Samsung Kernel Cryptographic Module

FIPS 140-2 Non-Proprietary Security Policy

Version 1.19

Last Update: 2016-06-23

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, v1.19, is for the revalidation of the Samsung Kernel Cryptographic Module (Version SKC1.7) to include information about new test platform and minor code changes. The security policy from the previous validated versions of the module can be found on the CMVP validation's website under certificates #1648, #1915, #2214, #2337 and #2430.

1.1. Purpose of the Security Policy

There are three major reasons for which a security policy is required:

- it is required for a 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 is 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 ARMv8 processor with Crypto Extensions.

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 & Version	Processor	Device	O/S & Version
Samsung Kernel Cryptographic Module (SKC1.7)	ARMv8 with Crypto Extensions	Samsung Galaxy S7	Android Marshmallow 6.0.1

Table 2. Tested Platforms

Note: Per FIPS 140-2 Implementation Guidance (IG) G.5, CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

2.2. Description of the Approved Mode

When the module is initialized, the self-tests are executed automatically at the loading time and the module enters the FIPS-Approved mode automatically if the self-tests pass. A kernel proc file is set to indicate if the device is in the FIPS-Approved mode or in the error state. The value of the

/proc/sys/crypto/fips_status contains 0 if the module is in the FIPS-Approved mode; and contains 1 if the module is in the error state (FIPS_ERR.)

Users can check the module status by connecting the device to a General Purpose Computer (GPC) and issuing the following command, $adb shell cat /proc/sys/crypto/fips_status to display the content of the /proc/sys/crypto/fips_status file.

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

The module provides the following CAVP validated algorithms which are written in C:

Algorithms	Standards	CAVS certificates
AES (ECB, CBC mode) encryption and decryption with 128, 192, 256 bits key	FIPS 197	Cert. #3837
AES (GCM mode) encryption and decryption with 128, 192, 256 bits key, IV generated externally	SP 800-38D	Cert. #3837
HMAC with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	FIPS 198-1	Cert. #2488
SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	FIPS 180-4	Cert. #3161
Hash-based DRBG with SHA-1, SHA-256, SHA-384, SHA-512	SP 800-90A	Cert. #1083
HMAC-based DRBG with SHA-1, SHA-256, SHA-384, SHA-512	SP 800-90A	Cert. #1083
Block Cipher (CTR) based DRBG with AES-128, AES-192, AES-256	SP 800-90A	Cert. #1083

Table 3. CAVP validated algorithms provided by the module

Note: The AES GCM mode has been validated by CAVP, but it is listed as a non-Approved service for this module. Thus, it shall not be used in FIPS Approved mode. Please refer to Section 4.2 "Services" in this document for the complete list of services in FIPS-Approved mode and non-FIPS mode.

Besides the cryptographic algorithms written in C language, the module also supports the use of the Crypto Extensions instructions provided by ARMv8 for AES, SHA-1, SHA-224 and SHA-256 implementations. The Crypto Extensions provides the instructions to accelerate the AES, SHA-1 and SHA-256 algorithms. SHA-224 and SHA-256 implementations use the same SHA-256 Crypto Extensions instructions set. 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 using C implementation: "aes-generic"
- AES using ARMv8 with Crypto Extensions: "cbc-aes-ce", "ecb-aes-ce"
- SHA-1 using C implementation: "sha1-generic"
- SHA-1 using ARMv8 with Crypto Extensions: "sha1-ce"
- SHA-224 using C implementation: "sha224-generic"
- SHA-224 using ARMv8 with Crypto Extensions: "sha224-ce"
- SHA-256 using C implementation: "sha256-generic"
- SHA-256 using ARMv8 with Crypto Extensions: "sha256-ce"

The AES, SHA-1 and SHA-256 implementation can also be loaded by simply using the string "aes", "ecb(aes)", "cbc(aes)", "sha1", "sha224" or "sha256" with the initialization call. In this case, the AES, SHA-1 and SHA-256 implementation whose kernel module is loaded with the highest priority is used. In Samsung Kernel Cryptographic Module, implementations supporting the use of ARMv8 with Crypto Extensions are defined to have higher priority than the regular C implementation for AES, SHA-1, SHA-224 and SHA-256. Thus, only one of these implementations can be executed with an initialization call.

Note: The cryptographic module testing performed in this revalidation covers both C implementations and the cryptographic supports from the ARMv8 with Crypto Extensions for AES, SHA-1, SHA-224 and SHA-256.

The module provides the following CAVP validated algorithms which are supported by the ARMv8 with Crypto Extensions:

Algorithms	Standards	CAVS certificates
AES (ECB, CBC mode) encryption and decryption with 128, 192, 256 bits key	FIPS 197	Cert. #3836
AES (GCM mode) encryption and decryption with 128, 192, 256 bits key, IV generated externally	SP 800-38D	Cert. #3836
HMAC with SHA-1, SHA-256	FIPS 198-1	Certs. #2487
HMAC with SHA-224	FIPS 198-1	Certs. #2516
SHA-1, SHA-256	FIPS 180-4	Certs. #3160
SHA-224	FIPS 180-4	Certs. #3193
Hash-based DRBG with SHA-1, SHA-256	SP 800-90A	Cert. #1082
Block Cipher (CTR) based DRBG with AES-128, AES-192, AES-256	SP 800-90A	Cert. #1082

Table 4. CAVP validated algorithms provided by the ARMv8 with Crypto Extensions

Note: The AES GCM mode has been validated by CAVP, but it does not meet the IV generation requirements described in FIPS 140-2 Implementation Guidance (IG) A.5. Thus, it shall only be used in non-FIPS mode.

2.3. Cryptographic Module Boundary

2.3.1. Software Block Diagram

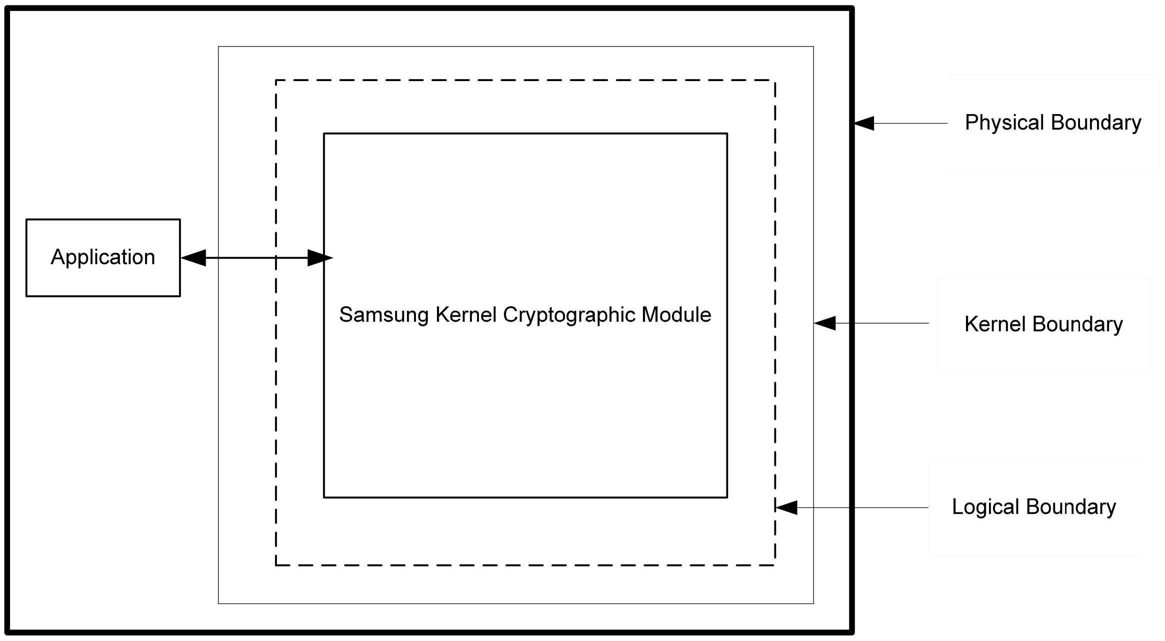

Figure 1: Software Block Diagram

The binary image that contains the Samsung Kernel Cryptographic Module for the appropriate platform is as follows:

- boot.img (version SKC1.7)

Related documentation:

- S/W Detailed Level Design v1.19
- 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 the 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.

Physical boundary

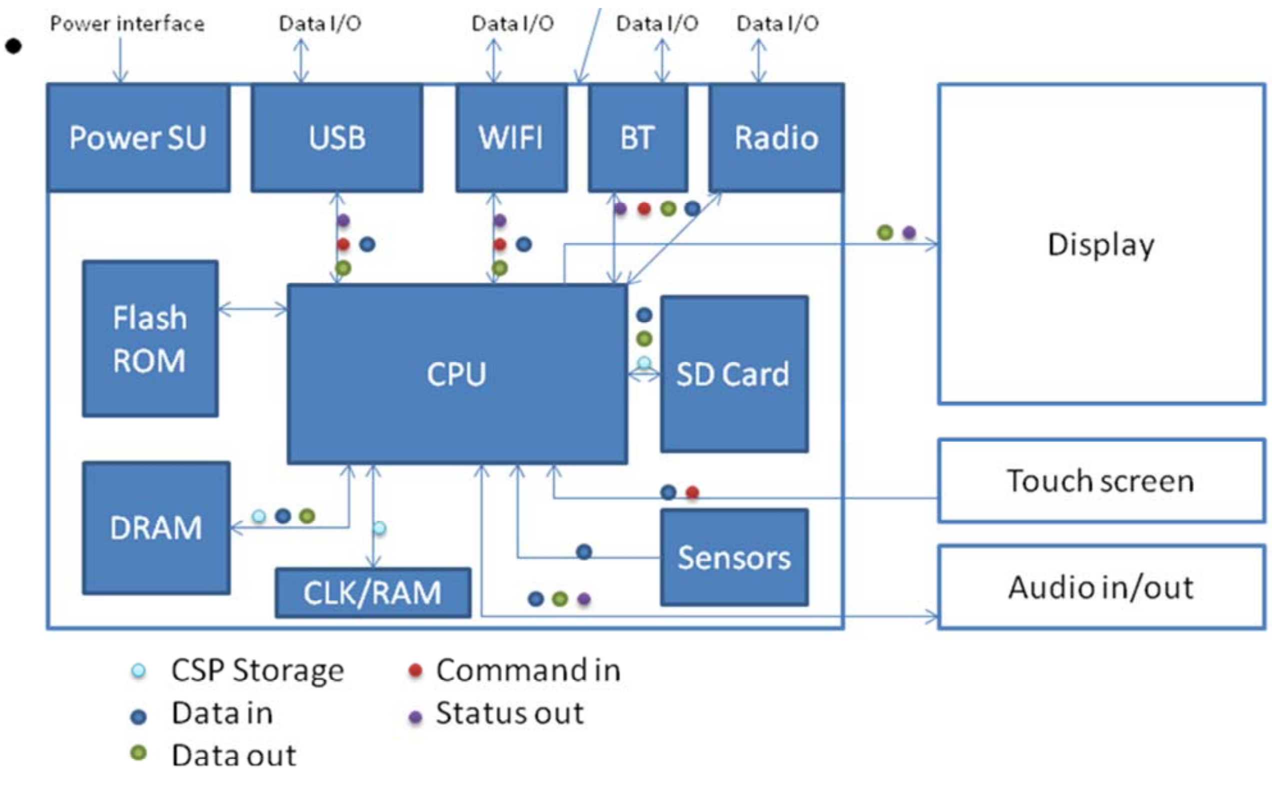

Figure 2: Hardware Block Diagram

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 5. Ports and Interfaces

4. Roles, Services and Authentication

4.1. Roles

Role	Description
User	Perform general security services, including cryptographic operations and other Approved security functions.
Crypto Officer	Perform Initialization of Module.

Table 6. 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 describes the services in FIPS-Approved mode (**Note**: R=Read, W=Write, EX=Execute.)

Role	Services	Algorithms	CSP	Access
User	Symmetric Encryption and Decryption	AES with: • ECB • CBC	128, 192, 256 bits key	R, W, EX
User	Keyed Hash	HMAC with: • SHA-1 • SHA-224 • SHA-256 • SHA-384 • SHA-512	At least 112 bits HMAC key	R, W, EX
User	Message Digest	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	N/A	R, W, EX
User	Random Number Generation with “drbg”	Hash-based DRBG with: • SHA-1 • SHA-256 • SHA-384 • SHA-512	Entropy input string, seed, V and C	R, W, EX
		HMAC-based DRBG with: • SHA-1 • SHA-256 • SHA-384 • SHA-512	Entropy input string, seed, V and Key	R, W, EX
		Block Cipher (CTR) based DRBG with: • AES-128 • AES-192 • AES-256	Entropy input string, seed, V and Key	R, W, EX

Role	Services	Algorithms	CSP	Access
Crypto Officer	Initialization	N/A	N/A	EX
User	Self-Test (Power-up self-tests are executed automatically when device is booted or restarted)	AES, HMAC, SHA and DRBG	344 bits HMAC Key for integrity test	R, EX
User	Check Status/Get State	N/A	N/A	R
User	Zeroization	N/A	AES key, HMAC key, DRBG CSPs	R, W, EX

Table 7. Approved Services

The following table describes the services in non-FIPS mode. Any use of these services with non-Approved algorithms will cause the module to operate in the non-FIPS mode implicitly.

Role	Services	Algorithms	CSP	Access
User	Symmetric Encryption and Decryption	DES	56 bit keys	R, W, EX
		Twofish	128, 192, 256 bits key	R, W, EX
		ARC4	Various key sizes	R, W, EX
		AES GCM mode (AES-GCM)	128, 192, 256 bits key	R, W, EX
		RFC 4106 AES GCM mode (RFC4106-AES-GCM)	128, 192, 256 bits key	R, W, EX
		RFC 4543 AES GCM mode (RFC4543-AES-GCM)	128, 192, 256 bits key	R, W, EX
		AES CTR mode (AES-CTR)	128, 192, 256 bits key	R, W, EX
		Triple-DES (ECB, CBC, CTR)	112, 168 bits key	R, W, EX
		XTS-AES using AES from the ARMv8 with Crypto Extensions	128, 256 bits key	R, W, EX
User	Message Digest	MD5	N/A	R, W, EX
User	Random Number Generation with “krng”	Krng	Seed	R, W, EX
User	Partial Compression and Decompression	Pcompress	N/A	R, W, EX
User	Error Detecting Code	CRC32c	N/A	R, W, EX
User	Data Compression	Deflate	N/A	R, W, EX
		LZO	N/A	R, W, EX
User	Hashing	GHASH	N/A	R, W, EX
User	Multiplication Function	GF128MUL	N/A	R, W, EX

Table 8. Non-Approved Services

Note: The module does not share CSPs between an Approved mode of operation and a non-Approved mode of operation. All cryptographic keys used in the FIPS-Approved must be imported to the module via API input parameters while running in the FIPS-Approved mode. The non-Approved RNG shall only be used for non-Approved services in non-FIPS mode.

The cryptographic module is part of the kernel image. The following documents provide a description and a list of the 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 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	128, 192 and 256 bits 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
SP 800-90A DRBG	Entropy input string	The seed is provided by the /dev/random	N/A	Call the zeroization API function for prng
SP 800-90A DRBG	Seed, V, C and Key	N/A	N/A	Call the zeroization API function for prng
HMAC with SHA-256	344 bits HMAC key for integrity test	N/A	N/A	Not required to be zeroized according to FIPS 140-2 IG 7.4

Table 9. Key Input, Key Output and Key Zeroization for Keys/CSPs

8.1. Random Number Generation

The module employs an Approved SP 800-90A DRBG with different mechanisms (i.e., Hash_DRBG, HMAC_DRBG and CTR_DRBG) for creation of random numbers in FIPS Approved mode. The entropy for seeding the SP 800-90A DRBG is obtained by calling the get_random_bytes() function to collect at least 128 bits of random data from the Linux-provided /dev/random pseudo-device provided by the underlying Operating System. When the DRBG is initialized and seeding for the first time, the module obtains additional entropy from the /dev/random pseudo-device for the nonce.

The following table describes how many bits of data obtained from /dev/random for each DRBG mechanism for instantiate and reseeding:

DRBG Mechanism	Algorithms	Entropy data obtained from /dev/random	
		Instantiate()	Reseed()
Hash_DRBG	SHA-1	192 bits	128 bits
	SHA-256	384 bits	256 bits
	SHA-384	384 bits	256 bits
	SHA-512	384 bits	256 bits
HMAC_DRBG	SHA-1	192 bits	128 bits
	SHA-256	384 bits	256 bits
	SHA-384	384 bits	256 bits
	SHA-512	384 bits	256 bits
CTR_DRBG	128 bits AES	192 bits	128 bits
	192 bits AES	288 bits	192 bits
	256 bits AES	384 bits	256 bits

Table 10. Entropy data obtained from the O/S

CAVEAT: The module generates random strings whose strengths are modified by available entropy.

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 calling application via algorithm APIs. Seeds for random number generation are imported into the module within the physical boundary of 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 the cryptographic module. A pointer to a plaintext key is passed through the algorithm APIs. Intermediate key storages are immediately replaced with 0s in the memory after used.

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: Samsung Electronics EMC Laboratory

FCC Registration: #451343

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-tests use the existing Crypto API tcrypt module to perform integrity test and known-answer self-test for the algorithms. The module is configured as a built-in kernel module instead of a loadable module as in 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. If any self-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 the error state (FIPS_ERR), and Crypto APIs that return cryptographic information are 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 support from ARMv8 with Crypto Extensions
- HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512; with and without SHA-1, SHA-224 or SHA-256 supports from ARMv8 with Crypto Extensions
- SHA-1, SHA-224, SHA-256, SHA-384, SHA-512; with and without SHA-1, SHA-224 or SHA-256 supports from ARMv8 with Crypto Extensions
- SP 800-90A DRBG (Hash_DRBG, HMAC_DRBG and CTR_DRBG)

10.2. Integrity Test

At build time -

- Calculate the HMAC-SHA-256 values of the module code only (available in .text, .rodata, .init.text and .exit.text sections of kernel image) with different offset values which define the kernel position at boot.
- Once generated, the build time HMAC-SHA-256 values of the module code with their associated offset values are updated in the kernel image itself as a read-only data.

At run time –

- Calculate a HMAC-SHA-256 value of the module code only, available in the running kernel on the device (.text, .init.text, .exit.text and .rodata sections of running kernel)
- Retrieve the build time HMAC-SHA-256 value of the module based on the offset value in the current kernel position
- If this calculated run-time HMAC-SHA-256 value is the same as the build time HMAC-SHA-256 value, the integrity test is considered passed. If the calculated and stored values do not match, set error state, FIPS_ERR.

10.3. Conditional Tests

A continuous random number generator test (CRNGT) is performed during each use of the Approved SP 800-90A DRBG. If the values of two consecutive random numbers match, then the cryptographic module goes into error state. A CRNGT is also implemented for the Linux provided /dev/random RNG which is usually used by calling the user for seeding the SP 800-90A DRBG.

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 an internal server. Release is based on the Changelist number, which is automatically 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 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 SKC1.7) 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. Products that do 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 an 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
CMVP	Cryptographic Module Validation Program
CRNGT	Continuous Random Number Generator Test
CSP	Critical Security Parameter
CTR	Counter
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
ECB	Electronic Codebook
EMI	Electromagnetic Interference
EMC	Electromagnetic Compatibility
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
GPC	General Purpose Computer
HMAC	Hash Message Authentication Code
IG	Implementation Guidance
IV	Initial Vector
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
O/S	Operating System
OTA	Over-The-Air
RFC	Request for Comments
RNG	Random Number Generator
SHA	Secure Hash Algorithm

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

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

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