Red Hat Enterprise Linux 5 Kernel Crypto API Cryptographic Module v1.0

FIPS 140-2 Security Policy

version 1.4

Last Update: 2010-07-01

Contents

Document History

Version	Date of Change	Author	Changes to Previous Version
0.1	2008-07-22	SHW - atsec	Initial
1.0	2009-11-02	SHW - atsec	Completion of document
1.1	2009-11-23	SHW - atsec	Listing of all CAVS certificate numbers
1.2	2010-04-13	SHW - atsec	Single User Mode
1.3	2010-06-13	SHW - atsec	Update of NSS CAVS certificates
1.4	2010-07-01	SHW - atsec	Update of SW block diagram

1.Cryptographic Module Specification

This document is the non-proprietary security policy for the Red Hat Enterprise Linux 5 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 Module

The Red Hat Enterprise Linux 5 Kernel Crypto API Cryptographic 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 5 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	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 Levels

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

Manufacturer	Model	O/S & Ver.
HP	HP Integrity Server RX2660	Red Hat Enterprise Linux 5.4 (Single User Mode)
HP	HP ProLiant Server DL585 (library in 64 bit word size and 32 bit word size)	Red Hat Enterprise Linux 5.4 (Single User Mode)

Table 2: Tested Platforms

This cryptographic module provides is 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 2.6.18-164.2.1.el5.

- 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 version of the utility is 0.9.6-1.el5 and is provided with the RPM package named hmaccalc.
- Network Security Service (NSS), a separately validated cryptographic module (FIPS 140-2 validation certificate #815) provides cryptographic algorithms and security functions for sha512hmac application performing the integrity validation of the static Linux kernel binary. The application uses this module in accordance with the Security Rules stated in the *NSS Cryptographic Module Version 3.11.4 Security Policy.* The RPM file that contains all of the files for the validated version of the Red Hat Enterprise Linux NSS Cryptographic module is version nss-3.12.6-2.el5_4. The vendor affirmation covering the tested hardware platforms can be found at: http://www.redhat.com/solutions/government/certifications/. Note: The NSS version subject to validation was 3.11.4. As the NSS FIPS140-2 certificate does not cover the IA64 hardware architecture, the source code was recompiled, without any change, for the IA64 hardware platform. This is consistent with the vendor affirmation requirements in the FIPS 140-2 Implementation Guidance, G.5 item 1) a) i).

1.2. Description of Approved Mode

In Approved mode the module will support the following Approved functions:

- AES encrypt / decrypt 128,192, 256 bit ECB, CBC, CTR, CTR(RFC3686), CCM (Cert #1224)
- Triple-DES encrypt / decrypt ECB, CBC (Cert #882)
- SHA-1, 256, 384, 512 (Cert #1125)
- HMAC SHA-1, 256, 384, 512 (Cert #715)
- RNG (X9.31) using AES 128 (Cert #679)
- DSA used for the integrity verification of kernel modules loaded at run-time of the Linux kernel – this DSA service is provided at run-time of the module (Cert #406)
- DSA for integrity verification of the NSS library at boot time of the module (Cert# 449)
- HMAC SHA-512 for integrity verification of the sha512hmac application and the static Linux kernel binary at boot time of the module (Cert# 812)

The Linux Kernel Crypto API implements the following Non-Approved algorithms, which shall not be used in the FIPS Approved mode of operation:

- DES
- Triple-DES in CTR mode

The module only has an approved mode of operation in which it performs FIPS-approved cryptographic algorithms and security functions.

There are two implementations of AES: aes-generic 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 platforms.

The module maintains a process flag to indicate that the module is in FIPS 140-2 Approved mode. The flag is provided in the file /proc/sys/crypto/fips_enabled. If this file contains a value of 1, the module operates in FIPS 140-2 approved mode.

1.3. Cryptographic Module Boundary

The Red Hat Enterprise Linux 5 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).

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.

1.3.1. Software Block Diagram

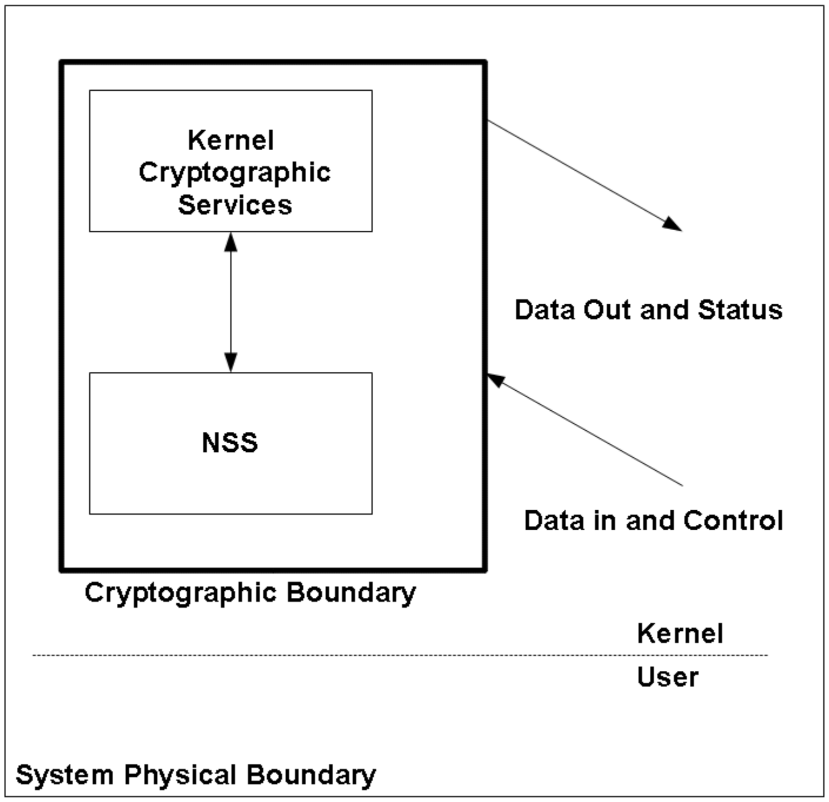

Figure 1: Software Block Diagram

1.3.2. Hardware Block Diagram

This figure illustrates the various data, status and control paths through the cryptographic 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.

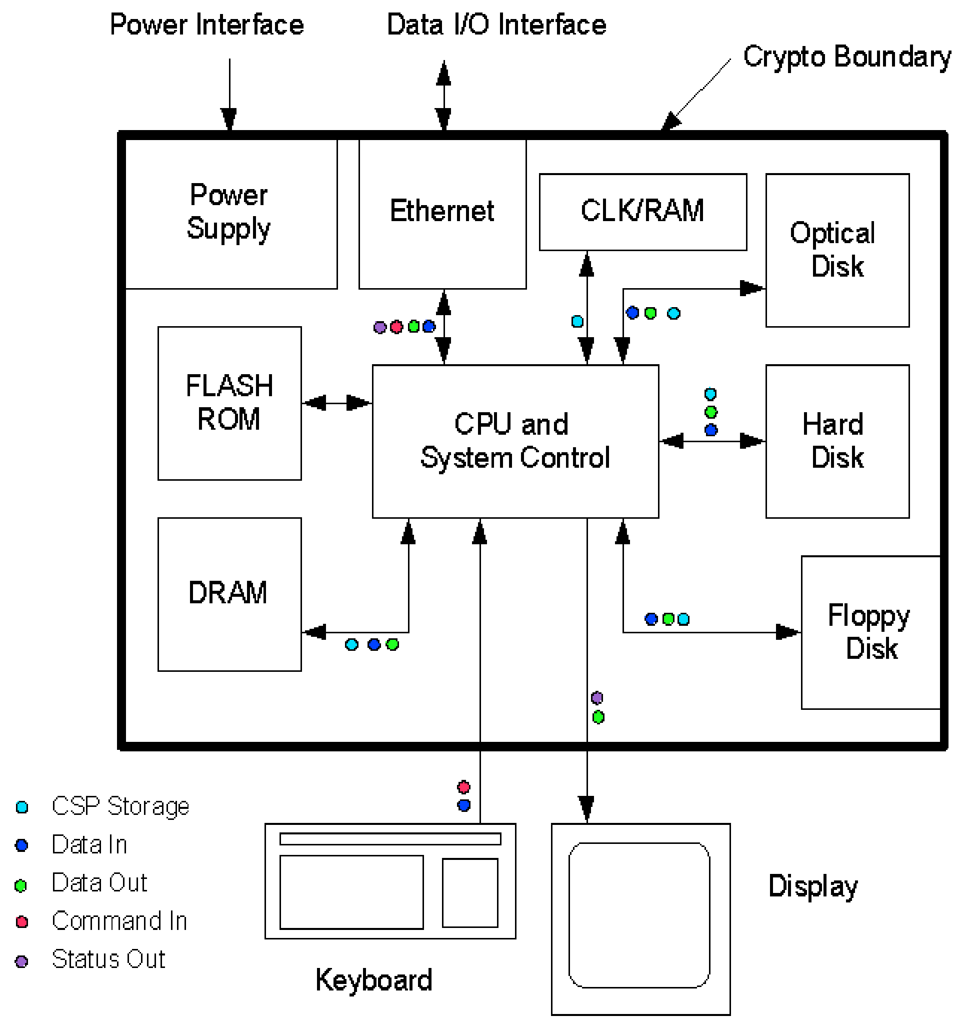

Figure 2: Hardware Block Diagram

2.Cryptographic Module Ports and Interfaces

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

FIPS Interface	***Physical Port***	***Module Interface***
Data Input	API	API input parameters
Data Output	API	API output parameters
Control Input	Kernel Command Line	API function calls
Status Output	Kernel log file (via dmesg)	API return codes
Power Input	PC Power Supply Port	Physical Power Connector

Table 3: Ports and Interfaces

3.Roles, Services and Authentication

3.1.Roles

Role	Services (see list below)
User	Encryption, Decryption, Random Numbers
Crypto Officer	Configuration, Encryption, Decryption, Random Numbers

Table 4: Roles

The Module meets all FIPS 140-2 level 1 requirements for Roles and Services, implementing both User and Crypto Officer roles. The Module does not allow concurrent operators.

The User and Crypto Officer roles are implicitly assumed by the entity accessing services implemented by the Module. No further authentication is required. The Crypto Officer can install and initialize the Module.

3.2.Services

See table in section 1.2 for the complete list of approved cryptographic services.

Role	**Service**	**CSP**	**Algo/Mode**	**API Calls**	**Access (Read, Write, Execute)**
User, Crypto Officer	AES encrypt / decrypt	128,192, 256 bit key	ECB, CBC, CTR, CTR(RFC3686), CCM	All API functions with the prefix of crypto_cipher_, crypto_ablkcipher_ and crypto_blkcipher_ crypto_free_ablkcipher crypto_has_ablkcipher ablkcipher_request_set_tfm ablkcipher_request_free	R, W, EX

Role	Service	CSP	Algo/Mode	API Calls	Access (Read, Write, Execute)
				ablkcipher_request_set_callback ablkcipher_request_set_crypt crypto_free_blkcipher crypto_has_blkcipher	
User, Crypto Officer	Triple-DES encrypt / decrypt	2 Key & 3 Key	ECB, CBC	All API functions with the prefix of crypto_cipher_, crypto_ablkcipher_ and crypto_blkcipher_ crypto_free_ablkcipher crypto_has_ablkcipher ablkcipher_request_set_tfm ablkcipher_request_free ablkcipher_request_set_callback ablkcipher_request_set_crypt crypto_free_blkcipher crypto_has_blkcipher	R, W, EX
User, Crypto Officer	SHA-1, SHA-256, SHA-384, SHA-512	N/A	N/A	All API functions with the prefix of crypto_digest_, crypto_hash_ crypto_free_hash crypto_has_hash	R, W, EX
User, Crypto Officer	HMAC SHA-1, HMAC SHA-256, HMAC SHA-384, HMAC SHA-512	HMAC Key for each SHA length	N/A	All API functions with the prefix of crypto_hmac_ crypto_free_hash	R, W, EX
User, Crypto Officer	RNG using AES 128	Seed and Seed Key	N/A	All API functions with the prefix of crypto_rng_ crypto_alloc_rng crypto_free_rng	R, W, EX
User, Crypto Officer	AEAD	This is a combined crypto protocol, that only supports approved algorithms that	N/A	All API functions with the prefix of crypto_aead_ and aead_request_ crypto_free_aead	R, W, EX

Role	Service	CSP	Algo/Mode	API Calls	Access (Read, Write, Execute)
		this module supports. Keys for each approved algorithm used will be associated as required.			
User, Crypto Officer	DSA used for the integrity verification of kernel modules loaded at runtime of the Linux kernel – this DSA service is provided at runtime of the module	DSA private key	N/A	N/A	R, W, EX
Crypto Officer	DSA for integrity verification of the NSS library at boot time of the module (	DSA private key	N/A	N/A	R, W, EX
Crypto Officer	HMAC SHA-512 for integrity verification of the sha512hmac application and the static Linux kernel binary at boot time of the module	HMAC Key	N/A	N/A	R, W, EX

Table 5: Services

3.3.Operator Authentication

There is no operator authentication, assumption of role is implicit by action.

3.4.Mechanism and Strength of Authentication

No authentication is required at security level 1, authentication is implicit by assumption of the role.

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 cryptographic module is the single user of the cryptographic module, even when the application is serving multiple clients.

In the FIPS approved mode the kernel debugger shall not be used.

6.Cryptographic Key Management

The module has a random number generator that returns key material for use in constructing cryptographic keys.

Key Generation – Key material generated by the module is ANSI X9.31 Appendix A.2.4 conformant.

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

Key Storage – With respect to the module, all keys are considered ephemeral. Any storage of keys is the responsibility of the calling program and is outside the scope of the module.

Key Destruction – 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).

6.1.Random Number Generation

The Module employs an ANSI X9.31 compliant random number generator for creation of keys. Note: the RNG seed is the tuple {V key DT}, where those values are defined in ANSI X9.31 Appendix A.2.4.

Based on AES 128; seed/seed key to be provided by calling user, usually by obtaining bits via get_random_bytes() which provides access to the Linux kernel hardware-based non-deterministic random number generator.

If the caller does not use the hardware RNG for seeding, the caller has to ensure that the seed and seed key for the DRNG is inserted into the DRNG consistent with FIPS 140-2 requirements, i.e. that they are not identical.

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

Product Name and Model: HP ProLiant Server DL585 Series
Regulatory Model Number: HSTNS-1025
Product Options: All
conforms to the following Product Specifications and Regulations:
EMC: Class A
CISPR 22:2005
EN 55022:2006
EN 55024:1998 +A1:2001 +A2:2003

EN 61000-3-2:2006
EN 61000-3-3:1995 +A1:2001 +A2:2005

Product Name and Model: HP Integrity Server rx2660
Regulatory Model Number: 1) RSVLA-0503
Product Options: All
conforms to the following Product Specifications and Regulations :
EMC: Class A
CISPR22:1997 / EN 55022:1998
CISPR 24:1997 + A1:2001 + A2: 2002 / EN 55024:1998 + A1:2001 + A2:2003
EN 61000-3-2:2000
EN 61000-3-3:1995 +A1:2001

8.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 as the module is single threaded at the time of 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.

Self Test	Description
Mandatory power-up tests performed at power-up and on demand:	
Cryptographic Algorithm Known Answer Tests	Each cryptographic algorithm provided by the Linux Kernel crypto API (see section 1.2 for list of algorithms), is tested using a “known answer” test to verify the correct operation of the algorithm with the exception of DSA. For complete details of the known answer tests, please see the Red Hat CryptoAPI Functional Spec document. DSA implemented by the Linux Kernel crypto API is verified by the kernel integrity test but may not be used in approved mode. NOTE: AES tests aes-x86_64 and aes-generic versions with the same KAT test vectors. HMAC SHA-512 and DSA provided by the NSS library are tested before the NSS library makes itself available to the sha512hmac application.
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.
Conditional tests performed, as needed, during operation:	
Continuous RNG	128 bits continuous testing is performed during each use of the approved RNG as well as the non-approved hardware/hybrid RNG. This test is a “stuck at” test to check the RNG output data for repetition of a value.
On Demand Execution of Self Tests	
On Demand Testing	Invocation of the self-tests on demand can be achieved by restarting the OS.

Table 6: Self-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 5.4 FIPS 140-2 validated kernel RPM package must not be loaded when in FIPS 140-2 approved mode as the kernel configuration is fixed in approved mode.

Secure installation and Startup:

Crypto officers use the Installation instructions to install the module in their environment.

The version of the RPM containing the validated module is stated in section 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 mode, the crypto officer has to regenerate the initrd by using the following command:

For the x86_64 platform, the command is:

```
mkinitrd --with-fips -f /boot/initrd-$(uname -r).img $(uname -r)
```

For the IA64, the command is:

```
mkinitrd --with-fips -f /boot/efi/efi/redhat/initrd-$(uname -r).img $(uname -r)
```

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
```

9.2.User Guidance

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

There are two implementations of AES: aes-generic 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 module does not have a non-approved mode of operation, however single DES is available and shall not be used. In addition, Triple-DES in CTR mode is available 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.

The Linux kernel crypto API-provided DRNG requires a reset operation after the initial boot sequence. This reset operation must be called by providing the seed and seed key for the DRNG. The caller should use the function get_random_bytes() to obtain the data for the seed and seed key. If the caller does not use get_random_bytes(), the caller has to ensure that the seed and seed key values for the DRNG are consistent with FIPS 140-2 requirements, i.e. that they are not identical.
Only the cryptographic mechanisms provided with the Linux kernel crypto API are considered to be used. The NSS library, although used in FIPS mode is only considered to support the integrity verification and is not intended for general-purpose use with respect to this cryptographic module.

10.Mitigation of Other Attacks

No other attacks are mitigated.

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
CC	Common Criteria
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
EAL	Evaluation Assurance Level
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
PP	Protection Profile
RNG	Randome Number Generator
RSA	Rivest, Shamir, Addleman
SAP	Service Access Points
SDK	Software Development Kit
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SOF	Strength of Function

SSH	Secure Shell
SVT	Scenario Verification Testing
TDES	Triple DES
TOE	Target of Evaluation
UI	User Interface

Table 7: Abbreviations

12.References

[1] kernel cryptographic services user guide (provided with installation RPM, see section 1.1 Description of Module for version)

[2] rx2660_EMIEMC_cert.pdf (On file at Red Hat)

[3] DL585_EMIEMC_CEcert.pdf (On file at Red Hat)

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

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

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

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

[8] FIPS 180-3 Secure Hash Standard, http://csrc.nist.gov/publications/PubsFIPS.html

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

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

[11] 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.