Samsung SCrypto Cryptographic Module

Software Version: 2.5

FIPS 140-2 Non-Proprietary Security Policy

Document Version: 3.0

Last Update: 2023-7-12

Prepared by:
Gossamer Security Solutions
9176 Red Branch Road, Suite L
Columbia, MD 21045
www.gossamersec.com

Table of Contents

List of Tables

1 Introduction

This document is a non-proprietary FIPS 140-2 Security Policy for the Samsung SCrypto Cryptographic Module hereafter referred to as the module. The module is a software library providing a C-language application program interface (API) for use by other processes that require cryptographic functionality. The module is classified by FIPS 140-2 (Federal Information Processing Standards Publication 140-2) as a Security Level 1 multi-chip standalone software module. The physical cryptographic boundary is the general purpose computer on which the module is installed. The logical cryptographic boundary of the module is the SCrypto cryptographic module, a single object module file named libscrypto.so[1] on the tested platform running TEEGRIS 4.0/4.1(32-bit or 64-bit) or libscrypto.a[2] on the tested platform running QSEE 5.8/5.10/5.11 (32-bit or 64-bit). Please refer to Table 2 for more information. The module performs no communications other than with the calling application (the process that invokes the module services). The module's software version for this validation is 2.5.

2 Cryptographic Module Specification

2.1 Description of Module

The Samsung SCrypto Cryptographic Module is a software only security level 1 cryptographic module that provides general-purpose cryptographic services. 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
Overall Level	**1**

Table 1. Security Levels

[1] The single object module file (libscrypto.so) compiled for the tested platform running TEEGRIS 4.0/4.1 (32-bit) is different from the one running TEEGRIS 4.0/4.1 (64-bit).
[2] The single object module file (libscrypto.a) compiled for the tested platform running QSEE 5.8/5.10/5.11 (32-bit) is different from the one running QSEE 5.8/5.10/5.11 (64-bit).

The module has been tested on the following platforms:

#	Operating System	Processor	Platform
1	TEEGRIS 4.1 (32-bit)	Samsung Electronics Exynos 990	Samsung Galaxy S20+
2	TEEGRIS 4.1 (64-bit)	Samsung Electronics Exynos 990	Samsung Galaxy S20+
3	QSEE 5.8 (32-bit)	Qualcomm SM8250	Samsung Galaxy S20+
4	QSEE 5.8 (64-bit)	Qualcomm SM8250	Samsung Galaxy S20+
5	QSEE 5.10 (32-bit)	Qualcomm SM7250	Samsung Galaxy A71
6	QSEE 5.10 (64-bit)	Qualcomm SM7250	Samsung Galaxy A71
7	TEEGRIS 4.0 (32-bit)	Samsung Electronics Exynos 9810	Samsung Galaxy Tab Active3
8	TEEGRIS 4.0 (64-bit)	Samsung Electronics Exynos 9810	Samsung Galaxy Tab Active3
9	QSEE 5.10 (32-bit)	Qualcomm Snapdragon 750	Samsung Galaxy A42
10	QSEE 5.10 (64-bit)	Qualcomm Snapdragon 750	Samsung Galaxy A42
11	QSEE 5.11 (32-bit)	Qualcomm Snapdragon 888	Samsung Galaxy S21+
12	QSEE 5.11 (64-bit)	Qualcomm Snapdragon 888	Samsung Galaxy S21+
13	QSEE 5.11 (32-bit)	Qualcomm Snapdragon 778G	Samsung Galaxy XCover6 Pro
14	QSEE 5.11 (64-bit)	Qualcomm Snapdragon 778G	Samsung Galaxy XCover6 Pro

Table 2. Tested Platforms

Notes:

- Please note that this validation tested both the 64-bit and 32-bit versions of the module on the corresponding platforms.
- Per 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 Operational State

When the module is initialized, the self-tests are executed automatically at load time and the module enters its operational state automatically if the self-tests pass. A status function is set to indicate if the device is in an operational state or in an error state. The error state flag is used for the value of the function FIPS_status() from /src/fips/ fips_manager.c. A calling application can check the module's status by calling the FIPS_status() function which returns 1 if in the operational state and returns 0 if in the error state. The module supports both FIPS approved and non-FIPS approved algorithms. Please refer to section 13 for instructions to operate the module in a FIPS approved manner.

2.3 Cryptographic Module Boundary

2.3.1 Software Block Diagram

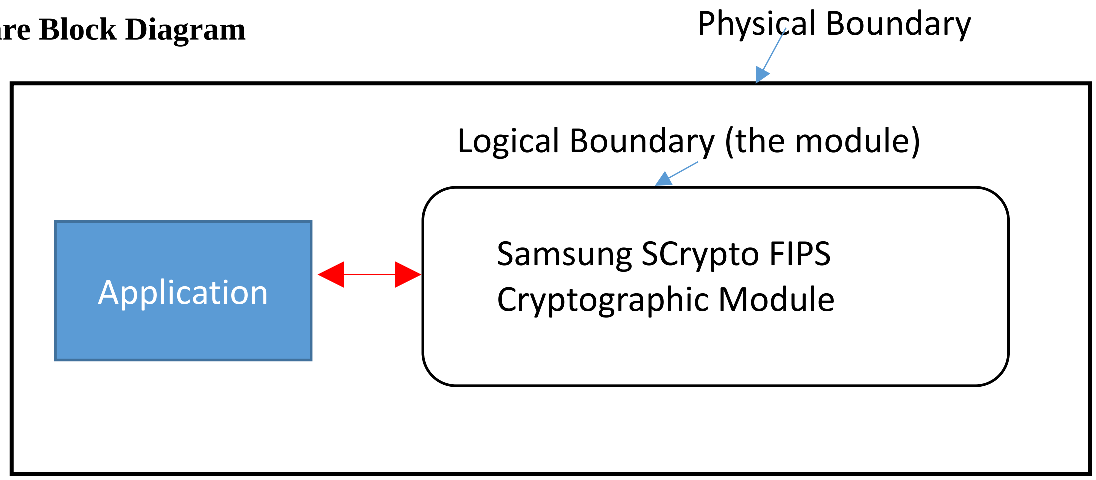

Figure 1: Software Block Diagram

3 Cryptographic Module Ports and Interfaces

The module runs on a General Purpose Computer (GPC). The Physical Cryptographic Boundary for the module is the case of that GPC. All the physical components are standard electronic components; there are not any custom integrated circuits or components dedicated to FIPS 140-2 functionality.

FIPS Interface	Ports
Data Input	API input parameters
Data Output	API output parameters
Control Input	API function calls
Status Output	API return codes, API output parameters for status
Power Input	Physical power connector

Table 3. Ports and Interfaces

The Data Input interface consists of the input parameters of the API functions and data received through the I/O system calls. The Data Output interface consists of the output parameters of the API functions and the data sent through the I/O system calls. The Control Input interface consists of the API function calls. The Status Output interface includes the return values of the API functions and status sent through output parameters.

4 Roles, Services and Authentication

4.1 Roles

The module supports the Crypto Officer (CO) role and User role, which meets all FIPS 140-2 level 1 requirements for Roles and Services. The module does not support a Maintenance role. The User and Crypto Officer roles are implicitly assumed by the entity accessing services implemented by the module. No further authentication is required. The module does not allow concurrent operators.

4.2 Services

4.2.1 Services Available in FIPS Mode of Operation

All services implemented by the module are listed below, along with a description of service CSP access.

Service	Role	Description/CSP	Access Permission
Initialize	User, CO	Module initialization. **CSPs**: None	N/A
Self-test	User, CO	Perform self tests (FIPS_SCRYPTO_selftest). **CSPs**: None	N/A
Show status	User, CO	Functions that provide module status information. **CSPs**: None	N/A
Zeroize	User, CO	Functions that destroy all CSPs.	read/write/execute
Random number generation	User, CO	Used for random number and symmetric key generation. **CSPs**: Entropy input string, DRBG seed, DRBG V and DRBG Key	read/write/execute
Asymmetric key generation	User, CO	Used to generate Asymmetric keys. **CSPs**: RSA SGK, RSA SVK, ECDSA SGK, ECDSA SVK	read/write/execute
Symmetric encrypt/decrypt	User, CO	Used to encrypt or decrypt data. **CSPs**: AES EDK, AES-GCM key	read/write/execute
Symmetric digest	User, CO	Used to generate or verify data integrity with CMAC. **CSPs**: AES CMAC Key	read/write/execute
Message digest	User, CO	Used to generate a SHA-1 or SHA-2 message digest. **CSPs**: None	read/write/execute
Keyed Hash	User, CO	Used to generate or verify data integrity with HMAC. **CSP**: HMAC Key	read/write/execute
Key wrapping	User, CO	Used to encrypt or decrypt a key value on behalf of the calling process (does not establish keys into the module). **CSPs**: Key Wrap, RSA KDK and RSA KEK	read/write/execute
Digital signature	User, CO	Used to generate or verify RSA or ECDSA digital signatures. **CSPs**: RSA SGK, RSA SVK, ECDSA SGK, ECDSA SVK	read/write/execute

Service	Role	Description/CSP	Access Permission
Key derivation	User, CO	Used to derive other keying material via SP800-108 KDF **CSP:** Key Derivation Key	read/write/execute
Utility	User, CO	Miscellaneous helper functions. **CSPs**: None	N/A

Table 4. Approved/Allowed Services and CSP Access

The following document provides a description and API functions of the cryptographic services listed above:

- FIPS_SCrypto_Functional_Design, a Samsung internal document

4.2.2 Non-Approved/Compliant Services

In addition to the above listed FIPS-Approved/Allowed services, the cryptographic module also provides non-Approved services; however, any use of the module's non-Approved/Compliant services causes the module to operate in a non-FIPS compliant manner. Thus, operators concerned with FIPS compliance should not utilize any of the following non-Approved service(s):

- Triple-DES: API functions DES_set_key() and DES_ede3_cbc_encrypt()
- DSA: API functions DSA_do_sign() and DSA_do_verify()
- Diffie-Hellman Key Agreement: API Function DH_generate_key()
- EC Diffie-Hellman Key Agreement: API function ECDH_compute_key()

Please note that prior to using any of the Non-Approved/Compliant services listed above, the CO must zeroize all CSPs. Likewise, to put the module back into the FIPS mode from the non-FIPS mode, the CO must zeroize all Keys/CSPs used in non-FIPS mode, and then strictly follow the instructions in section 13.1 of this document to put the module into the FIPS mode

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 Samsung's internal FIPS_SCrypto_Functional_Design document.

6 Physical Security

The module is a software entity 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. 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 module, even when the application is serving multiple clients.

8 Cryptographic Algorithms

8.1 Approved Cryptographic Algorithms

The module implemented the following CAVP validated algorithms.

Algorithms	Modes and Description	API Function	FIPS Approved (Cert #)
AES	ECB (e/d; 128, 192, 256); CBC (e/d; 128, 192, 256); OFB (e/d; 128, 256); CTR (ext only; 128, 192, 256) KW (Direction: Decrypt, Encrypt; Cipher: Cipher Key Length: 128, 192, 256; Payload Length: 128, 256, 320)	AES_set_encrypt_key AES_set_decrypt_key AES_encrypt AES_decrypt AES_ecb_encrypt AES_cbc_encrypt AES_ctr128_encrypt AES_ofb128_encrypt AES_wrap_key AES_unwrap_key	Certs. #A889 and #C1360
AES-CMAC	CMAC (Generation/Verification, 128, 192, 256)	CMAC_Init CMAC_Update CMAC_Final CMAC_Reset	Certs. #A889 and #C1360
AES-GCM	GCM (e/d), (128, 192 or 256)	CRYPTO_gcm128_init CRYPTO_gcm128_setiv CRYPTO_gcm128_aad CRYPTO_gcm128_encrypt CRYPTO_gcm128_decrypt CRYPTO_gcm128_encrypt_ctr32 CRYPTO_gcm128_decrypt_ctr32	Certs. #A889 and #C1360

Algorithms	Modes and Description	API Function	FIPS Approved (Cert #)
FIPS 186-4 ECDSA	PKG: CURVES (P-224 P-256 P-384 P-521 ExtraRandomBits); PKV: CURVES (P-224 P-256 P-384 P-521) SigGen: CURVES (P-224, P-256, P-384, P-521) with SHA2-224, SHA2-256, SHA2-384, SHA2-512 SigVer: CURVES (P-224, P-256, P-384, P-521) with SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	EC_KEY_generate_key ECDSA_do_sign ECDSA_do_verify ECDSA_sign ECDSA_verify	Certs. #A889 and #C1360
SP800-90A DRBG	Block Cipher (CTR) based DRBG with AES-256	CTR_DRBG_init CTR_DRBG_reseed CTR_DRBG_generate CTR_DRBG_clear	Certs. #A889 and #C1360
HMAC	HMAC with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	HMAC_Init_ex HMAC_Update HMAC_Final	Certs. #A889 and #C1360
SHA	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	SHA1_Init SHA1_Update SHA1_Final SHA1 SHA224_Init SHA224_Update SHA224_Final SHA224 SHA256_Init SHA256_Update SHA256_Final SHA256 SHA384_Init SHA384_Update SHA384_Final SHA384 SHA512_Init SHA512_Update SHA512_Final SHA512	Certs. #A889 and #C1360

Algorithms	Modes and Description	API Function	FIPS Approved (Cert #)
FIPS 186-4 RSA	FIPS186-4: 186-4KEY(gen): Key Generation Mode: B.3.3, Modulo: 2048, Primality Tests: C.2; Modulo: 3072, Primality Tests: C.2. Public Exponent Mode: Fixed; Fixed Public Exponent: 10001; Private Key Format: Standard [RSASSA-PKCS1_V1_5] SIG(gen) (2048/3072) with SHA-1, SHA2-224, SHA2-256, SHA2-384 and SHA2-512; SIG(Ver) (1024/2048/3072) with SHA-1, SHA2-224, SHA2-256, SHA2-384 and SHA2-512. [RSASSA-PSS]: SIG(gen) (2048/3072) with SHA-1, SHA2-224, SHA2-256, SHA2-384 and SHA2-512; SIG(Ver) (1024/2048/3072) with SHA-1, SHA2-224, SHA2-256, SHA2-384 and SHA2-512	RSA_sign RSA_verify RSA_private_encrypt RSA_public_decrypt RSA_generate_key_fips	Certs. #A889 and #C1360
KBKDF (SP800-108)	KDF Mode: Counter; MAC Mode: HMAC-SHA2-512; Fixed Data Order: After Fixed Data, Before Fixed Data, In the Middle of Fixed DataCounter Length: 8, 16, 24, 32; Supported Lengths: 512, 4096	NIST_KDF_HMAC_SHA512_Ctr_Mode	Certs. #A889 and #C1360
RSADP Primitive	RSADP: (Mod 2048)	RSA_decrypt	CVL Certs. #A889 and #C1360
CKG (vendor affirmed)	Vendor affirmed Cryptographic Key Generation	EC_KEY_generate_key_fips RSA_generate_key_fips	

Table 5. CAVP validated algorithms

Notes:

- Cert. #C1360 from Table 5 above was tested for the OEs (lines 1-8) listed in Table 2, whereas Cert. #A889 was tested for the OEs (lines 9-14) listed in Table 2.
- There are some algorithm modes/key lengths that were tested but not used by the module. Only the algorithms, modes, and key sizes that are implemented by the module are listed in this table.
- The AES-GCM IV generation method from each of AES Certs. #A889 and #C1360 is in compliance with IG A.5, scenario #2. The DRBG from each of DRBG Certs. #A889 and #C1360 is called to generate the IV inside the module, and the IV length is 96 bits. The module generates new AES-GCM keys if the module loses power.
- KTS (AES Certs. #A889 and #C1360; key establishment methodology provides between 128 and 256 bits of encryption strength).
- In accordance with FIPS 140-2 IG D.12, the cryptographic module performs Cryptographic Key Generation as per scenario 1 of section 5 in SP800-133r2. The resulting generated symmetric key and the seed used in the asymmetric key generation are the unmodified output from SP800-90A DRBG.

In addition, the API functions listed in Table 5 were obtained from a Samsung provided source code document and the FIPS_SCrypto_Functional_Design document.

8.2 Non-FIPS Approved (but Allowed) Cryptographic Algorithms

In addition, the module supports the following Non-Approved but Allowed algorithms that can be used in FIPS-mode:

Algorithm	Notes
RSA (encrypt, decrypt)	The RSA algorithm may be used by the calling application for encryption or decryption of keys. No claim is made for SP 800-56B compliance, and no CSPs are established into or exported out of the module using these services.
NDRNG	Non-Approved RNG. Used to seed FIPS approved SP800-90A DRBG. The NDRNG is implemented by the underlying operating system (and not by the module) which is outside its logical boundary.

Table 6. Non-Approved (but Allowed) Algorithms

Caveats:

- RSA: CVL Certs. #A889 and #C1360, key wrapping; key establishment methodology provides 112 or 128 bits of encryption strength.

8.3 Non-FIPS Allowed Cryptographic Algorithm

The cryptographic module implements the following non-FIPS Allowed algorithms that are not permitted for use while operating in a FIPS 140-2 compliant fashion:

- Triple-DES (non-compliant)
- DSA (non-compliant)
- Diffie-Hellman Key Agreement (non-compliant)
- EC Diffie-Hellman Key Agreement (non-compliant)

9 Cryptographic Key Management

All **CSPs** used by the Module are described in this section. All access to these CSPs by Module services are described in Section 4. The CSP names are generic, corresponding to API parameter data structures.

CSP Name	Description	Generated/Input	Output
RSA SGK	RSA (2048/3072 bits) signature generation key	Internally Generated or Input from the calling application via the Module's API in plaintext	Output to the calling application via the Module's API in plaintext
RSA KDK	RSA (2048 bits) key decryption (private key transport) key	Internally Generated or Input from the calling application via the Module's API in plaintext	Output to the calling application via the Module's API in plaintext
ECDSA SGK	FIPS 186-4 ECDSA (P-224/P-256/P-384/P-521 Curves) signature generation key	Internally Generated or Input from the calling application via the Module's API in plaintext	Output to the calling application via the Module's API in plaintext
AES EDK	AES (128/192/256 bits) encrypt / decrypt key	Input via the Module's API in plaintext	N/A
AES CMAC Key	AES (128/192/256 bits) CMAC generate / verify key	Input via the Module's API in plaintext	N/A
AES GCM Key	AES (128/192/256 bits) encrypt / decrypt / generate / verify key	Input via the Module's API in plaintext	N/A
HMAC Key	Keyed hash key (160/224/256/384/512 bits)	Input via the Module's API in plaintext	N/A
SP800-90A CTR_DRBG CSPs	V (128 bits), Seed (256/320/384 bits) and Key (AES 128/192/256 bits), Entropy input (384 bits from hardware entropy noise source)	Internally Generated per SP800-90A DRBG	N/A
Key Wrap	AES Key Wrap using 128, 192 or 256-bit keys	Input via the Module's API in plaintext	N/A
Key Derivation Key	SP800-108 KBKDF Key Derivation Key (512 bits)	Input via the Module's API in plaintext	N/A

Table 7. Critical Security Parameters

Notes:

- Please refer to Section 9.2 for Key Generation
- Please refer to Section 9.3 for Key Entry and Output

Below is the table listing all public keys used within the module.

CSP Name	Description
RSA SVK	RSA (1024/2048/3072 bits) signature verification public key. Internally Generated or Input from the calling application via the Module's API in plaintext; Output to the calling application via the Module's API in plaintext.
RSA KEK	RSA (2048 bits) key encryption (public key Transport) key. Internally Generated or Input from the calling application via the Module's API in plaintext; Output to the calling application via the Module's API in plaintext.
ECDSA SVK	ECDSA (P-224/P-256/P-384/P-521 Curves) signature verification key. Internally Generated or Input from the calling application via the Module's API in plaintext; Output to the calling application via the Module's API in plaintext.

Table 8. Public Keys

9.1 Random Number Generation

The module employs an Approved SP 800-90A CTR_DRBG for creation of random numbers. The module uses NDRNG from the operational environment as the source of random numbers for DRBG seeds. The NDRNG produces the random numbers from an entropy pool maintained by the underlying Operating System. By default, the module gets the entropy input from the hardware random source which is residing on each tested platform. The minimum strength of DRBG seed is 256 bits.

The Module performs a Continuous Random Number Generation Test (CRNGT) on the random seed used to instantiate the module's DRBG.

9.2 Key Generation

The module performs Cryptographic Key Generation (CKG) for asymmetric keys as per scenario 1 of section 5 in SP800-133r2 (vendor affirmed), and uses DRBG compliant with [SP800-90A]. The resulting generated seed used in the asymmetric keys (RSA SGK, RSA KDK and ECDSA SGK listed in Table 7) generation are the unmodified output from SP800-90A DRBG. Again, the module draws its DRBG seeds from hardware noise source.

9.3 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 Module API calls within the tested platform's physical boundary. The module does not output intermediate key generation values.

9.4 Key Storage

Keys are not stored inside the cryptographic module. A pointer to a plaintext key is passed through the algorithm APIs. Intermediate keys stored in the module's memory are immediately replaced with 0s in the memory after use. Keys residing in internally allocated data structures (during the lifetime of an API call) can only be accessed using the module defined API. The operating system protects memory and process space from unauthorized access. Only the calling application that creates or imports keys can use or export such keys. All API functions are executed by the invoking calling application in a non-overlapping sequence such that no two API functions will execute concurrently.

9.5 Zeroization

The zeroization mechanism for all of the CSPs is to replace 0s in the memory which originally stored the CSPs. Zeroization of sensitive data is performed automatically by calling zeroization API function OPENSSL_cleanse() for temporarily stored CSPs. In addition, the module provides functions to explicitly destroy CSPs related to random number generation services. The calling application is responsible for parameters passed in and out of the module.

10 Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

Lab Name: CS & Environment Center of Samsung Electronics Co., Ltd.

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).

FCC ID for each Tested Platform:

Device	Processor	Operating System	FCC ID
Samsung Galaxy S20+	Samsung Electronics Exynos 990	TEEGRIS 4.1 (32-bit)	A3LSMG986B
Samsung Galaxy S20+	Samsung Electronics Exynos 990	TEEGRIS 4.1 (64-bit)	A3LSMG986B
Samsung Galaxy S20+	Qualcomm SM8250	QSEE 5.8 (32-bit)	A3LSMG986U
Samsung Galaxy S20+	Qualcomm SM8250	QSEE 5.8 (64-bit)	A3LSMG986U
Samsung Galaxy A71	Qualcomm SM7250	QSEE 5.10 (32-bit)	A3LSMA716U
Samsung Galaxy A71	Qualcomm SM7250	QSEE 5.10 (64-bit)	A3LSMA716U
Samsung Galaxy Tab Active3	Samsung Electronics Exynos 9810	TEEGRIS 4.0 (32-bit)	A3LSMT575
Samsung Galaxy Tab Active3	Samsung Electronics Exynos 9810	TEEGRIS 4.0 (64-bit)	A3LSMT575
Samsung Galaxy A42	Qualcomm Snapdragon 750	QSEE 5.10 (32-bit)	A3LSMA426B
Samsung Galaxy A42	Qualcomm Snapdragon 750	QSEE 5.10 (64-bit)	A3LSMA426B
Samsung Galaxy S21+	Qualcomm Snapdragon 888	QSEE 5.11 (32-bit)	A3LSMG996U
Samsung Galaxy S21+	Qualcomm Snapdragon 888	QSEE 5.11 (64-bit)	A3LSMG996U
Samsung Galaxy XCover6 Pro	Qualcomm Snapdragon 778G	QSEE 5.11 (32-bit)	A3LSMG736B
Samsung Galaxy XCover6 Pro	Qualcomm Snapdragon 778G	QSEE 5.11 (64-bit)	A3LSMG736B

Table 9. FCC IDs

For information related to the FCC ID of the devices, please refer to Samsung's internal FIPS_SCrypto_Functional_Design document.

11 Self-Tests

The module performs a series of power-up self-tests that covers all of its FIPS-approved algorithms. The module executes all self-tests when the module is initialized during the boot process. Self-tests can also be manually invoked by calling FIPS_SCRYPTO_post(1). When the module passes all of its power-up self-tests, the module sets an internal variable to reflect this. A calling application can call the FIPS_status() API to obtain the value of this internal variable (1 if the self-tests have passed, and 0 if a FIPS error has occurred). In addition to Known Answer Tests (KATs) for each of the module's cryptographic algorithms, the module also performs a binary integrity test to check for corruption. If any KAT self-test or the integrity test fails, the module sets its error flag (static variable), returns an error code to the API function caller to indicate the error, enters an error state (FIPS_ERR), and inhibits Crypto APIs that return cryptographic information.

11.1 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 do 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 Power On Self-Tests:

- AES encryption and decryption Known Answer Tests (KATs)
- AES-CMAC KAT
- AES-GCM KAT
- AES-KW KAT
- ECDSA (sign and verify) Power On Self-Test
- SP800-108 KDF (KBKDF) KAT
- RSA KATs (separate KAT for signing; separate KAT for verification)
- SP 800-90A CTR_DRBG KAT (Note: DRBG Health Tests as specified in SP800-90A Section 11.3 are performed)
- HMAC KATs (HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512)

11.2 Software Integrity Check

The integrity of the module is verified by comparing an HMAC-SHA-256 value calculated at run time with the value stored in the module that was computed at build time.

11.3 Conditional Tests

The Module also implements the following conditional tests:

- NDRNG Continuous Random Number Test
- RSA Pairwise Consistency Test (PWCT)
- ECDSA Pairwise Consistency Test (PWCT)

12 Mitigation of Other Attacks

No other attacks are mitigated.

13 Secure Operation

13.1 Crypto Officer/User Guidance

The module is provided directly to solution developers and is not available for direct download to the general public. The module and its host application are to be installed on an operating system specified in Section 2.1 or one where portability is maintained.

Additional Rules of Operation:

1. The writable memory areas of the module (data and stack segments) are accessible only by the application so that the operating system is in "single user" mode, i.e. only the application has access to that instance of the module.
2. The operating system is responsible for multitasking operations so that other processes cannot access the address space of the process containing the module.
3. Only the services defined in Table 4 shall be used in FIPS mode of operation.

13.2 General Guidance

The module is not distributed as a standalone library and is only used in conjunction with the solution.

The end user of the operating system is also responsible for zeroizing CSPs via wipe/secure delete procedures. If the module power is lost and restored, the calling application can reset the IV to the last value used.

14 Design Assurance

14.1 Configuration Management

Perforce is used as the repository for both source code and documents. All source code and documents are maintained on 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 versions 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 the Functional Design document, is used to build target binary.

14.2 Delivery and Distribution

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

The vendor distributes the module as part of the larger system firmware images specific to each phone model and carrier or as part of mobile applications. Only authorized personnel can register the module binary with the vendor's automated manufacturing system, so that it can be loaded (as part of a larger image) onto newly manufactured phone hardware without any manual intervention or (for mobile applications) distributed through a secure method (for example, the Google Play Store). Employees are not allowed to bring in any personal belongings to the manufacturing facility and entrance to the facility is controlled through employee ID based badge access and monitored using CCTV. SAMSUNG only releases the module binary in image form and through OTA (Over the Air) mechanisms, the latter of which is controlled by SAMSUNG or mobile carriers. In either case, if the binary were modified by an unauthorized entity, the device would detect the modification and not accept the modified binary.

15 Glossary and Abbreviations

AES	Advanced Encryption Specification
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCTV	Closed Caption Television
CMT	Cryptographic Module Testing
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
FCC	Federal Communications Commission
FSM	Finite State Model
GCM	Galois/Counter Mode
HMAC	Hash Message Authentication Code
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
O/S	Operating System
NDRNG	Non-Deterministic Random Number Generator
SHA	Secure Hash Algorithm

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

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

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

[11] SP 800-131A Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths, http://csrc.nist.gov/publications/PubsSPs.html