt
RFC3394	**Advanced Encryption Standard (AES) Key Wrap Algorithm** September 2002 http://www.ietf.org/rfc/rfc3394.txt
RFC5649	**Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm** September 2009 http://www.ietf.org/rfc/rfc5649.txt
SP800-38A	**NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
SP800-38B	**NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf
SP800-38C	**NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality** May 2004 http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated July20_2007.pdf
SP800-38D	**NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf
SP800-38E	**NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices** January 2010 http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf

SP800-38F	**NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP800-56A	**NIST Special Publication 800-56A Revision 2 - Recommendation for Pair Wise Key Establishment Schemes Using Discrete Logarithm Cryptography** May 2013 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800 56Ar2.pdf
SP800-56C	**Recommendation for Key Derivation through Extraction-then-Expansion** November 2011 http://csrc.nist.gov/publications/nistpubs/800-56C/SP-800-56C.pdf
SP800-67	**NIST Special Publication 800-67 Revision 1 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher** January 2012 http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf
SP800-90A	**NIST Special Publication 800-90A - Recommendation for Random Number Generation Using Deterministic Random Bit Generators** January 2012 http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf
SP800-90B	**NIST Draft Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation** August 2012 http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf
SP800-108	**NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions** October 2009 http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf
SP800-131A	**NIST Special Publication 800-131A - Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths** January 2011 http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf