Samsung Kernel Cryptographic Module

Software Versions: 1.6.1, 1.7.1 and 1.8

FIPS 140-2 Non-Proprietary Security Policy

Version 1.20

Last Update: 2017-11-13

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.

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 laboratory
- 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 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	OS & Kernel Versions
Samsung Kernel Cryptographic Module (SKC1.8)	Exynos 8895 with Crypto-Extension	Samsung Galaxy S8	Android 7.0 (Kernel 4.4)
	Exynos 8895 without Crypto-Extension	Samsung Galaxy S8	Android 7.0 (Kernel 4.4)
	MSM 8998 with Crypto-Extension	Samsung Galaxy S8	Android 7.0 (Kernel 4.4)
	MSM 8998 without Crypto-Extension	Samsung Galaxy S8	Android 7.0 (Kernel 4.4)
Samsung Kernel Cryptographic Module (SKC 1.6.1)	MSM 8916	Samsung Galaxy J3	Android 6.0.1 (Kernel 3.10)
Samsung Kernel Cryptographic Module (SKC 1.7.1)	Exynos 7870 with Crypto-Extension	Samsung Tab Active2	Android 7.1 (Kernel 3.18)
	Exynos 7870 without Crypto-Extension	Samsung Tab Active2	Android 7.1 (Kernel 3.18)

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 operational state if the self-tests pass. A kernel proc file is set to indicate if the device is in the operational state or in the error state. The value of the /proc/sys/crypto/fips_status contains 0 if the module is in the operational state; and contains 1 if the module is in the error state (FIPS_ERR). The module is in the FIPS Approved mode only when the module is in the operational state and no non-Approved service is running.

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 (running Approved services) and non-FIPS mode (running non-Approved services).

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

Algorithm	CAVP Cert	Standard	Mode/Method	Key Lengths	Use
SKC 1.6.1					
AES[1]	4403	FIPS 197, SP 800-38A	CBC, ECB	128, 192, 256	Data Encryption / Decryption
HMAC	2926	FIPS 198-1	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	112 minimum	Message Authentication
SHA	3630	FIPS 180-4	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		Message Digest
SKC 1.8 (without Crypto-Extension)					
AES[1]	4424 and 4426	FIPS 197, SP 800-38A	CBC, ECB	128, 192, 256	Data Encryption / Decryption
HMAC	2936 and 2938	FIPS 198-1	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	112 minimum	Message Authentication
SHA	3641 and 3643	FIPS 180-4	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		Message Digest
SKC 1.8 (with Crypto-Extension)					
AES	4425 and 4427	FIPS 197, SP 800-38A	CBC, ECB	128, 192, 256	Data Encryption / Decryption
HMAC	2937 and 2939	FIPS 198-1	SHA-1, SHA-224, SHA-256	112 minimum	Message Authentication

Algorithm	CAVP Cert	Standard	Mode/Method	Key Lengths	Use
SHA	3642 and 3644	FIPS 180-4	SHA-1, SHA-224, SHA-256		Message Digest
SKC 1.7.1 (without Crypto-Extension)					
AES[1]	4744	FIPS 197, SP 800-38A	CBC, ECB	128, 192, 256	Data Encryption / Decryption
HMAC	3159	FIPS 198-1	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	112 minimum	Message Authentication
SHA	3888	FIPS 180-4	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		Message Digest
SKC 1.7.1 (with Crypto-Extension)					
AES	4745	FIPS 197, SP 800-38A	CBC, ECB	128, 192, 256	Data Encryption / Decryption
HMAC	3160	FIPS 198-1	SHA-1, SHA-224, SHA-256	112 minimum	Message Authentication
SHA	3889	FIPS 180-4	SHA-1, SHA-224, SHA-256		Message Digest

Table 3. Approved Algorithms

Note[1]: The AES GCM mode has been validated by CAVP, but it is listed as a non-Approved algorithm 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, SKC 1.8 and SKC 1.7.1, also support the use of AES, SHA-1, SHA-224 and SHA-256 implementations from the Exynos 7870, Exynos 8895 and MSM 8998 with Crypto-Extension. 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 CPU’s Crypto-Extension: “cbc-aes-ce”, “ecb-aes-ce”
- SHA-1 using C implementation: “sha1-generic”
- SHA-1 using CPU’s Crypto-Extension: “sha1-ce”
- SHA-224 using C implementation: “sha224-generic”
- SHA-224 using CPU’s Crypto-Extension: “sha224-ce”
- SHA-256 using C implementation: “sha256-generic”
- SHA-256 using CPU’s Crypto-Extension: “sha256-ce”
- SHA-384 using C implementation: “sha384-generic”
- SHA-512 using C implementation: “sha512-generic”

The AES, SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 implementations can also be loaded by simply using the string “aes”, “ecb(aes)”, “cbc(aes)”, “sha1”, “sha224”, “sha256”, “sha384” or “sha512” with the initialization call. In this case, the AES, SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 implementations whose kernel module is loaded with the highest priority is used. In Samsung Kernel Cryptographic Module, implementations supporting the use of Exynos 7870, Exynos 8895 and MSM 8998 with Crypto-Extension are defined to have higher priority than the regular C implementation. Thus, only one of these implementations can be executed with an initialization call.

Note: The cryptographic module testing performed in this validation for SKC 1.6.1 covers only C implementations because MSM 8916 does not contain Crypto-Extension.

Note: The cryptographic module testing performed in this validation for SKC 1.7.1 and SKC 1.8 covers both C implementations and the cryptographic supports from the Exynos 7870, Exynos 8895 and MSM 8998 with Crypto-Extension for AES, SHA-1, SHA-224 and SHA-256.

The cryptographic module contains the following non-approved algorithms.

Algorithm	Use
NDRNG	Seeding the HMAC_DRBG
HMAC_DRBG (DRBG #1452 and #1453; non-compliant)	Salt and IV Generation for GCM
DES	Symmetric Encryption and Decryption
Twofish	
ARC4	
AES GCM[2] mode (AES-GCM)	
RFC 4106 AES GCM mode (RFC4106-AES-GCM)	
RFC 4543 AES GCM mode (RFC4543-AES-GCM)	
AES CTR mode (AES-CTR)	
Triple-DES (ECB, CBC, CTR)	
XTS-AES using AES from the Exynos or MSM with Crypto-Extension	
MD5	Message Digest
Pcompress	Partial Compression and Decompression
CRC32c	Error Detecting Code
Deflate	Data Compression
LZO	
GHASH	Hashing
GF128MUL	Multiplication Function

Table 4. Non-Approved Algorithms

Note[2]: 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

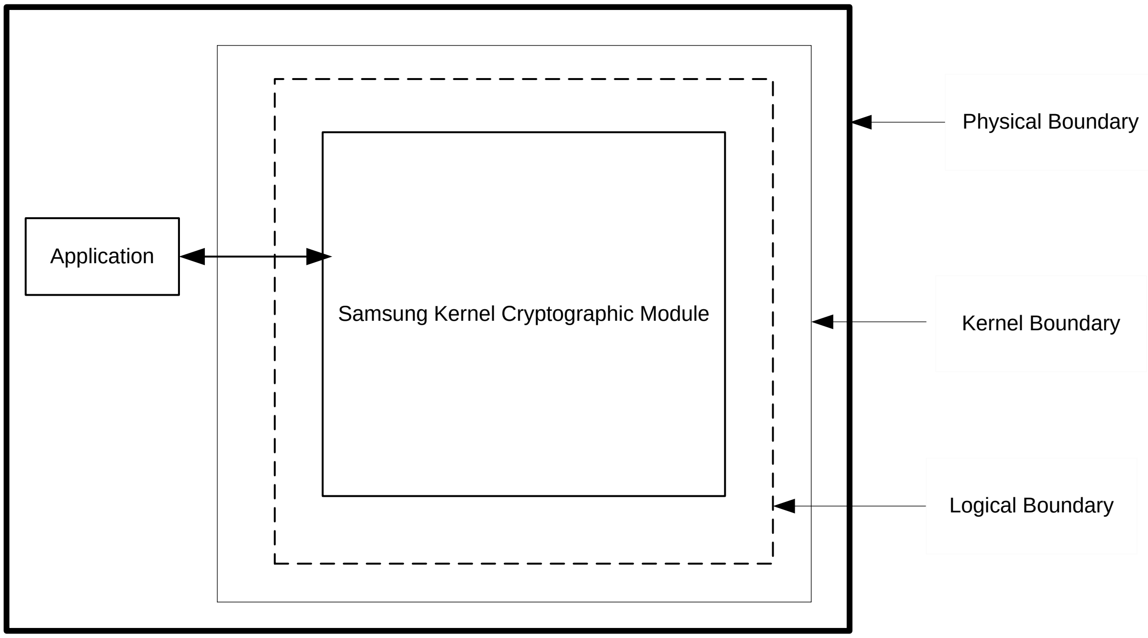

Figure 1: Software Block Diagram

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

- boot.img (version SKC 1.6.1)
- boot.img (version SKC 1.7.1)
- boot.img (version SKC 1.8)

Related documentation:

- Kernel Crypto FIPS Functional Design document
- Samsung Kernel Cryptographic Module FIPS 140-2 Non-Proprietary Security Policy (this 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.

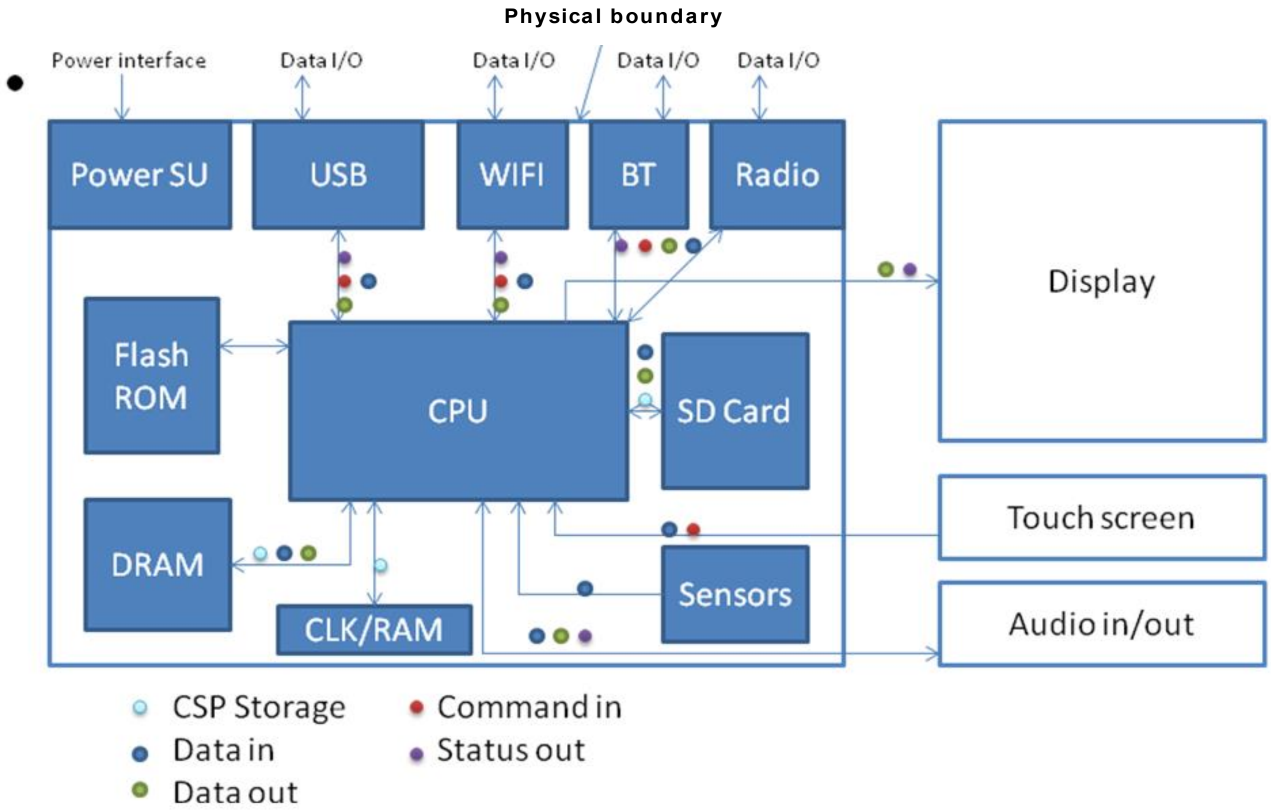

Figure 2: Hardware Block Diagram

3. Cryptographic Module Ports and Interfaces

FIPS Interface	Ports
Data Input	API input parameters
Data Output	API output parameters
Control Input	API control input parameters
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
Crypto Officer	Initialization	N/A	N/A	N/A
User	Symmetric Encryption and Decryption	AES with: • ECB • CBC	AES key (128, 192, 256 bits)	R, W, EX
User	Keyed Hash	HMAC with: • SHA-1 • SHA-224 • SHA-256 • SHA-384 • SHA-512	HMAC key (at least 112 bits)	R, W, EX
User	Message Digest	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	N/A	N/A
User	Self-Test (Power-up self-tests are executed automatically when device is booted or restarted)	AES, HMAC, SHA and DRBG	HMAC Key for integrity test (344 bits)	R, EX
User	Check Status/Get State	N/A	N/A	N/A
User	Zeroization	N/A	AES key, HMAC key	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
User	Symmetric Encryption and Decryption	DES
		Twofish
		ARC4
		AES GCM mode (AES-GCM)
		RFC 4106 AES GCM mode (RFC4106-AES-GCM)
		RFC 4543 AES GCM mode (RFC4543-AES-GCM)
		AES CTR mode (AES-CTR)
		Triple-DES (ECB, CBC, CTR)
		XTS-AES using AES from the Exynos or MSM with Crypto-Extension
User	Salt/IV Generation for AES GCM	Non-Approved HMAC_DRBG
User	Message Digest	MD5
User	Random Number Generation	NDRNG / non-approved HMAC_DRBG
User	Partial Compression and Decompression	Pcompress
User	Error Detecting Code	CRC32c
User	Data Compression	Deflate
		LZO
User	Hashing	GHASH
User	Multiplication Function	GF128MUL

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 Approved services 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 key generation 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

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. Physical Security

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

6. Operational Environment

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

6.1. 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.

7. 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
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

7.1. Random Number Generation

The module employs an NDRNG and a non-approved SP 800-90Ar1 DRBG to provide random data to initialize some system parameters when kernel is loading during device startup. The random data are not used for key generation or output outside the physical boundary when the module is functioning in FIPS-Approved mode.

7.2. 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.

7.3. 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.

7.4. 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.

7.5. 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.

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

The module is a software module that has been tested on the test platforms listed in section 2.1. The test platforms are accepted by the FCC with the following information:

Lab Name: Samsung Electronics EMC Laboratory

FCC Registration: #451343

The test devices which run the modules conform 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).

FCC ID: A3LSMJ327A (for Samsung Galaxy J3 with MSM 8916)
FCC ID: A3LSMG955F (for Samsung Galaxy S8 with Exynos 8895)
FCC ID: A3LSMG955U (for Samsung Galaxy S8 with MSM 8998)
FCC ID: A3LSMT395 (for Samsung Galaxy Tab Active2 with Exynos 7870)

9. Self-Tests

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.

9.1. 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, version 1.6.1, implements each of the following Known Answer Tests:

- AES encryption and decryption tested separately
- HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512
- SHA-1, SHA-224, SHA-256, SHA-384, SHA-512
- SP 800-90A HMAC_DRBG (includes SP 800-90A Section 11.3 Health Checks)

The modules, version 1.7.1 and version 1.8, implement each of the following Known Answer Tests:

- AES encryption and decryption tested separately; with support from CPU Crypto-Extension
- AES encryption and decryption tested separately; without support from CPU Crypto-Extension
- HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256; with SHA-1, SHA-224 and SHA-256 support from CPU Crypto-Extension
- HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512; without support from CPU Crypto-Extension
- SHA-1, SHA-224, SHA-256; with support from CPU Crypto-Extension
- SHA-1, SHA-224, SHA-256, SHA-384, SHA-512; without support from CPU Crypto-Extension
- SP 800-90A HMAC_DRBG (includes SP 800-90A Section 11.3 Health Checks)

9.2. Integrity Test

At build time -

- Calculate the HMAC-SHA-256 values of the module code only (available in .text, .rodata, .init.text, asm.text, asm.rodata and asm.init.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, asm.text, asm.rodata and asm.init.text 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.

9.3. Conditional Tests

A continuous random number generator test (CRNGT) is performed during each use of the non-Approved SP 800-90Ar1 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 NDRNG (/dev/urandom) which is used for seeding the non-approved SP 800-90Ar1 DRBG.

10. Design Assurance

10.1. 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, document version number 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 available in the Samsung internal Perforce repository, as listed in Functional Design document, is used to build target binary.

10.2. 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.

Samsung 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.

11. Mitigation of Other Attacks

No other attacks are mitigated.

12. 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

13. References

[1] FIPS 140-2 Standard, https://csrc.nist.gov/publications/detail/fips/140/2/final

[2] FIPS 140-2 Implementation Guidance, https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Module-Validation-Program/documents/fips140-2/FIPS1402IG.pdf

[3] FIPS 140-2 Derived Test Requirements, https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Module-Validation-Program/documents/fips140-2/FIPS1402DTR.pdf

[4] FIPS 197 Advanced Encryption Standard, https://csrc.nist.gov/publications/detail/fips/197/final

[5] FIPS 180-4 Secure Hash Standard, https://csrc.nist.gov/publications/detail/fips/180/4/final

[6] FIPS 198-1 The Keyed-Hash Message Authentication Code (HMAC), https://csrc.nist.gov/publications/detail/fips/198/1/final

[7] SP 800-67 Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, https://doi.org/10.6028/NIST.SP.800-67r1

[8] SP 800-38D Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, https://doi.org/10.6028/NIST.SP.800-38D

[9] SP 800-90Ar1 Recommendation for Random Number Generation Using Deterministic Random Bit Generators, https://doi.org/10.6028/NIST.SP.800-90Ar1