Samsung Kernel Cryptographic Module

FIPS 140-2 Non-Proprietary Security Policy

Version 1.12

Last Update: 2015-03-06

1. Introduction

This document is a non-proprietary FIPS 140-2 Security Policy for the Samsung Kernel Cryptographic Module. It contains a specification of the 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 multi-chip standalone software module. This security policy, v1.12, is for the revalidation of the Samsung Kernel Cryptographic Module to include information about new test platform and minor code changes. The security policy from the previous validated version of the module can be found on CMVP validation website under certificates #1648, #1915 and #2214.

1.1. Purpose of the Security Policy

There are three major reasons that a security policy is required:

- it is required for FIPS 140-2 validation,
- it allows individuals and organizations to determine whether the cryptographic module, as implemented, satisfies the stated security policy, and
- it describes the capabilities, protection, and access rights provided by the cryptographic module, allowing individuals and organizations to determine whether it will meet their security requirements.

1.2. Target Audience

This document is intended to be part of the package of documents that are submitted for FIPS validation. It is intended for the following people:

- Developers working on the release
- FIPS 140-2 testing lab
- Cryptographic Module Validation Program (CMVP)
- Consumers

2. Cryptographic Module Specification

This document is the non-proprietary security policy for the Samsung Kernel 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.

2.1. Description of Module

The Samsung Kernel Cryptographic Module is a software only security level 1 cryptographic module that provides general-purpose cryptographic services to the remainder of the Linux kernel. The cryptographic module runs on an ARMv7 processor.

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	3
Self-Tests	1
Design Assurance	3
Mitigation of Other Attacks	N/A

Table 1: Security Levels

The module has been tested on the following platforms:

Module/Implementation	Processor	Device	O/S & Ver.
Samsung Kernel Cryptographic Module (SKC1.5)	ARMv7	Samsung Galaxy Note 4	Android KitKat 4.4.4

Table 2: Tested Platforms

2.2. Description of Approved Mode

When the module is initialized, the self-tests are executed automatically at the loading time and the module enters FIPS-Approved mode automatically if the self-tests pass. A kernel proc file is set to indicate if device is in FIPS-Approved mode or in error state. The error state flag is used for the value of the process file /proc/sys/crypto/fips_status (return 0 if in FIPS-Approved mode; return 1 if in FIPS Error.)

Users can check the module status by connecting the device to a PC and issuing the following command, $adb shell cat /proc/sys/crypto/fips_status. (return 0 if in FIPS-Approved mode; return 1 if in FIPS Error.)

When the module is in the operational state, it can alternate service by service between FIPS-Approved mode and non-FIPS mode.

In the FIPS-Approved mode, the module provides the following Approved functions:

- AES (CBC, ECB) which can be C (aes-generic) or Assembly (aes-asm) implementations.
- SHA-1, SHA-224, SHA-256, SHA-384, SHA-512. SHA-1 can be C (sha1) or Assembly (sha1-asm) implementations.
- RNG (ANSI X9.31) using AES 128-bit when called with fips(ansi_cprng)
- Triple-DES (CBC, ECB)
- HMAC (with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512)

The cryptographic module provides 2 implementations of AES and SHA-1: regular C or streamlined assembler codes which are loaded in kernel module form. Note, if both implementations are loaded in the kernel, the respective implementation can be requested by using the following cipher mechanism strings with the initialization calls (such as crypto_alloc_blkcipher or crypto_alloc_hash):

- AES regular C code: “aes-generic”
- AES assembler code: “aes-asm”
- SHA-1 regular C code: “sha1-generic”
- SHA-1 assembler code: “sha1-asm”

The AES and SHA-1 implementation can also be loaded by simply using the string “aes” or “sha1” with the initialization call. In this case, the AES and SHA-1 implementation whose kernel module is loaded with the highest priority is used. In Samsung Kernel Cryptographic Module, assembler implementations are defined to have higher priority than the regular C implementation for both AES and SHA-1. Thus, only one of these implementations can be executed with an initialization call.

Kernel Crypto API implements the following non-Approved functions, which shall not be used in the FIPS-Approved mode of operation. Any use of the non-Approved functions will cause the module to operate in the non-FIPS mode implicitly:

- DES
- Twofish
- MD4
- MD5
- ansi_cprng
- krng
- ARC4
- Pcompress
- AES-XCBC (non-compliant)
- CRC32c
- Deflate
- LZO
- AES-GCM (non-compliant)
- RFC 4106 AES-GCM (non-compliant)
- RFC 4543 AES-GCM (non-compliant)
- AES-CTR
- Triple-DES-CTR
- GHASH
- GF128MUL

2.3. Cryptographic Module Boundary

2.3.1. Software Block Diagram

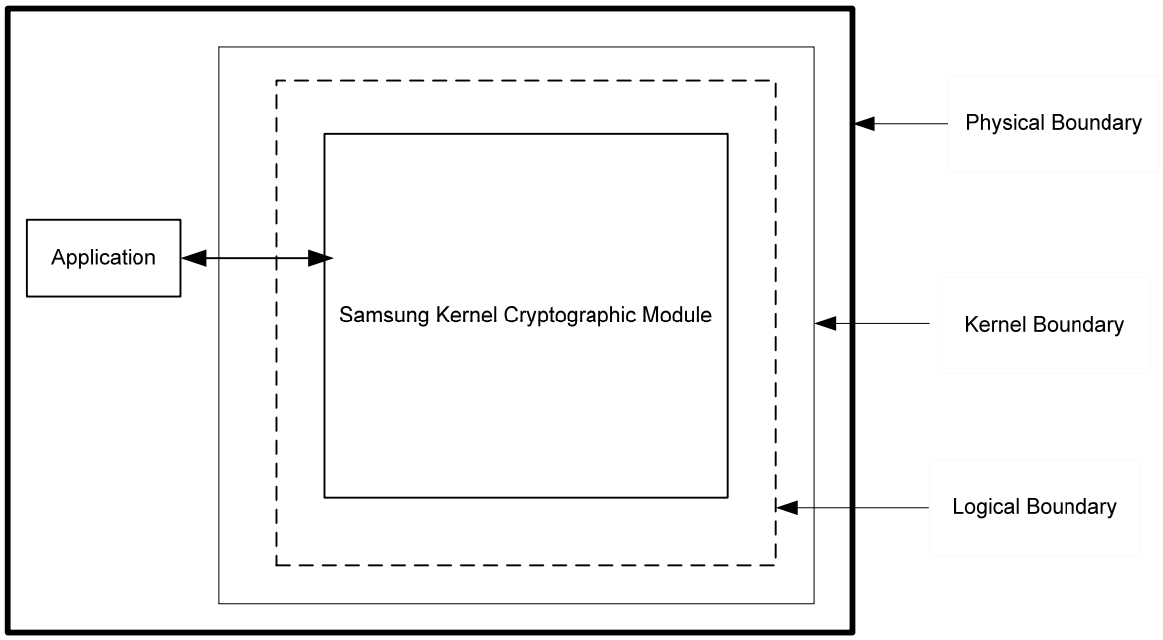

Figure 1: Software Block Diagram

The binary image that contains the Crypto API module for the appropriate platform is as follows:

- boot.img (version SKC1.5)

Related documentation:

- S/W Detailed Level Design v1.17
- Samsung Kernel Cryptographic Module FIPS 140-2 Non-Proprietary Security Policy (this document)

Note: The master component list is provided in Section 2.10 of S/W Detailed Level Design document.

2.3.2. Hardware Block Diagram

This figure illustrates the various data, status and control paths through the cryptographic module. Inside, the physical boundary of the module, the mobile device consists of standard integrated circuits, including processors and memory. These do not include any security-relevant, semi- or custom integrated circuits or other active electronic circuit elements. The physical boundary includes power inputs and outputs, and internal power supplies. The logical boundary of the cryptographic module contains only the security-relevant software elements that comprise the module.

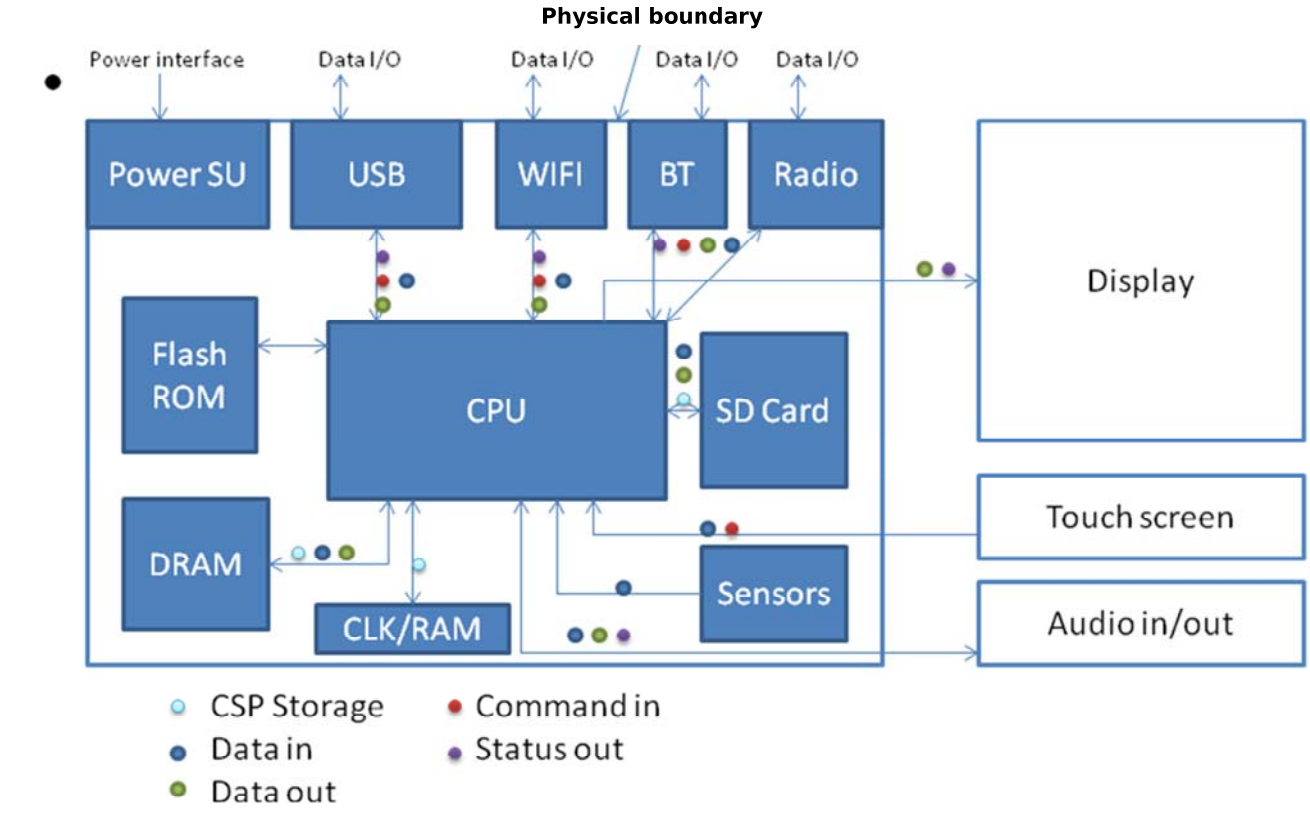

Figure 2: Hardware Block Diagram

3. Cryptographic Module Ports and Interfaces

FIPS Interface	Ports
Data Input	API input parameters
Data Output	API output parameters
Control Input	API function calls
Status Output	API return codes; kernel log file, /proc/sys/crypto/fips_status
Power Input	Physical power connector

Table 3: Ports and Interfaces

4. Roles, Services and Authentication

4.1. Roles

Role	Services (see list below)
User	Encryption, Decryption, Random Numbers Generation, Digest Creation, Self-Test, Get Status, Zeroization
Crypto Officer	Initialization of Module

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 initialize the Module.

4.2. Services

The following table identifies the Approved services in FIPS-Approved mode:

Role	Algorithms and Services	CSP	Modes	FIPS Approved (Cert #)	Standards	Access (Read, Write, Execute)
User	AES encryption and decryption	128, 192, 256 bit keys	ECB, CBC	Cert. #3007	FIPS 197	R, W, EX
User	AES (Assembly implementation) encryption and decryption	128, 192, 256 bit keys	ECB, CBC	Cert. #3010	FIPS 197	R, W, EX
User	HMAC (with SHA-1, SHA-224, SHA-256, SHA 384, SHA 512)	At least 112 bits HMAC key	N/A	Cert. #1900	FIPS 198-1	R, W, EX
User	HMAC (with SHA-1 Assembly implementation)	At least 112 bits HMAC key	N/A	Cert. #1902	FIPS 198-1	R, W, EX
User	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	N/A	N/A	Cert. #2514	FIPS 180-4	R, W, EX

Role	Algorithms and Services	CSP	Modes	FIPS Approved (Cert #)	Standards	Access (Read, Write, Execute)
User	SHA-1 Assembly Implementation	N/A	N/A	Cert. #2518	FIPS 180-4	R, W, EX
User	Triple-DES encryption and decryption	2 key & 3 key	CBC, ECB	Cert. #1766	SP 800-67	R, W, EX
User	RNG	128 bits seed and seed key	AES-128	Certs. #1304 and #1305	ANSI X9.31	R, W, EX
Crypto Officer	Initialization	N/A	N/A	N/A	N/A	N/A
User	Self-Test (Self-test is executed automatically when device is booted or restarted)	256 bits HMAC value for integrity test	N/A	N/A	N/A	R, EX
User	Check Status/Get State	N/A	N/A	N/A	N/A	R
User	Zeroization	AES key, Triple-DES key, HMAC key, seed and seed key	N/A	N/A	N/A	R, W, EX

Table 5: Approved Services

Note: The module supports 2-key and 3-key Triple-DES. Restriction applies to the 2-key Triple-DES that at most 2^{20} blocks of data can be encrypted with the same key in FIPS-Approved mode according to SP800-131A. Two-key Triple-DES encryption will be disallowed by NIST after 2015.

The following table identifies the non-Approved services in non-FIPS mode:

Role	Algorithms and Services	CSP	Access (Read, Write, Execute)
User	DES	56 bit keys	R, W, EX
User	Twofish	128, 192, 256 bit keys	R, W, EX
User	MD4	N/A	R, W, EX
User	MD5	N/A	R, W, EX
User	ansi_cprng (non-Approved RNG)	Seed	R, W, EX
User	krng (non-Approved RNG)	Seed	R, W, EX
User	ARC4	Various key sizes	R, W, EX
User	Pcompress (Partial (de)compression)	N/A	R, W, EX
User	AES-XCBC (RFC 3566; The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec)	128 bits keys	R, W, EX
User	CRC32c	N/A	R, W, EX
User	Deflate (Data compression	N/A	R, W, EX

Role	Algorithms and Services	CSP	Access (Read, Write, Execute)
	algorithm)		
User	LZO (Data compression algorithm)	N/A	R, W, EX
User	AES-GCM (non-compliant)	128, 192, 256 bit keys	R, W, EX
User	RFC 4106 AES-GCM (non-compliant)	128, 192, 256 bit keys	R, W, EX
User	RFC 4543 AES-GCM (non-compliant)	128, 192, 256 bit keys	R, W, EX
User	AES-CTR	128, 192, 256 bit keys	R, W, EX
User	Triple-DES-CTR	112, 168 bit keys	R, W, EX
User	GHASH	N/A	R, W, EX
User	GF128MUL	N/A	R, W, EX

Table 6: Non-Approved Services

The cryptographic module is part of the kernel image. The following documents provide a description and API functions of the cryptographic services listed above:

- https://www.kernel.org/doc/Documentation/crypto/api-intro.txt
- http://www.linuxjournal.com/article/6451?page=0,0
- Linux system call API man pages provided in chapter 2 of the Linux man pages obtainable from git://github.com/mkerrisk/man-pages.git
- 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
- Functional Design document provided by Samsung is available upon request.

4.3. Operator Authentication

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

4.4. Mechanism and Strength of Authentication

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

5. Finite State Machine

For information pertaining to the Finite State Model, please refer to the Functional Design document.

6. Physical Security

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

7. Operational Environment

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

7.1. Policy

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

The external application that makes calls to the cryptographic module is the single user of the cryptographic module, even when the application is serving multiple clients.

8. Cryptographic Key Management

The keys and CSPs in FIPS-Approved mode are described in the following table:

Algorithm	Keys/CSPs	Keys/CSPs Input	Keys/CSPs Output	Keys/CSPs Zeroization
AES Key	128-, 192- and 256-bits key	API input parameter	N/A	Call the zeroization API function for block cipher
Triple-DES	2-keyed or 3-keyed Triple-DES key	API input parameter	N/A	Call the zeroization API function for block cipher
HMAC	At least 112 bits HMAC key	API input parameter	N/A	Call the zeroization API function for hash
ANSI X9.31 RNG	128 bits seed and seed key	Seed is provided by the /dev/random	API output parameter	Call the zeorization API function for prng

Table 7: Key Input, Key Output and Key Zeroization for Keys/CSPs

8.1. Random Number Generation

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

The calling user provides the seed and seed key, usually by obtaining bits via get_random_bytes() from the Linux-provided /dev/random pseudo-device which is provided by the underlying Operating System. The caller must 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. Failure to comply with this requirement will cause the module to go into an Error state.

8.2. Key Generation

The module does not provide any key generation service or perform key generation for any of its Approved algorithms. Keys are passed in from clients via algorithm APIs. Seeds for random number generation are imported into the module.

8.3. Key Entry and Output

The module does not support manual key entry or key output. Keys or other CSPs can only be exchanged between the module and the calling application using appropriate API calls.

8.4. Key Storage

Keys are not stored inside cryptographic module. A pointer to plaintext key is passed through. Intermediate key storages are immediately assigned to Zero.

8.5. Zeroization Procedure

The zeroization mechanism for all of the CSPs is to replace 0s in the memory which originally store the CSPs. It is the calling application responsibility to call the appropriate key zeroization API function to zeroize the keys/CSPs after their use. For more details on zeroization API functions, please refer to the Functional Design document which is provided by Samsung upon request.

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

Lab Name: PCTEST Engineering Laboratory, Inc.
FCC Registration: #90864

The test device which runs the module conforms to the EMI/EMC requirements specified by 47 Code of Federal Regulations, Part 15, Subpart B, Unintentional Radiators, Digital Devices, Class B (i.e., for home use).

For information related to FCC ID of the devices, please refer to the Functional Design document which is provided by Samsung upon request.

10. Self-Tests

Self-test uses the existing Crypto API tcrypt module to perform known-answer self-test of algorithms. The module is configured as a built-in kernel module instead of a loadable module as is the case of Linux Crypto API. Tests of all FIPS-approved algorithms are executed. The self-tests are run during early-kernel startup when built-in kernel modules are initialized. Self-tests can also be invoked by the user by restarting the device. When self-tests are done successfully, the proc entry for /proc/sys/crypto/fips_status will return 0. A binary integrity test will then be performed in call from tcrypt. If self-test or integrity test fail, an error flag (static variable) is set, an error code returns to the API function caller to indicate the error, the module enters in an error state (FIPS_ERR), and Crypto APIs that return cryptographic information is blocked. The proc entry for /proc/sys/crypto/fips_status will return 1 when the module is in ERROR state.

10.1. Power-Up Tests

At module start-up, Known Answer Tests are performed. These tests are automatic and do not need operator intervention. If the value calculated and the known answer does not match, the module immediately enters into FIPS_ERR state. Once the module is in FIPS_ERR state, the module becomes unusable via any interface.

The module implements each of the following Known Answer Tests separately:

- AES encryption and decryption tested separately, with and without AES assembler
- Triple-DES encryption and decryption tested separately
- HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, with and without SHA-1 assembler
- SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, with and without SHA-1 assembler
- ANSI X9.31 Random Number Generator

10.2. Integrity Check

At build time -

- Calculate a hmac(sha256) of only the module code (available in .text, .rodata, .init.text and .exit.text sections of kernel image).
- Once generated, the build time hmac of the module code is updated in the kernel image itself as a read-only data.

At run time –

- Calculate a hmac(sha256) of only the module code, available in running kernel on the device (.text, .init.text, .exit.text and .rodata sections of running kernel)

If this run-time hmac is same as build time hmac, Integrity test is considered passed. If calculated and stored values do not match, set error state, FIPS_ERR.

10.3. Conditional Tests

A continuous random number generator test is performed during each use of the Approved ANSI X9.31 RNG. If values of two consecutive random numbers match, then cryptographic module goes into error state. A CRNG test is also implemented for the Linux provided /dev/random RNG which is usually used by calling user for seeding the ANSI X9.31 RNG.

11. Design Assurance

11.1. Configuration Management

Perforce is used as the repository for both source code and documents. All source code and documents are maintained in internal server. Release is based on the Changelist number, which is auto-generated. Every check-in process creates a new Changelist number.

Versions of controlled items include information about each version. For documentation, revision history inside the document provides the current version of the document. Version control maintains the all the previous version and the version control system automatically numbers revisions. For source code, unique information is associated with each version such that source code versions can be associated with binary versions of the final product. The source code of the module (version SKC1.5) available in the Samsung internal Perforce repository, as listed in Functional Design document, is used to build target binary.

11.2. Delivery and Operation

The cryptographic module is never released as Source code. The module sources are stored and maintained at a secure development facility with controlled access.

This cryptographic module is built-in along with the Linux Kernel. Product that does not need FIPS 140-2 certified cryptographic module may decide to change the build flag CONFIG_CRYPTO_FIPS in Kernel config. The development team and the manufacturing factory share a secured internal server for exchanging binary software images. The factory is also a secure site with strict access control to the manufacturing facilities. The module binary is installed on the mobile devices (phone and tablets) using direct binary image installation at the factory. The mobile devices are then delivered to mobile service operators. Users cannot install or modify the module. The developer also has the capability to deliver software updates to service operators who in turn can update end-user phones and tablets using Over-The-Air (OTA) updates. Alternatively, the users may bring their mobile devices to service stations where authorized operators may use developer-supplied tools to install software updates on the phone. The developer vets all service providers and establishes secure communication with them for delivery of tools and software updates. If the binary is modified by unauthorized entity, the device has a feature to detect the change and thus not accept the binary modified by an unauthorized entity.

12. Mitigation of Other Attacks

No other attacks are mitigated.

13. Glossary and Abbreviations

AES	Advanced Encryption Specification
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CMT	Cryptographic Module Testing
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
DES	Data Encryption Standard
FSM	Finite State Model
HMAC	Hash Message Authentication Code
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
O/S	Operating System
RNG	Random Number Generator
SDK	Software Development Kit
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
TDES	Triple DES

14. References

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

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

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

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

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

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

[7] SP800-67 Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, http://csrc.nist.gov/publications/PubsFIPS.html

[8] ANSI X9.31 Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)