



Samsung SCrypto Cryptographic Module

Software Version: 2.2

FIPS 140-2 Non-Proprietary Security Policy

Document Version: 1.2

Last Update: 2018-6-12

Prepared by:

Gossamer Security Solutions

1352 N Rolling Rd

Catonsville, MD 21228

www.gossamersec.com

Table of Contents

| | | |
|-----------|-----------------------------------------------------------------------------|----|
| 1 | Introduction | 4 |
| 2 | Cryptographic Module Specification | 4 |
| 2.1 | Description of Module | 4 |
| 2.2 | Description of Operational State | 5 |
| 2.3 | Cryptographic Module Boundary | 6 |
| 2.3.1 | Software Block Diagram | 6 |
| 3 | Cryptographic Module Ports and Interfaces | 6 |
| 4 | Roles, Services and Authentication | 7 |
| 4.1 | Roles | 7 |
| 4.2 | Services | 7 |
| 4.2.1 | Services Available in FIPS Mode of Operation | 7 |
| 4.2.2 | Non-Approved/Compliant Services | 8 |
| 4.3 | Operator Authentication | 8 |
| 4.4 | Mechanism and Strength of Authentication | 8 |
| 5 | Finite State Machine | 8 |
| 6 | Physical Security | 9 |
| 7 | Operational Environment | 9 |
| 8 | Cryptographic Algorithms | 9 |
| 8.1 | Approved Cryptographic Algorithms | 9 |
| 8.2 | Non-FIPS Approved (but Allowed) Cryptographic Algorithms | 12 |
| 8.3 | Non-FIPS Allowed Cryptographic Algorithm | 12 |
| 9 | Cryptographic Key Management | 13 |
| 9.1 | Random Number Generation | 14 |
| 9.2 | Key Generation | 14 |
| 9.3 | Key Entry and Output | 14 |
| 9.4 | Key Storage | 14 |
| 9.5 | Zeroization | 14 |
| 10 | Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC) | 15 |
| 11 | Self-Tests | 15 |
| 11.1 | Power-Up Tests | 15 |
| 11.2 | Software Integrity Check | 16 |
| 11.3 | Conditional Tests | 16 |
| 12 | Mitigation of Other Attacks | 16 |
| 13 | Secure Operation | 16 |

| | | |
|------|----------------------------------|----|
| 13.1 | Crypto Officer Guidance | 16 |
| 13.2 | General Guidance | 16 |
| 14 | Design Assurance | 17 |
| 14.1 | Configuration Management | 17 |
| 14.2 | Delivery and Distribution | 17 |
| 15 | Glossary and Abbreviations | 18 |
| 16 | References | 18 |

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.a*. 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 2.2.

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

The module has been tested on the following platforms:

| # | Operating System | Processor | Platform |
|---|----------------------|--------------------------------|--------------------|
| 1 | Kinibi 400A (32-bit) | Samsung Electronics Exynos9810 | Samsung Galaxy S9+ |
| 2 | QSEE 5.0 (32-bit) | Qualcomm SDM845 | Samsung Galaxy S9+ |
| 3 | QSEE 5.0 (64-bit) | Qualcomm SDM845 | Samsung Galaxy S9+ |

Table 2. Tested Platforms

Notes:

- Please note that the Samsung Galaxy S9+ with Qualcomm SDM845 platform runs the QSEE 5.0 trusted execution environment, and QSEE 5.0, when running on the SDM845, supports both 64-bit and 32-bit trusted applications. Thus, the validation tested both a 64-bit version of the module and a 32-bit version of the module. Note as well that the Samsung Galaxy S9+ with Samsung Electronics Exynos9810 platform runs the Kinibi 400A trusted execution environment, and the Kinibi 400A trusted execution environment only supports 32-bits applications, thus the validation tested a single, 32-bits version of the module.
- 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

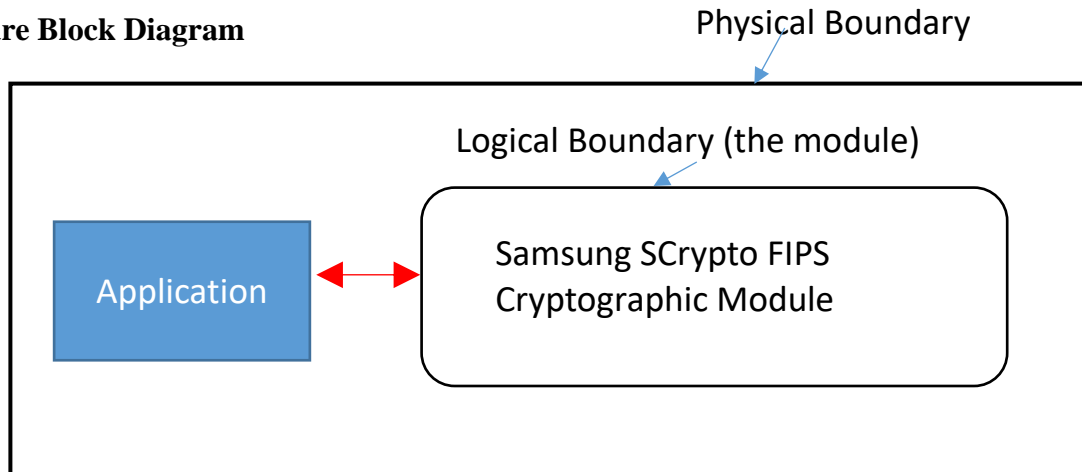


Figure 1: Software Block Diagram

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, DSA SGK, DSA 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 |

| Service | Role | Description/CSP | Access Permission |
|-------------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| Key agreement | User, CO | Used to perform key agreement primitives on behalf of the calling process (does not establish keys into the module). CSPs: DH Private, DH Public, DH Shared Secret, EC DH Private, EC DH Public and EC DH Shared Secret | read/write/execute |
| Digital signature | User, CO | Used to generate or verify RSA, DSA or ECDSA digital signatures. CSPs: RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK | read/write/execute |
| 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 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()

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 provides the following CAVP validated algorithms in either FIPS Approved mode or non-FIPS mode:

| 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 (AE, AD, AES-128, AES-256, FWD, 128, 256, 320, 320, 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_cfb128_encrypt | Cert. #5180 |
| AES-CMAC | CMAC (Generation/Verification, 128, 192, 256) | CMAC_Init CMAC_Update CMAC_Final CMAC_Reset | Cert. #5180 |
| AES-GCM | GCM (e/d), (128, 192 and 256) | CRYPTO_gcm128_init CRYPTO_gcm128_setiv CRYPTO_gcm128_aad CRYPTO_gcm128_encrypt CRYPTO_gcm128_decrypt CRYPTO_gcm128_new CRYPTO_gcm128_encrypt_ctr32 CRYPTO_gcm128_decrypt_ctr32 | Cert. #5180 |
| FIPS 186-4 DSA | KeyPairGen: (2048 and 3072 bits) SIG(gen): (2048 and 3072) with SHA-1/224/256/384/512. SIG(gen) with SHA-1 affirmed for use with protocols only SIG(ver): (1024, 2048, 3072) with SHA-1/224/256/384/512 | DSA_generate_key DSA_do_sign DSA_do_verify DSA_do_check_signature | Cert. #1344 |

| 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 SHA-1/224/256/384/512. Note: SIG(gen) with SHA- 1 affirmed for use with protocols only SigVer: CURVES (P-224, P-256, P-384, P-521) with SHA-1/224/256/384/512 | EC_KEY_generate_key ECDSA_do_sign ECDSA_do_verify ECDSA_sign ECDSA_verify | Cert. #1343 |
| KAS FFC/ECC | Curves tested: P-224 P-256 P-384 P-521 FFC: (FUNCTIONS INCLUDED IN IMPLEMENTATION: KPG) SCHEMES: Ephem: (KARole: Initiator) FB FC ECC: (FUNCTIONS INCLUDED IN IMPLEMENTATION: KPG) SCHEMES: EphemUnified: (KARole: Initiator) EC: P-256 ED: P-384 EE: P-521 | EC_KEY_generate_key DH_generate_parameters_ex ECDH_compute_key | CVL Cert. #1681 |
| SP800-90A DRBG | Block Cipher (CTR) based DRBG with AES-128, AES- 192, AES-256 | FIPS_drbg_new FIPS_drbg_init FIPS_drbg_instantiate FIPS_drbg_reseed FIPS_drbg_generate FIPS_drbg_uninstantiate FIPS_drbg_free FIPS_drbg_set_callbacks FIPS_drbg_set_rand_callbacks FIPS_drbg_get_app_data FIPS_drbg_set_app_data FIPS_drbg_get_blocklength FIPS_drbg_get_strength FIPS_drbg_set_check_interval FIPS_drbg_set_reseed_interval FIPS_drbg_health_check FIPS_drbg_default_drbg | Cert. #1955 |
| HMAC | HMAC with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 | HMAC_CTX_init HMAC_CTX_cleanup HMAC_Init_ex HMAC_Update HMAC_Final | Cert. #3436 |

| Algorithms | Modes and Description | API Function | FIPS Approved (Cert #) |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| 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 | Cert. #4184 |
| FIPS 186-4 RSA | FIPS186-4: 186-4KEY(gen): FIPS186-4_Fixed_e (10001); PGM (ProbRandom: (2048, 3072) PPTT:(C.2) [RSASSA-PKCS1_V1_5] SIG(gen) (2048/3072) with SHA-1/224/256/384/512. Note: SIG(gen) with SHA-1 affirmed for use with protocols only; SIG(Ver) (1024/2048/3072) with SHA- 1/224/256/384/512. [RSASSA-PSS]: SIG(gen) (2048/3072) with SHA- 1/224/256/384/512. Note: SIG(gen) with SHA-1 affirmed for use with protocols only; SIG(Ver) (1024/2048/3072) with SHA- 1/224/256/384/512 | RSA_generate_key_186_4 RSA_sign RSA_verify RSA_private_encrypt RSA_public_decrypt | Cert. #2781 |
| KBKDF (SP800-108) | CTR_Mode: (Llength (Min64 Max512) MACSupported([HMACSHA512]) LocationCounter([BeforeFixedData] [AfterFixedData]) rlength ([8] [16] [24] [32])) | NIST_KDF_HMAC_SHA512_Ctr _Mode | Cert. #175 |
| RSADP Primitive | RSADP: (Mod 2048) | RSA_decrypt | CVL Cert. #1682 |
| CKG (vendor affirmed) | | RSA_generate_key_186_4 DSA_generate_key EC_KEY_generate_key | |

Table 5. CAVP validated algorithms

Notes:

- Diffie-Hellman and EC Diffie-Hellman were tested with CVL Cert. #1681, but both were not self tested. Thus, CVL Cert. #1681 is not an approved algorithm.
- The AES-GCM IV generation method from each of AES #5180 is in compliance with IG A.5, scenario #2. The DRBG Cert. #1955 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 Cert. #5180; key establishment methodology provides 128 or 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 SP800-133. 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. |
| Diffie-Hellman | Key agreement scheme using Diffie-Hellman (2048-4096 bits). Key agreement is a service provided for calling process use, but not used to establish keys into the module. |
| EC Diffie-Hellman | Key agreement scheme using elliptic curve, supporting NIST defined P curves (P-224 P-256 P-384 P-521). Key agreement is a service provided for calling process use, but not used to establish keys into the module. |
| 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:

- Diffie-Hellman: key agreement; key establishment methodology provides between 112 and 150 bits of encryption strength.
- EC Diffie Hellman: key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength.
- RSA: CVL Cert. #1682, 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-Approved algorithm that is not permitted for use while operating in a FIPS 140-2 compliant fashion:

- Triple-DES (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 |
|-------------------------|-----------------------------------------------------------------------------------------------------------------|
| RSA SGK | RSA (2048/3072 bits) signature generation key |
| RSA KDK | RSA (2048/3072 bits) key decryption (private key transport) key |
| DSA SGK | FIPS 186-4 DSA (2048/3072 bits) signature generation key |
| ECDSA SGK | FIPS 186-4 ECDSA (P-224/P-256/P-384/P-521 Curves) signature generation key |
| DH Private | DH (224 - 379 bits) private key agreement key |
| DH Shared Secret | DH (2048 – 4096 bits) shared secret |
| EC DH Private | EC DH (P-224/P-256/P-384/P-521 Curves) private key agreement key |
| EC DH Shared Secret | EC DH (P-224/P-256/P-384/P-521 Curves) shared secret |
| AES EDK | AES (128/192/256 bits) encrypt / decrypt key |
| AES CMAC Key | AES (128/192/256 bits) CMAC generate / verify key |
| AES GCM Key | AES (128/192/256 bits) encrypt / decrypt / generate / verify key |
| HMAC Key | Keyed hash key (160/224/256/384/512 bits) |
| 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 /dev/random) |
| Key Wrap | AES Key Wrap using 128 or 256-bit keys |
| Key Derivation Key | SP800-108 KBKDF Key Derivation Key (512 bits) |

Table 7. Critical Security Parameters

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 |
| RSA KEK | RSA (2048/3072 bits) key encryption (public key Transport) key |
| DSA SVK | FIPS 186-4 DSA (1024/2048/3072 bits) signature verification key |
| ECDSA SVK | ECDSA (P-224/P-256/P-384/P-521 Curves) signature verification key |
| DH Public | DH (2048 – 4096 bits) public key agreement key |
| EC DH Public | EC DH (P-224/P-256/P-384/P-521 Curves) public key agreement key |

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 provides an SP 800-90A compliant DRBG service for creation of random data (which a calling application may use as symmetric key material) and for generation of DSA, Diffie-Hellman, EC DH and ECDSA private keys as shown in Table 7. Again, the module draws its DRBG seeds from dev/random.

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 API calls. 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 | FCC ID |
|--------------------|--------------------------------|------------|
| Samsung Galaxy S9+ | Samsung Electronics Exynos9810 | A3LSMG965F |
| Samsung Galaxy S9+ | Qualcomm SDM845 | A3LSMG965U |

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 for all of its FIPS-approved algorithms. The module executes these self-tests when the module is initialized. Self-tests can also be invoked by the calling application by unloading and reloading the module. 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
- DSA (sign and verify) Power On Self-Test
- ECDSA (sign and verify) Power On Self-Test

- RSA (sign and verify) KATs
- 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-1 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)
- DSA 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; instead 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>