Samsung Key Management Module v1.0

FIPS 140-2 Security Policy

version 2.2
Last Update: 2011-11-04

1. Introduction

This document is a non-proprietary FIPS 140-2 Security Policy for the Samsung Key Management Module v1.0 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 that a security policy is required:

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

1.2. Target Audience

This document is intended to be part of the package of documents that are submitted for FIPS 140-2 validation. It is intended for the following people:

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

2. Cryptographic Module Specification

This document is the non-proprietary security policy for the Samsung Key Management 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 Key Management Module is a software only security level 1 cryptographic module that provides key management services for user space applications. The Key Management module runs on an ARM processor.

The following table shows the overview of the security level for each of the eleven sections of the validation.

Security Component	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	N/A

Security Component	Security Level
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:

Manufacturer	Model	O/S & Ver.
Samsung	Galaxy S2	Android Gingerbread with Linux kernel based on version 2.6.35.7
Samsung	P4 LTE P4 WiFi	Android Honeycomb with Linux kernel based on version 2.6.36.3

Table 2: Tested Platforms

2.2. Description of Approved Mode

The crypto module has only FIPS 140-2 approved mode.

In Approved mode the module will support the following approved functions:

- AES (ECB)
- SHA-256
- RNG (ANSI X9.31)
- HMAC (with SHA-256)
- Password-Based Key Derivation Function (PBKDF) (NIST 800-132)

The Key Management module has only FIPS approved mode.

2.3. Cryptographic Module Boundary

2.3.1. Software Block Diagram

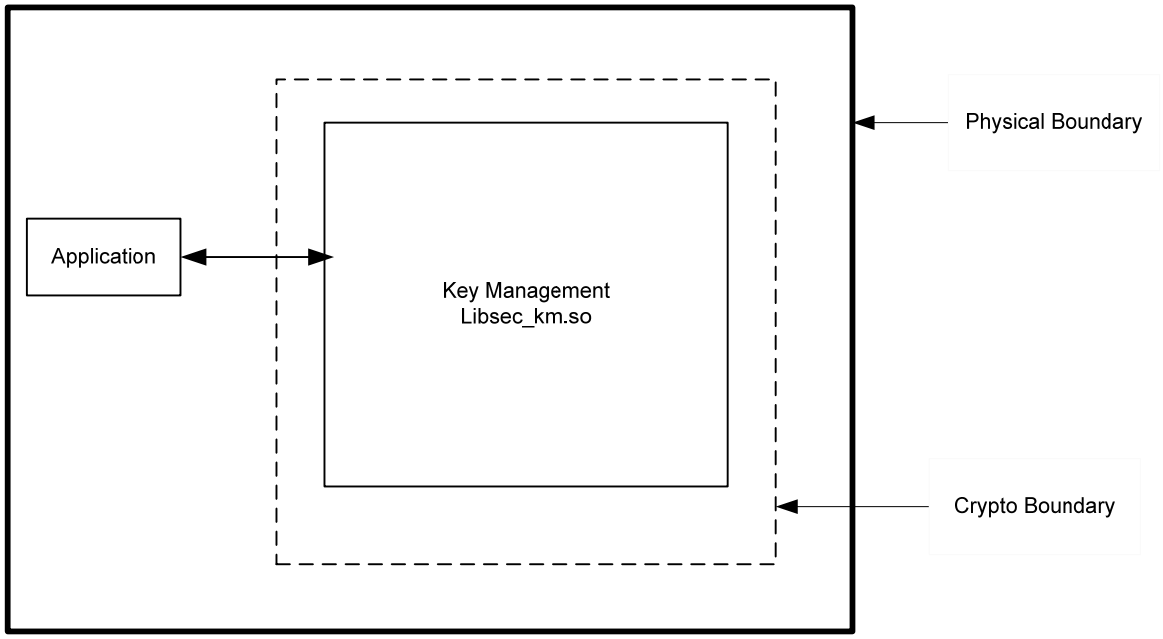

Figure 1: Software Block Diagram

The binary image that contains the Key Management module is as follows:

- libsec_km.so (version LK2.6.35.7_AGB_KM1.0) – Galaxy S2
- libsec_km.so (version LK2.6.36.3_AHC_KM1.0) – P4 LTE, P4 WiFi

Related documentation:

- S/W Detailed Level Design (FIPS_KM_Func_Design_v6.0.docx)
- Samsung Key Management Cryptographic Module v1.0 (SamsungKeyManagement_SPv.1.0.doc)

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

2.3.2. Hardware Block Diagram

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

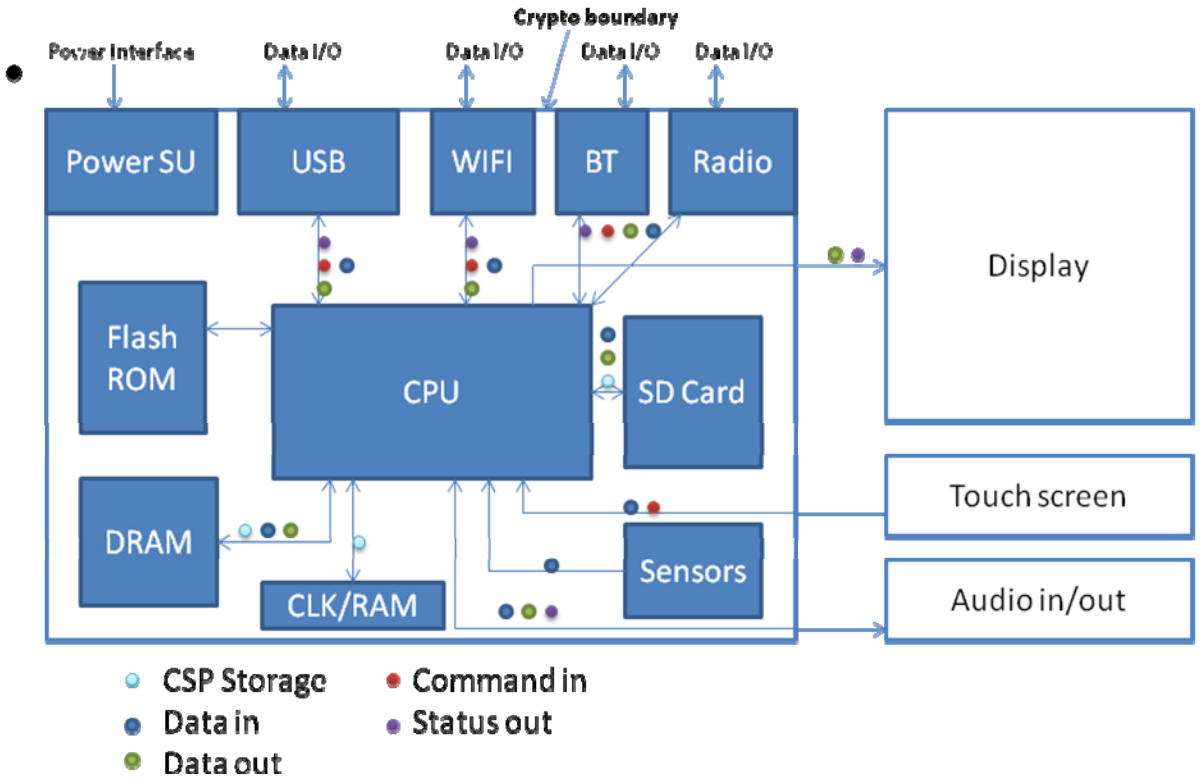

Figure 2: Hardware Block Diagram

Figure 3: Front View of Galaxy S2

Figure 4: Back View of Galaxy S2

Figure 5: Front View of P4 WiFi

Figure 6: Back View of P4 WiFi

Figure 7: Front View of P4 LTE

Figure 8: Back View of P4 LTE

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; device log file; the status of the module is provided at UI via system property rw.km_fips_status
Power Input	Physical power connector

Table 3: Ports and Interfaces

4. Roles, Services and Authentication

4.1. Roles

Role	Services (see list below)
User	Encryption, Decryption, Random Numbers, Digest Creation, Key Generation, Change Password, Verify Password, Check Status
Crypto Officer	Configuration, Encryption, Decryption, Random Numbers, Initialization of Module, Digest Creation, Key Generation, Change Password, Verify Password, Check Status

Table 4: Roles

The Module meets all FIPS 140-2 level 1 requirements for Roles and Services, implementing both User and Crypto Officer roles. The Module does not allow concurrent operators.

4.2. Services

Role	Service	CSP	Modes	FIPS Approved (Cert #)	API Calls	Access (Read, Write, Execute)
User, Crypto Officer	AES encryption and decryption	256 bit keys	ECB	(Cert #1742) - P4 LTE and P4 Wifi (Cert #1741) – Galaxy S2	SECKM_AES_set_encrypt_key() SECKM_AES_set_decrypt_key() SECKM_AES_encrypt() SECKM_AES_decrypt()	R, W, EX
User, Crypto Officer	HMAC (with SHA-256)	HMAC Key		(Cert #1019) – P4 LTE and P4 Wifi (Cert #1018) – Galaxy S2	SECKM_HMAC_SHA256()	R, W, EX
User, Crypto Officer	SHA-256	N/A	N/A	(Cert #1529) – P4 LTE and P4 Wifi (Cert #1528) – Galaxy S2	SECKM_SHA256_Init() SECKM_SHA256_Update() SECKM_SHA256_Final()	R, W, EX
User, Crypto Officer	RNG ANSI X9.31	Seed Key	AES-128	(Cert #929) – P4 LTE and P4 Wifi (Cert #928) –	SECKM_PRNG_init(), SECKM_PRNG_set_seed() SECKM_PRNG_get16()	R, W, EX

Role	Service	CSP	Modes	FIPS Approved (Cert #)	API Calls	Access (Read, Write, Execute)
				Galaxy S2		
User, Crypto Officer	Generate Key	Password, DEK, EDK	N/A	N/A	Create_EDK Input: password Output: key(DEK), encrypted key (EDK) Return: status	R, W, EX
User, Crypto Officer	Verify Password	Password, EDK	N/A	N/A	Verify_EDK Input: password, encrypted key (EDK) Return: status	R, W, EX
User, Crypto Officer	Password Change	Password, EDK	N/A	N/A	Change_EDK Input: old, new passwords, encrypted key (EDK) Output: encrypted key (EDK) Return: status	R, W, EX
User, Crypto Officer	Get Key	Password, EDK	N/A	N/A	Decrypt_EDK Input: password, encrypted key (EDK) Output: key Return: status	R, W, EX
User, Crypto Officer	Initialization	N/A	N/A	N/A	When any application invokes the module	N/A
User, Crypto Officer	Self Test	N/A	N/A	N/A	_all_checks() get_fips_status()	N/A
Crypto Officer	Check Status/Get State	N/A	N/A	N/A	get_fips_status()	R

Table 5: Services

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

The following diagram represents the states and transitions of the crypto module. States are represented by blue boxes and transitions by arrows.

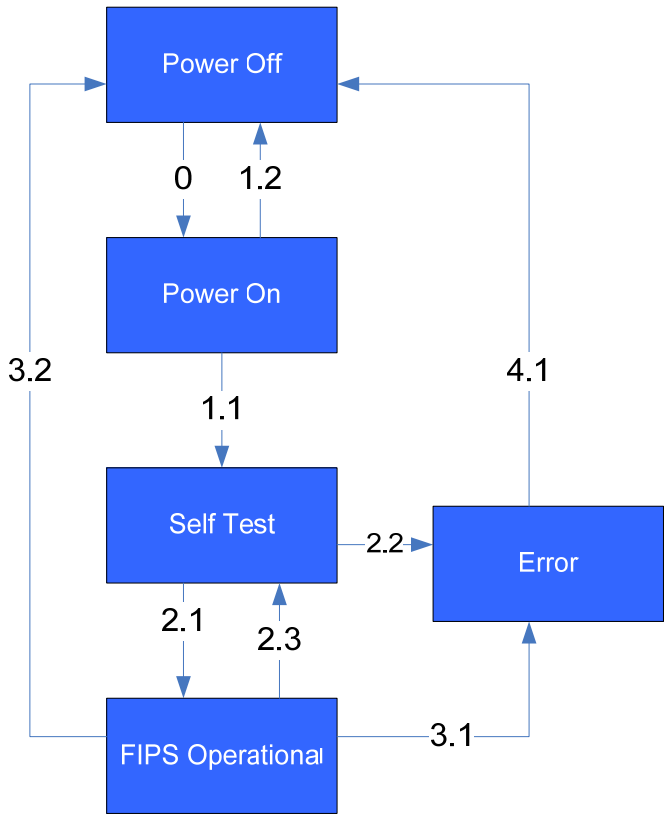

Figure 9: Crypto Module Finite State Machine

Power Off: No power. The Power Off state is entered from any state when power is removed or a controlled shutdown is performed. The only transition from the Power Off state is to the Power On state.

Power On: Power is on, module is initialized.

Self Test: The crypto module performs power-up self tests (consisting of software integrity test and known answer tests). If any self test fails, the system immediately transitions to the Error state. If the self test is successful, the system transitions to FIPS Operational state.

FIPS 140-2 Operational State: This is the normal operational mode of the module in which FIPS 140-2 approved services and non-approved services are available.

Error State: The crypto module is in an error state due to errors during self test and conditional tests. It does not allow any cryptographic operation in this state. The error state is non-recoverable. To clear the error state, the module has to be rebooted.

Transition	Starting State	Ending State	Reason for Transition	Control/Data Input	Data/Status Output
0	Power Off	Power On	Power on	Environment Init	No Data Output
1.1	Power On	Self Test	Initialization in FIPS 140-2 approved mode	Upon first invocation of any API	No Data Output
1.2	Power On	Power Off	Crypto module unloaded	Crypto module unload	No Data Output
2.1	Self Test	FIPS Operational	Successful completion of self tests	get_fips_status() runs the self test	System property km_fips_status set as “ready”
2.2	Self Test	Error	Self Test failure	get_fips_status() runs the self test	System property km_fips_status set as “error_selftest” or “error_integrity”
2.3	FIPS Operational	Self Test	Access request from external application	get_fips_status()	If passes: System property km_fips_status If fails: “error_selftest” or “error_integrity”
3.1	FIPS Operational	Error	Conditional tests failed	PRNG_get16()	System property km_fips_status set as “error_continuity”
3.2	FIPS Operational	Power Off	Finish crypto module	Crypto module unload	No Data Output
4.1	Error	Power Off	Finish crypto module	Crypto module unload	No Data Output

Table 6: Transitions

6. Physical Security

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

7. Operational Environment

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

7.1. Policy

Both the phone and the tablet are single user devices. The operating system shall be restricted to a single operator mode of operation (i.e., concurrent operators are explicitly excluded).

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

8. Cryptographic Key Management

8.1. Random Number Generation

The Module employs an ANSI X9.31 compliant random number generator for creation of keys which is externally seeded by the /dev/random utility. The /dev/random utility is outside of the cryptographic boundary of this module.

The /dev/random utility uses environmental events to accumulate its entropy pool. Sources that provide input to entropy are the light sensor, proximity sensor, gyro sensor, vibrator, touch sensor, and the bluetooth on the device. When random numbers are requested, /dev/random provides data only if the entropy pool has enough data based on the estimate of the randomness generated from the environmental noises. If there is not enough data to be provided, the random read is blocked after entropy is exhausted. SHA digest is applied on the entropy collected before the data is given as output. This assures the unpredictability of the entropy collection itself.

8.2. Key Generation

The following keys/CSPs are used in the key management module:

- Seed Key
- Password: User entered 4 to 32 characters long
- Master key: Generated using PBKDF
- Device encryption key: 256 bits long, generated from RNG
- Encrypted device encryption key: Encrypted DEK with master key using AES
- EDK Payload or Payload: consists of EDK, salt, HMAC of EDK

8.2.1. Master key

Master Key (MK) is generated using PBKDF (NIST 800-132).

The master key generation processes is shown in the following diagram:

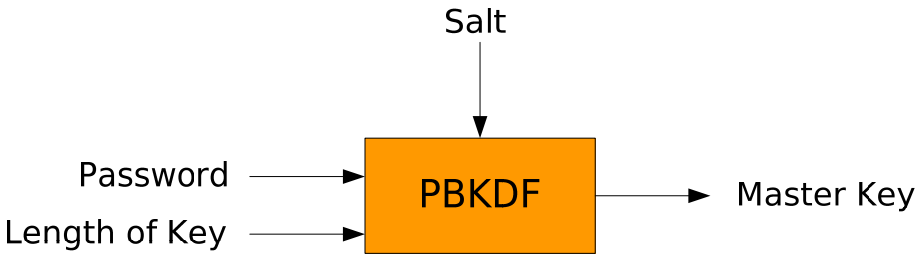

Figure 10: Master Key Generation Processes

Salt for the PBKDF is 16 bytes read from RNG. Iteration count used is 1024 considering the limitations of the processing speed available in the environment.

Password length must be greater than 4, but cannot be greater than 32 characters.

Note: As per NIST 800-132, passwords shorter than 10 characters are usually considered to be weak.

This master key is used as an AES encryption key (256 bits) to encrypt the device encryption key (DEK). It is also used to compute HMAC with SHA-256.

8.2.2. Device encryption key

Device encryption key (DEK) is derived as 32 bytes read from RNG. A pre-defined hex-string is encrypted with master key, which is XORed with the first 128 bits of DEK. The first byte of the pre-defined hex-string is incremented by one. The new hex-string is encrypted with master key, which is XORed with the second 128 bits of DEK.

8.2.3 Encrypted device encryption key (EDK) and EDK payload

DEK is protected by encrypting it with the master key. The resulting encrypted key is called EDK.

The following diagram provides overall encryption procedure:

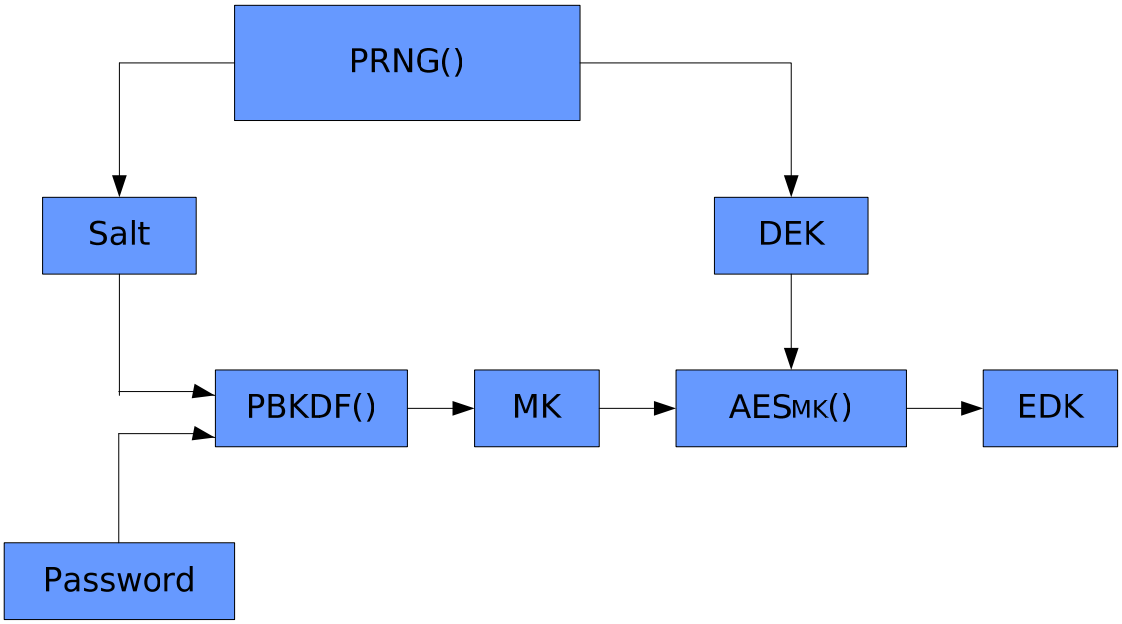

Figure 11: Overall Encryption Procedure

From the above procedure, Salt, EDK, and Hash are given as output or payload.

EDK Payload is defined as:

- Encrypted DEK – EDK itself
- HMAC of EDK
- Salt

The following diagram provides overall verification procedure:

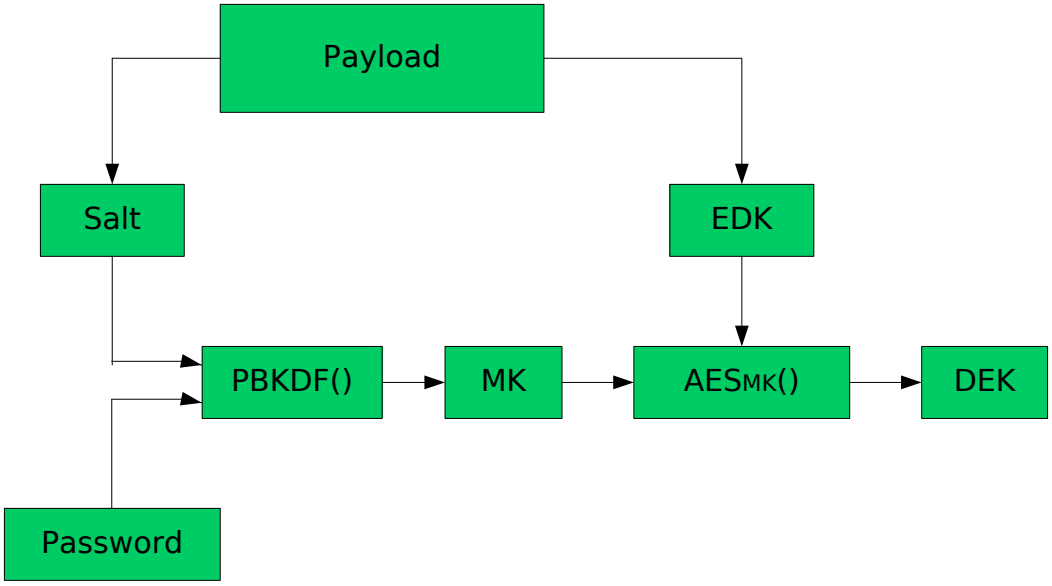

Figure 12: Overall Verification Procedure

The following diagram shows the message digest creation using the master key:

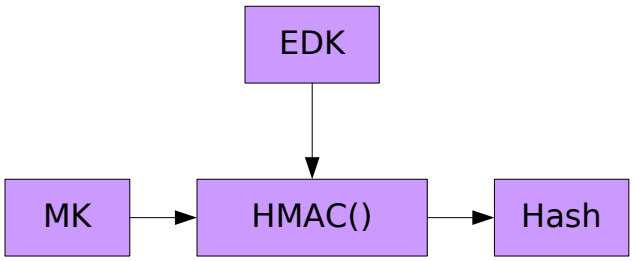

Figure 13: Message Digest Creation Using Master Key

EDK payload is input. Based on given password and the Salt from EDK Payload, using PBKDF function, master key is generated. Using the generated master key, HMAC of EDK is calculated.

If the input HMAC of EDK is equal to the calculated HMAC of EDK, password and/or EDK is successfully verified.

Upon successful verification of EDK, EDK can be decrypted using the calculated master key to get the DEK as explained above.

Change of EDK is allowed and it follows the process:

- Verify the given password using Verify EDK Step
- Upon successful verification, decrypt the EDK to get DEK
- Generate new master key by giving new password to PBKDF
- Generate EDK Payload by encrypting DEK and calculating HMAC of EDK

8.3. Key Entry and Output

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 output of PBKDF is DEK, which is supplied to the calling application via API call.

8.4. Key Storage

The encryption key (EDK) for the purposes of storage, is randomly generated which is protected using master key.

8.5. Zeroization Procedure

This is a dynamic library module which provides API based services. All the temporary variables for CSP are zeroized using memset. DEK is zeroized using free_DEK(). It is the responsibility of the calling application to zeroize DEK by invoking this function.

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

Lab Name: PC Engineering Laboratory, Inc
FCC Registration: #90864

Device	Model Name	FCC ID
Galaxy S2	GT-I9100	A3LGTI9100
P4-LTE	SCH-I905	A3LSCHI905
P4-WiFi	GT-P7510	A3LGTP7510

Table 7: FCC IDs

10. Self Tests

Self test is initiated upon invoking any of the exposed ESDK APIs. Internal algorithm APIs are internal to the module.

All the exposed APIs check get_fips_status(). If the FIPS status is UNDEFINED, it will start the self test procedure. If the self test is successful, the API will continue to get executed. If the self test fails, the module will be put in error state and thus disabled. There will be an indicator in User Interface as well as in system property “rw.km_fips_status.”

During self test, the value calculated is verified against the known answer. If there is no match, then immediately the module enters into FIPS_ERR state. Once the module is in FIPS_ERR state, the module becomes unusable via any interface.

Cryptographic algorithm tests (Known Answer Tests):

- AES encryption/decryption
- HMAC-SHA-256
- SHA-256
- Random Number Generator

Users can check the module status in two ways:

- From the main screen, start the Settings application. Under the Settings menu, go to “About Phone.”(“About tablet” if using P4 LTE or WiFi) Status is displayed in the listings.
- When device encryption is enabled, the Settings application will not be available because a password is required to show the main screen. In such cases, the error status is shown on the password screen itself.

10.1. Integrity Check

- Build Time

    - SHA-256- HMAC calculated on libsec_km.so (dynamic library) file
        - HMAC appended to libsec_km.so file

- Run Time
    - libsec_km.so is read as a file
    - When Integrity test routine is called
        - Perform HMAC-SHA-256 on the read libsec_km.so value in ram
        - Read stored hmac located after libsec_km.so (last 32 bytes)
        - If calculated and stored values do not match, set error state, STATUS_ERROR_INTEGRITY and the system property as “error_integrity”

Note: Similarly, integrity check for the P4 Tablet is performed on boot.img in the same manner as above.

10.2. Conditional Tests

A continuous random number generator test is performed during each use of the approved RNG. If values of two consecutive random numbers match, then crypto module goes into error state. This RNG is externally seeded by /dev/random, which is also subjected to a CRNG test. Note that /dev/random is outside the module boundary.

11. Design Assurance

11.1. Configuration Management

All source code is maintained in internal source code servers and the tool, Perforce, is used as code control. Release is based on the Change List number maintained by Perforce, which is auto-generated. Every check-in process creates a new change list number.

Versions of controlled items include information about each version. For documentation, revision history inside the document provides the current version of the document. Version control maintains the all the previous version and the version control system automatically numbers revisions.

For product versioning, a major/minor scheme is used. The Linux kernel version is included in the version. For example, LK2.6.35.7_AGB_KM1.0 is a particular version. Here LK stands for Linux Kernel version and the number following that is the actual Linux Kernel version. AGB stands for Android Gingerbread. KM1.0 stands for version major and Minor. KM1.1 would represent an updated minor version.

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.

All documents are maintained in an internal document server per project. The versioning tool used is Sub version (svn). The version number is auto generated by the tool and version is controlled by a check-in and check-out mechanism.

In the development team, only authorized developers verified by login/password is allowed to access permitted documents in version control system.

11.2. Delivery and Operation

The key management module is never released as Source code. It may be released as Source for internal purposes based on Change List number generated by Perforce.

Official release binaries can only be made through build system, run by the Software Project Lead.

Once the device enters manufacturing phase, source code branch is locked. Once it is locked, source control system provides only read access. Thus, no one can then modify the source code in the Perforce depot.

Final binary thus built is registered with its hash value to internal system which is not connected to any other network.

Only authorized personnel through VPN can register the binary to automated manufacturing system, so that it can be downloaded to hardware without any manual intervention. Employees are not allowed to bring in any personal belongings to the manufacturing facility and entrance is controlled with employee ID based badge access and monitored using CCTV.

The binary is released only by Samsung released tool and OTA (Over the Air). Over the air mechanism is controlled by service providers. If the binary is modified by unauthorized entity, the device has a feature to detect the change and thus not accept the binary modified by an unauthorized entity.

12. Mitigation of Other Attacks

No other attacks are mitigated.

13. Glossary and Abbreviations

AES	Advanced Encryption Specification
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CMT	Cryptographic Module Testing
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CVT	Component Verification Testing
DEK	Device Encryption Key
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
EAL	Evaluation Assurance Level
EDK	Encrypted Device Encryption Key
FSM	Finite State Model
HMAC	Hash Message Authentication Code
MAC	Message Authentication Code
MK	Master Key
NIST	National Institute of Science and Technology
NVLAP	National Voluntary Laboratory Accreditation Program
OFB	Output Feedback
O/S	Operating System
PBKDF	Password Based Key Derivation Function
RNG	Random Number Generator
RNGVS	Random Number Generator Validation System
RSA	Rivest, Shamir, Addleman
SDK	Software Development Kit
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SLA	Service Level Agreement
SOF	Strength of Function
SSH	Secure Shell
SVT	Scenario Verification Testing

TDES	Triple DES
TOE	Target of Evaluation
UI	User Interface

Table 8: Abbreviations

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 180-3 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] FIPS 186-3 Digital Signature Standard (DSS), http://csrc.nist.gov/publications/PubsFIPS.html

[8] The Random Number Generator Validation System (RNGVS), http://csrc.nist.gov/groups/STM/cavp/documents/rng/RNGVS.pdf