LG Electronics Inc. G3 Smartphone (MDFPP11) Security Target

Version 1.0
2014/11/13

Prepared for:

LG Electronics Inc.

20 Yoido-dong, Youngdungpogu, Seoul 152-721, Korea

Prepared By:

www.gossamersec.com

LIST OF TABLES

1. Security Target Introduction

This section identifies the Security Target (ST) and Target of Evaluation (TOE) identification, ST conventions, ST conformance claims, and the ST organization. The TOE is G3 Smartphone provided by LG Electronics Inc.. The TOE is being evaluated as a mobile device.

The Security Target contains the following additional sections:

- Conformance Claims (Section 2)
- Security Objectives (Section 3)
- Extended Components Definition (Section 4)
- Security Requirements (Section 5)
- TOE Summary Specification (Section 6)

Conventions

The following conventions have been applied in this document:

- Security Functional Requirements – Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.
    - Iteration: allows a component to be used more than once with varying operations. In the ST, iteration is indicated by a letter placed at the end of the component. For example FDP_ACC.1a and FDP_ACC.1b indicate that the ST includes two iterations of the FDP_ACC.1 requirement, a and b.
    - Assignment: allows the specification of an identified parameter. Assignments are indicated using bold and are surrounded by brackets (e.g., [**assignment**]). Note that an assignment within a selection would be identified in italics and with embedded bold brackets (e.g., [***[selected-assignment]***]).
    - Selection: allows the specification of one or more elements from a list. Selections are indicated using bold italics and are surrounded by brackets (e.g., [***selection***]).
    - Refinement: allows the addition of details. Refinements are indicated using bold, for additions, and strike-through, for deletions (e.g., "… **all** objects …" or "… ~~some~~ **big** things …").
- The NDPP uses an additional convention – the 'case' – which defines parts of an SFR that apply only when corresponding selections are made or some other identified conditions exist. Only the applicable cases are identified in this ST and they are identified using **bold** text.
- Other sections of the ST – Other sections of the ST use bolding to highlight text of special interest, such as captions.

1.1 Security Target Reference

ST Title – LG Electronics Inc. G3 Smartphone (MDFPP11) Security Target

ST Version – Version 1.0

ST Date – 2014/11/13

1.2 TOE Reference

TOE Identification – LG Electronics Inc. G3 Smartphone

TOE Developer – LG Electronics Inc.

Evaluation Sponsor – LG Electronics Inc.

1.3 TOE Overview

The Target of Evaluation (TOE) is G3 Smartphone featuring a 5.5 Inch, Quad HD 2,560 x 1,440 resolution IPS display; 2.1 MP front camera; 13 MP rear camera with Laser Auto Focus, Optical Image Stablization and Dual Flash; 1W speaker with boost amp; Bluetooth 4.0 LE (APT-x); USB 2.0HS; Wi-Fi 802.11 a/b/g/n/ac; A-GPS; Glonass; HDMI supported through SlimPort; NFC; 32GB RAM; 32GB Flash; and microSD (up to 2TB) Slot.

The G3 allows basic telephony features (make and receive phone calls, send and receive SMS/MMS messages) as well as advance network connectivity (allowing connections to both 802.11 Wi-Fi and 2G/3G/4G LTE mobile data networks). The G3 supports using client certificates to connect to access points offering WPA2 networks with 802.1x/EAP-TLS, or alternatively connecting to cellular base stations when utilizing mobile data.

The G3 offers mobile applications an Application Programming Interface (API) including that provided by the Android framework and extensions to the MDM API by LG.

1.4 TOE Description

The TOE is a mobile device designed to support enterprises and individual users alike. Based upon Android 4.4 and improved by LG, the TOE provides wireless connectivity and provides an execution environment for mobile applications.

The evaluated TOE contains 32GB of internal Flash storage and 3GB of memory.

LG also has different carrier versions including:

LG G3 D850 (AT&T)
LG G3 VS985 (Verizon)
LG G3 LS990 (Sprint)
LG G3 D851 (T-Mobile)

1.4.1 TOE Architecture

The TOE provides a rich Application Programming Interface to mobile applications and provides users installing an application to either approve or reject an application based upon the API access that the application requires.

The TOE also provides users with the ability to protect Data-At-Rest with AES encryption, including all user and mobile application data stored in the user's data partition. The TOE affords special protection to all user and application cryptographic keys stored in the TOE. Moreover, the TOE provides users the ability to AES encrypt data and files stored on an SD Card inserted into the device.

Finally, the TOE interacts with a Mobile Device Management to allow enterprise control of the configuration and operation of the device so as to ensure adherence to enterprise-wide policies.

1.4.1.1 Physical Boundaries

The TOE's physical boundary is the physical perimeter of its enclosure (without the rear access cover present, so that one can access and replace the device's battery, SIM, and SD Card).

1.4.1.2 Logical Boundaries

This section summarizes the security functions provided by G3:

- Cryptographic support
- User data protection
- Identification and authentication

- Security management
- Protection of the TSF
- TOE access
- Trusted path/channels

1.4.1.2.1 Cryptographic support

The TOE includes cryptographic modules with FIPS certified algorithms for a wide range of cryptographic functions including: asymmetric key generation and establishment, symmetric key generation, encryption/decryption, cryptographic hashing and keyed-hash message authentication. These functions are supported with suitable random bit generation, key derivation, salt generation, initialization vector generation, secure key storage, and key and protected data destruction. These primitive cryptographic functions are used to implement security protocols such as TLS and HTTPS and also to encrypt Data-At-Rest (including the generation and protection of keys and key encryption keys) used by the TOE. Many of these cryptographic functions are also accessible as services to applications running on the TOE.

1.4.1.2.2 User data protection

The TOE is designed to control access to system services by hosted applications, including protection of the Trust Anchor Database. Additionally, the TOE is designed to protect user and other sensitive data using encryption so that even if a device is physically lost, the data remains protected.

1.4.1.2.3 Identification and authentication

The TOE supports a number of features related to identification and authentication. From a user perspective, except for making phone calls to an emergency number, a password (i.e., Password Authentication Factor) must be correctly entered to unlock the TOE. Also, even when the TOE is unlocked the password must be re-entered to change the password. Passwords are obscured when entered so they cannot be read from the TOE's display and the frequency of entering passwords is limited and when a configured number of failures occurs, the TOE will be wiped to protect its contents. Passwords can be constructed using upper and lower cases characters, numbers, and special characters and passwords up to 14 characters are supported.

The TOE can also serve as an 802.1X supplicant and can use X509v3 and validate certificates for EAP-TLS, TLS, and HTTPS exchanges.

1.4.1.2.4 Security management

The TOE provides all the interfaces necessary to manage the security functions identified throughout this Security Target as well as other functions commonly found in mobile devices. Many of the available functions are available to users of the TOE while many are restricted to administrators operating through a Mobile Device Management solution once the TOE has been enrolled. Once the TOE has been enrolled and then un-enrolled, it will remove Enterprise applications, remove MDM policies, and disable CC mode.

1.4.1.2.5 Protection of the TSF

The TOE implements a number of features designed to protect itself to ensure the reliability and integrity of its security features. It protects particularly sensitive data such as cryptographic keys so that they are not accessible or exportable. It also provides its own timing mechanism to ensure that reliable time information is available (e.g., for log accountability). It enforces read, write, and execute memory page protections, uses address space layout randomization, and stack-based buffer overflow protections to minimize the potential to exploit application flaws. It is also designed to protect itself from modification by applications as well as to isolate the address spaces of applications from one another to protect those applications.

The TOE includes functions to perform self-tests and software/firmware integrity checking so that it might detect when it is failing or may be corrupt. If any of the self-tests fail, the TOE will not go into an operational mode. It also includes mechanisms (i.e., verification of the digital signature of each new image) so that the TOE itself can be updated while ensuring that the updates will not introduce malicious or other unexpected changes in the TOE.

Digital signature checking also extends to verifying applications prior to their installation as all applications must have signatures (even if self-signed)..

1.4.1.2.6 TOE access

The TOE can be locked, obscuring its display, by the user or after a configured interval of inactivity. The TOE also has the capability to display an administrator specified (using an MDM) advisory message (banner) when the user unlocks the TOE for the first use after reboot.

The TOE is also able to attempt to connect to wireless networks as configured.

1.4.1.2.7 Trusted path/channels

The TOE supports the use of 802.11-2012, 802.1X, and EAP-TLS to secure communications channels between itself and other trusted network devices.

1.4.2 TOE Documentation

LG Electronics Inc. G3 Administrator Guidance, Version 1.0, 2014/10/30

2. Conformance Claims

This TOE is conformant to the following CC specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 3, July 2009.
    - Part 2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1 Revision 3, July 2009.
    - Part 3 Extended
- Protection Profile For Mobile Device Fundamentals, Version 1.1, 12 February 2014 (MDFPP11)
- Package Claims:
    - Assurance Level: EAL 1 augmented with ALC_TSU_EXT.1

2.1 Conformance Rationale

The ST conforms to the MDFPP11. As explained previously, the security problem definition, security objectives, and security requirements have been drawn from the PP.

3. Security Objectives

The Security Problem Definition may be found in the MDFPP11 and this section reproduces only the corresponding Security Objectives for operational environment for reader convenience. The MDFPP11 offers additional information about the identified security objectives, but that has not been reproduced here and the MDFPP11 should be consulted if there is interest in that material.

In general, the MDFPP11 has defined Security Objectives appropriate for mobile device and as such are applicable to the G3 Smartphone TOE.

3.1 Security Objectives for the Operational Environment

OE.CONFIG TOE administrators will configure the Mobile Device security functions correctly to create the intended security policy.

OE.NOTIFY The Mobile User will immediately notify the administrator if the Mobile Device is lost or stolen.

OE.PRECAUTION The Mobile User exercises precautions to reduce the risk of loss or theft of the Mobile Device.

4. Extended Components Definition

All of the extended requirements in this ST have been drawn from the MDFPP11. The MDFPP11 defines the following extended requirements and since they are not redefined in this ST the MDFPP11 should be consulted for more information in regard to those CC extensions.

Extended SFRs:

- FCS_CKM_EXT.1: Extended: Cryptographic Key Support
- FCS_CKM_EXT.2: Extended: Cryptographic Key Random Generation
- FCS_CKM_EXT.3: Extended: Cryptographic Key Generation
- FCS_CKM_EXT.4: Extended: Key Destruction
- FCS_CKM_EXT.5: Extended: TSF Wipe
- FCS_CKM_EXT.6: Extended: Salt Generation
- FCS_HTTPS_EXT.1: HTTPS Protocol
- FCS_IV_EXT.1: Extended: Initialization Vector Generation
- FCS_RBG_EXT.1(*): Extended: Cryptographic Operation (Random Bit Generation)
- FCS_SRV_EXT.1: Extended: Cryptographic Algorithm Services
- FCS_STG_EXT.1: Extended: Cryptographic Key Storage
- FCS_STG_EXT.2: Extended: Encrypted Cryptographic Key Storage
- FCS_STG_EXT.3: Extended: Integrity of encrypted key storage
- FCS_TLS_EXT.1: Extended: EAP TLS Protocol
- FCS_TLS_EXT.2: TLS Protocol
- FDP_ACF_EXT.1: Extended: Security access control
- FDP_DAR_EXT.1: Extended: Data-At-Rest Protection
- FDP_IFC_EXT.1: Extended: Subset information flow control
- FDP_STG_EXT.1(1): Extended: User Data Storage
- FIA_AFL_EXT.1: Authentication failure handling
- FIA_PAE_EXT.1: Extended: PAE Authentication
- FIA_PMG_EXT.1: Extended: Password Management
- FIA_TRT_EXT.1: Extended: Authentication Throttling
- FIA_UAU_EXT.1: Extended: Authentication for Cryptographic Operation
- FIA_UAU_EXT.2: Timing of Authentication
- FIA_UAU_EXT.3: Extended: Re-Authentication
- FIA_X509_EXT.1: Extended: Validation of certificates
- FIA_X509_EXT.2: Extended: X509 certificate authentication
- FIA_X509_EXT.3: Extended: Request Validation of certificates
- FMT_SMF_EXT.1: Extended: Specification of Remediation Actions
- FPT_AEX_EXT.1: Extended: Anti-Exploitation Services (ASLR)

- FPT_AEX_EXT.2: Extended: Anti-Exploitation Services (Memory Page Permissions)
- FPT_AEX_EXT.3: Extended: Anti-Exploitation Services (Stack Overflow Protection)
- FPT_AEX_EXT.4: Extended: Domain Isolation
- FPT_KST_EXT.1: Extended: Key Storage
- FPT_KST_EXT.2: Extended: No Key Transmission
- FPT_KST_EXT.3: Extended: No Plaintext Key Export
- FPT_NOT_EXT.1: Extended: Event Notification
- FPT_TST_EXT.1: Extended: TSF Cryptographic Functionality Testing
- FPT_TST_EXT.2: Extended: TSF Integrity Testing
- FPT_TUD_EXT.1: Extended: Trusted Update: TSF version query
- FPT_TUD_EXT.2: Extended: Trusted Update Verification
- FTA_SSL_EXT.1: Extended: TSF- and User-initiated locked state
- FTA_TAB.1: Default TOE Access Banners - FTA_WSE_EXT.1: Extended: Wireless Network Access
- FTP_ITC_EXT.1: Extended: Trusted channel Communication

Extended SARs:

- ALC_TSU_EXT.1: Timely Security Updates

5. Security Requirements

This section defines the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) that serve to represent the security functional claims for the Target of Evaluation (TOE) and to scope the evaluation effort.

The SFRs have all been drawn from the MDFPP11. The refinements and operations already performed in the MDFPP11 are not identified (e.g., highlighted) here, rather the requirements have been copied from the MDFPP11 and any residual operations have been completed herein. Of particular note, the MDFPP11 made a number of refinements and completed some of the SFR operations defined in the Common Criteria (CC) and that PP should be consulted to identify those changes if necessary.

The SARs are also drawn from the MDFPP11 which includes all the SARs for EAL 1 augmented with ALC_TSU_EXT.1. However, the SARs are effectively refined since requirement-specific 'Assurance Activities' are defined in the MDFPP11 that serve to ensure corresponding evaluations will yield more practical and consistent assurance than the EAL 1 augmented with ALC_TSU_EXT.1 assurance requirements alone. The MDFPP11 should be consulted for the assurance activity definitions.

5.1 TOE Security Functional Requirements

The following table identifies the SFRs that are satisfied by G3 Smartphone TOE.

Requirement Class	Requirement Component
FCS: Cryptographic support	FCS_CKM.1(1): Refinement: Cryptographic key generation
	FCS_CKM.1(2): Cryptographic key generation
	FCS_CKM.1(3): Cryptographic key generation
	FCS_CKM.2: Cryptographic key distribution
	FCS_CKM_EXT.1: Extended: Cryptographic Key Support
	FCS_CKM_EXT.2: Extended: Cryptographic Key Random Generation
	FCS_CKM_EXT.3: Extended: Cryptographic Key Generation
	FCS_CKM_EXT.4: Extended: Key Destruction
	FCS_CKM_EXT.5: Extended: TSF Wipe
	FCS_CKM_EXT.6: Extended: Salt Generation
	FCS_COP.1(1): Cryptographic operation
	FCS_COP.1(2): Cryptographic operation
	FCS_COP.1(3): Refinement: Cryptographic operation
	FCS_COP.1(4): Refinement: Cryptographic operation
	FCS_COP.1(5): Refinement: Cryptographic operation
	FCS_HTTPS_EXT.1: HTTPS Protocol
	FCS_IV_EXT.1: Extended: Initialization Vector Generation
	FCS_RBG_EXT.1(*): Extended: Cryptographic Operation (Random Bit Generation)
	FCS_SRV_EXT.1: Extended: Cryptographic Algorithm Services
	FCS_STG_EXT.1: Extended: Cryptographic Key Storage
	FCS_STG_EXT.2: Extended: Encrypted Cryptographic Key Storage
	FCS_STG_EXT.3: Extended: Integrity of encrypted key storage
	FCS_TLS_EXT.1: Extended: EAP TLS Protocol
	FCS_TLS_EXT.2: TLS Protocol
FDP: User data protection	FDP_ACF_EXT.1: Extended: Security access control
	FDP_DAR_EXT.1: Extended: Data-At-Rest Protection

	FDP_IFC_EXT.1: Extended: Subset information flow control
	FDP_STG_EXT.1(1): Extended: User Data Storage
FIA: Identification and authentication	FIA_AFL_EXT.1: Authentication failure handling
	FIA_PAE_EXT.1: Extended: PAE Authentication
	FIA_PMG_EXT.1: Extended: Password Management
	FIA_TRT_EXT.1: Extended: Authentication Throttling
	FIA_UAU.7: Protected authentication feedback
	FIA_UAU_EXT.1: Extended: Authentication for Cryptographic Operation
	FIA_UAU_EXT.2: Timing of Authentication
	FIA_UAU_EXT.3: Extended: Re-Authentication
	FIA_X509_EXT.1: Extended: Validation of certificates
	FIA_X509_EXT.2: Extended: X509 certificate authentication
	FIA_X509_EXT.3: Extended: Request Validation of certificates
FMT: Security management	FMT_MOF.1(1): Management of security functions behavior
	FMT_MOF.1(2): Management of security functions behavior
	FMT_SMF.1: Specification of Management Functions
	FMT_SMF_EXT.1: Extended: Specification of Remediation Actions
FPT: Protection of the TSF	FPT_AEX_EXT.1: Extended: Anti-Exploitation Services (ASLR)
	FPT_AEX_EXT.2: Extended: Anti-Exploitation Services (Memory Page Permissions)
	FPT_AEX_EXT.3: Extended: Anti-Exploitation Services (Stack Overflow Protection)
	FPT_AEX_EXT.4: Extended: Domain Isolation
	FPT_KST_EXT.1: Extended: Key Storage
	FPT_KST_EXT.2: Extended: No Key Transmission
	FPT_KST_EXT.3: Extended: No Plaintext Key Export
	FPT_NOT_EXT.1: Extended: Event Notification
	FPT_STM.1: Reliable time stamps
	FPT_TST_EXT.1: Extended: TSF Cryptographic Functionality Testing
	FPT_TST_EXT.2: Extended: TSF Integrity Testing
	FPT_TUD_EXT.1: Extended: Trusted Update: TSF version query
	FPT_TUD_EXT.2: Extended: Trusted Update Verification
FTA: TOE access	FTA_SSL_EXT.1: Extended: TSF- and User-initiated locked state
	FTA_TAB.1: Default TOE Access Banners
	FTA_WSE_EXT.1: Extended: Wireless Network Access
FTP: Trusted path/channels	FTP_ITC_EXT.1: Extended: Trusted channel Communication

Table 1 TOE Security Functional Components

5.1.1 Cryptographic support (FCS)

5.1.1.1 Refinement: Cryptographic key generation (FCS_CKM.1(1))

FCS_CKM.1(1).1

The TSF shall generate asymmetric cryptographic keys used for key establishment in accordance with:

- NIST Special Publication 800-56B, 'Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography' for RSA-based key establishment schemes and
[***- NIST Special Publication 800-56A, 'Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography' for finite field-based key establishment schemes; ,***
- NIST Special Publication 800-56A, 'Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography' for elliptic curve-based key establishment schemes and implementing 'NIST curves' P-256, P-384 and [P-521] (as defined in FIPS PUB 186-4, 'Digital Signature Standard')]
and specified cryptographic key sizes equivalent to, or greater than, a symmetric key strength of 112 bits.

5.1.1.2 Cryptographic key generation (FCS_CKM.1(2))

FCS_CKM.1(2).1

The TSF shall generate asymmetric cryptographic keys used for authentication in accordance with a specified cryptographic key generation algorithm [
- FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3 for RSA schemes;
- FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Appendix B.4 for ECDSA schemes and implementing 'NIST curves' P-256, P-384 and [P-521];]
and specified cryptographic key sizes equivalent to, or greater than, a symmetric key strength of 112 bits.

5.1.1.3 Cryptographic key generation (FCS_CKM.1(3))

FCS_CKM.1(3).1

The TSF shall generate symmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm PRF-384 and specified cryptographic key sizes 128 bits using a Random Bit Generator as specified in FCS_RBG_EXT.1(1) that meet the following: IEEE 802.11-2012.

5.1.1.4 Cryptographic key distribution (FCS_CKM.2)

FCS_CKM.2.1

The TSF shall decrypt Group Temporal Key (GTK) in accordance with a specified cryptographic key distribution method AES Key Wrap in an EAPOL-Key frame that meets the following: NIST SP 800-38F, IEEE 802.11-2012 for the packet format and timing considerations and does not expose the cryptographic keys..

5.1.1.5 Extended: Cryptographic Key Support (FCS_CKM_EXT.1)

FCS_CKM_EXT.1.1

The TSF shall support a hardware-protected REK with an AES key of size [***256 bits***].

FCS_CKM_EXT.1.2

A REK shall not be able to be read from or exported from the hardware.

FCS_CKM_EXT.1.3

System software on the TSF shall be able only to request encryption/decryption by the key and shall not be able to read, import, or export a REK.

FCS_CKM_EXT.1.4

A REK shall be generated by a RBG in accordance with FCS_RBG_EXT.1(1).

5.1.1.6 Extended: Cryptographic Key Random Generation (FCS_CKM_EXT.2)

FCS_CKM_EXT.2.1

All DEKs shall be randomly generated with entropy corresponding to the security strength of AES key sizes of [***128, 256***] bits.

5.1.1.7 Extended: Cryptographic Key Generation (FCS_CKM_EXT.3)

FCS_CKM_EXT.3.1

All KEKs shall be [***256-bit***] keys corresponding to at least the security strength of the keys encrypted by the KEK.

FCS_CKM_EXT.3.2

The TSF shall generate KEKs by deriving the KEK from a Password Authentication Factor using PBKDF and [***a) an RBG that meets this profile (as specified FCS_RBG_EXT.1(*)),***
b) combined from other KEKs in a way that preserves the effective entropy of each factor by [encrypting one key with another]]

5.1.1.8 Extended: Key Destruction (FCS_CKM_EXT.4)

FCS_CKM_EXT.4.1

The TSF shall destroy cryptographic keys in accordance with the specified cryptographic key destruction method [***by clearing the KEK encrypting the target key, in accordance with the following rules: [- For volatile flash memory the destruction shall be executed by [a single direct overwrite consisting of zeros followed by a read-verify]]***].

FCS_CKM_EXT.4.2

The TSF shall destroy all plaintext keying material and cryptographic security parameters when no longer needed.

5.1.1.9 Extended: TSF Wipe (FCS_CKM_EXT.5)

FCS_CKM_EXT.5.1

The TSF shall wipe all protected data by [***Cryptographically erasing the encrypted DEKs and/or the KEKs in non-volatile memory by following the requirements in FCS_CKM_EXT.4.1;***]

FCS_CKM_EXT.5.2

The TSF shall perform a power cycle on conclusion of the wipe procedure.

5.1.1.10 Extended: Salt Generation (FCS_CKM_EXT.6)

FCS_CKM_EXT.6.1

The TSF shall generate all salts using a RBG that meets FCS_RBG_EXT.1(*).

5.1.1.11 Cryptographic operation (FCS_COP.1(1))

FCS_COP.1(1).1

The TSF shall perform encryption/decryption in accordance with a specified cryptographic algorithm
- AES-CBC (as defined in NIST SP 800-38A) mode,
- AES-CCMP (as defined in FIPS PUB 197, NIST SP 800-38C and IEEE 802.11-2012), and
- [***no other modes***]

and cryptographic key sizes 128-bit key sizes and [***256-bit key sizes***].

5.1.1.12 Cryptographic operation (FCS_COP.1(2))

FCS_COP.1(2).1

The TSF shall perform cryptographic hashing in accordance with a specified cryptographic algorithm SHA-1 and [***SHA-256, SHA-384, SHA-512***] and message digest sizes 160 and [***256, 384, 512 bits***] that meet the following: FIPS Pub 180-4.

5.1.1.13 Refinement: Cryptographic operation (FCS_COP.1(3))

FCS_COP.1(3).1

The TSF shall perform cryptographic signature services (generation and verification) in accordance with a specified cryptographic algorithm

- FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Section 4 for RSA schemes
[***- FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Section 5 for ECDSA schemes and implementing 'NIST curves' P-256, P-384 and [P-521]***]
and cryptographic key sizes equivalent to, or greater than, a symmetric key strength of 112 bits.

5.1.1.14 Refinement: Cryptographic operation (FCS_COP.1(4))

FCS_COP.1(4).1

The TSF shall perform keyed-hash message authentication in accordance with a specified cryptographic algorithm HMAC-SHA-1 and [***HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512***] and cryptographic key sizes [**160, 256, 384, 512-bits**] and message digest sizes 160 and [***256, 384, 512***] bits that meet the following: FIPS Pub 198-1, 'The Keyed-Hash Message Authentication Code, and FIPS Pub 180-4, 'Secure Hash Standard'.

5.1.1.15 Refinement: Cryptographic operation (FCS_COP.1(5))

FCS_COP.1(5).1

The TSF shall perform Password-based Key Derivation Functions in accordance with a specified cryptographic algorithm [HMAC-[***SHA-256***]] and output cryptographic key sizes [***256***] that meet the following: NIST SP 800-132.

5.1.1.16 HTTPS Protocol (FCS_HTTPS_EXT.1)

FCS_HTTPS_EXT.1.1

The TSF shall implement the HTTPS protocol that complies with RFC 2818.

FCS_HTTPS_EXT.1.2

The TSF shall implement HTTPS using TLS (FCS_TLS_EXT.2).

5.1.1.17 Extended: Initialization Vector Generation (FCS_IV_EXT.1)

FCS_IV_EXT.1.1

The TSF shall generate IVs in accordance with **MDFPP11** Table 11: References and IV Requirements for NIST-approved Cipher Modes.

5.1.1.18 Extended: Cryptographic Operation (Random Bit Generation) (FCS_RBG_EXT.1(1))

FCS_RBG_EXT.1(1).1

The TSF shall perform all deterministic random bit generation services in accordance with [***NIST Special Publication 800-90A using [Hash_DRBG (any), HMAC_DRBG (any), CTR_DRBG (AES)]***].

FCS_RBG_EXT.1(1).2

The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [***TSF-hardware-based noise source***] with a minimum of [***256 bits***] of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

FCS_RBG_EXT.1(1).3

The TSF shall be capable of providing output of the RBG to applications running on the TSF that request random bits.

5.1.1.19 Extended: Cryptographic Operation (Random Bit Generation) (FCS_RBG_EXT.1(2))

FCS_RBG_EXT.1(2).1

The TSF shall perform all deterministic random bit generation services in accordance with [***FIPS Pub 140-2 Annex C: X9.31 Appendix 2.4 using AES***].

FCS_RBG_EXT.1(2).2

The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [***TSF-hardware-based noise source***] with a minimum of [***128 bits***] of entropy at least equal to the

greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

FCS_RBG_EXT.1(2).3

The TSF shall be capable of providing output of the RBG to applications running on the TSF that request random bits.

5.1.1.20 Extended: Cryptographic Algorithm Services (FCS_SRV_EXT.1)

FCS_SRV_EXT.1.1

The TSF shall provide a mechanism for applications to request the TSF to perform the following cryptographic operations:
- FCS_COP.1(1)
- FCS_COP.1(2)
- FCS_COP.1(3)
- FCS_COP.1(4)
- FCS_COP.1(5)
- FCS_CKM.1(1)
- [***FCS_CKM.1(2)***].

5.1.1.21 Extended: Cryptographic Key Storage (FCS_STG_EXT.1)

FCS_STG_EXT.1.1

The TSF shall provide secure key storage for asymmetric private keys and [***no other keys***].

FCS_STG_EXT.1.2

The TSF shall be capable of importing keys/secrets into the secure key storage upon request of [***the user, the administrator***] and [***applications running on the TSF***].

FCS_STG_EXT.1.3

The TSF shall be capable of destroying keys/secrets in the secure key storage upon request of [***the user, the administrator***].

FCS_STG_EXT.1.4

The TSF shall have the capability to allow only the application that imported the key/secret the use of the key/secret. Exceptions may only be explicitly authorized by [***a common application developer***].

FCS_STG_EXT.1.5

The TSF shall allow only the application that imported the key/secret to request that the key/secret be destroyed. Exceptions may only be explicitly authorized by [***a common application developer***].

5.1.1.22 Extended: Encrypted Cryptographic Key Storage (FCS_STG_EXT.2)

FCS_STG_EXT.2.1

The TSF shall encrypt all DEKs and KEKs and [***all software-based key storage***] by KEKs that are [***1) Protected by the REK with [a. encryption by a REK] ,***
2) Protected by the REK and the password with [b. encryption by a KEK chaining to a REK and the password-derived KEK]].

FCS_STG_EXT.2.2

All keys shall be encrypted using AES in the [***CBC mode***].

5.1.1.23 Extended: Integrity of encrypted key storage (FCS_STG_EXT.3)

FCS_STG_EXT.3.1

The TSF shall protect the integrity of any encrypted KEK by [***- a keyed hash (FCS_COP.1(4)) using a key protected by a key protected by FCS_STG_EXT.2;***].

FCS_STG_EXT.3.2

The TSF shall verify the integrity of the [***hash***] of the stored key prior to use of the key.

5.1.1.24 Extended: EAP TLS Protocol (FCS_TLS_EXT.1)

FCS_TLS_EXT.1.1

The TSF shall implement the EAP-TLS protocol as specified in RFC 5216 implementing TLS 1.0 (RFC 2246) and [***no other TLS versions***] supporting the following ciphersuites:
- Mandatory Ciphersuites in accordance with RFC 3268:
o TLS_RSA_WITH_AES_128_CBC_SHA
- Optional Ciphersuites:
[***o TLS_RSA_WITH_AES_256_CBC_SHA,***
o TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
o TLS_DHE_RSA_WITH_AES_256_CBC_SHA]

FCS_TLS_EXT.1.2

The TSF shall verify that the server certificate presented for EAP-TLS [***chains to one of the specified CAs***].

5.1.1.25 TLS Protocol (FCS_TLS_EXT.2)

FCS_TLS_EXT.2.1

The TSF shall implement one or more of the following protocols TLS 1.2 (RFC 5246) and [***TLS 1.0 (RFC 2246), TLS 1.1 (RFC 4346)***] supporting the following ciphersuites:
- Mandatory Ciphersuites in accordance with RFC 3268:
o TLS_RSA_WITH_AES_128_CBC_SHA
- Optional Ciphersuites:
[***o TLS_RSA_WITH_AES_256_CBC_SHA,***
o TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
o TLS_DHE_RSA_WITH_AES_256_CBC_SHA,
o TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,
o TLS_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246,
o TLS_DHE_RSA_WITH_AES_128_CBC_ SHA256 as defined in RFC 5246,
o TLS_DHE_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246,
o TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
o TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,
o TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 6460,
o TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 6460]

FCS_TLS_EXT.2.2

The TSF shall not establish a trusted channel if the distinguished name (DN) contained in a certificate does not match the expected DN for the peer.

5.1.2 User data protection (FDP)

5.1.2.1 Extended: Security access control (FDP_ACF_EXT.1)

FDP_ACF_EXT.1.1

The TSF shall provide a mechanism to restrict the system services that are accessible to an application.

5.1.2.2 Extended: Data-At-Rest Protection (FDP_DAR_EXT.1)

FDP_DAR_EXT.1.1

Encryption shall cover all protected data.

FDP_DAR_EXT.1.2

Encryption shall be performed using DEKs with AES in the [***CBC***] mode with key size [***128, 256***] bits.

5.1.2.3 Extended: Subset information flow control (FDP_IFC_EXT.1)

FDP_IFC_EXT.1.1

The TSF shall [***provide an interface to VPN applications to enable all IP traffic (other than IP traffic required to establish the VPN connection) to flow through the IPsec VPN client***].

5.1.2.4 Extended: User Data Storage (FDP_STG_EXT.1(1))

FDP_STG_EXT.1(1).1

The TSF shall provide protected storage for the Trust Anchor Database.

5.1.3 Identification and authentication (FIA)

5.1.3.1 Authentication failure handling (FIA_AFL_EXT.1)

FIA_AFL_EXT.1.1

The TSF shall detect when a configurable positive integer within [**1-100**] of unsuccessful authentication attempts occur related to last successful authentication by that user.

FIA_AFL_EXT.1.2

When the defined number of unsuccessful authentication attempts has been [***met***], the TSF shall perform [***full wipe of all protected data***].

5.1.3.2 Extended: PAE Authentication (FIA_PAE_EXT.1)

FIA_PAE_EXT.1.1

The TSF shall conform to IEEE Standard 802.1X for a Port Access Entity (PAE) in the 'Supplicant' role.

5.1.3.3 Extended: Password Management (FIA_PMG_EXT.1)

FIA_PMG_EXT.1.1

The TSF shall support the following for the Password Authentication Factor:
1. Passwords shall be able to be composed of any combination of [***upper and lower case letters***], numbers, and special characters: [***! @ # $ % ^ & * ( ) [+ = _ / - ' " : ; , ? ` ~ | < > { } [ ]]***];
2. Password length up to [**16**] characters shall be supported.

5.1.3.4 Extended: Authentication Throttling (FIA_TRT_EXT.1)

FIA_TRT_EXT.1.1

The TSF shall limit automated user authentication attempts by [***enforcing a delay between incorrect authentication attempts***]. The minimum delay shall be such that no more than 10 attempts can be attempted per 500 milliseconds.

5.1.3.5 Protected authentication feedback (FIA_UAU.7)

FIA_UAU.7.1

The TSF shall provide only obscured feedback to the device's display to the user while the authentication is in progress.

5.1.3.6 Extended: Authentication for Cryptographic Operation (FIA_UAU_EXT.1)

FIA_UAU_EXT.1.1

The TSF shall require the user to present the Password Authentication Factor prior to decryption of protected data and keys at startup.

5.1.3.7 Timing of Authentication (FIA_UAU_EXT.2)

FIA_UAU_EXT.2.1

The TSF shall allow [***make emergency phone calls, take screen shots (stored internally), receive calls, turn TOE off, restart TOE, enable/disable airplane mode, configure sound, vibrate, or mute, display user configured widgets, take photos only, start Quick Memo application***] on behalf of the user to be performed before the user is authenticated.

FIA_UAU_EXT.2.2

The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

5.1.3.8 Extended: Re-Authentication (FIA_UAU_EXT.3)

FIA_UAU_EXT.3.1

The TSF shall require the user to enter the correct Password Authentication Factor when the user changes the Password Authentication Factor, and following TSF- and user-initiated locking in order to transition to the unlocked state, and [***no other conditions***].

5.1.3.9 Extended: Validation of certificates (FIA_X509_EXT.1)

FIA_X509_EXT.1.1

The TSF shall validate certificates in accordance with the following rules:
- RFC 5280 certificate validation and certificate path validation
- The certificate path must terminate with a certificate in the Trust Anchor Database.
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the cA flag is set to TRUE for all CA certificates.
- The TSF shall validate the revocation status of the certificate using [***a Certificate Revocation List (CRL) as specified in RFC 5759***].
- The TSF shall validate the extendedKeyUsage field according to the following rules:
o Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3).
o Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.

FIA_X509_EXT.1.2

The TSF shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.

5.1.3.10 Extended: X509 certificate authentication (FIA_X509_EXT.2)

FIA_X509_EXT.2.1

The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for EAP-TLS exchanges, and [***TLS, HTTPS***], and [***no additional uses***].

FIA_X509_EXT.2.2

When the TSF cannot establish a connection to determine the validity of a certificate, the TSF shall [***not accept the certificate***].

FIA_X509_EXT.2.3

The TSF shall not establish a trusted communication channel if the peer certificate is deemed invalid.

5.1.3.11 Extended: Request Validation of certificates (FIA_X509_EXT.3)

FIA_X509_EXT.3.1

The TSF shall provide a certificate validation service to applications.

FIA_X509_EXT.3.2

The TSF shall respond to the requesting application with the success or failure of the validation.

5.1.4 Security management (FMT)

5.1.4.1 Management of security functions behavior (FMT_MOF.1(1))

FMT_MOF.1(1).1

The TSF shall restrict the ability to perform the functions
1. enroll the TOE in management
[***5. enable/disable [personal Hotspot connections (Wi-Fi), tethered connections (USB)]***]
to the user.

5.1.4.2 Management of security functions behavior (FMT_MOF.1(2))

FMT_MOF.1(2).1

The TSF shall restrict the ability to perform the functions
1. configure password policy:
a. minimum password length
b. minimum password complexity
c. maximum password lifetime
2. configure session locking policy:
a. screen-lock enabled/disabled
b. screen lock timeout
c. number of authentication failures
3. enable/disable [**camera, microphone devices**]
4. configure application installation policy by [
a. specifying authorized application repository(s)[1],
b. specifying a set of allowed applications and versions (an application whitelist),
c. denying installation of applications]
[***6. enable/disable [Wi-Fi, GPS, NFC, Bluetooth],***
7. enable/disable data transfer capabilities over [USB, SD Card],
8. enable/disable [wireless remote access connections except for personal Hotspot service (Bluetooth tethering), personal Hotspot connections (Wi-Fi), tethered connections (USB)],
9. specify wireless networks (SSIDs) to which the TSF may connect,
10. configure security policy for each wireless network:
a. [specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s)]
b. ability to specify security type
c. ability to specify authentication protocol
d. specify the client credentials to be used for authentication
11. enable/disable developer modes,
12. enable data-at rest protection,
13. enable removable media's data-at-rest protection,
14. enable/disable local authentication bypass,
22. enable/disable device messaging capabilities,
24. enable/disable voice command control of device functions,
28. install applications,
31. [enable/disable Location services, enable/disable USB mass storage mode]]
to the administrator when the device is enrolled and according to the administrator-configured policy.

5.1.4.3 Specification of Management Functions (FMT_SMF.1)

FMT_SMF.1.1

The TSF shall be capable of performing the following management functions:
1. configure password policy:

[1] The TOE can specify an authorized application repository of Enterprise applications installed through an MDM Agent.

a. minimum password length
b. minimum password complexity
c. maximum password lifetime
2. configure session locking policy:
a. screen-lock enabled/disabled
b. screen lock timeout
c. number of authentication failures
3. enable/disable the VPN protection
4. enable/disable [**Wi-Fi, GPS, cellular, NFC, Bluetooth**]
5. enable/disable[**camera, microphone devices**]
6. specify wireless networks (SSIDs) to which the TSF may connect
7. configure security policy for each wireless network:
a. [***specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s)***]
b. ability to specify security type
c. ability to specify authentication protocol
d. specify the client credentials to be used for authentication
8. transition to the locked state
9. full wipe of protected data
10. configure application installation policy by [***a. specifying authorized application repository(s)***[2]***,***
b. specifying a set of allowed applications and versions (an application whitelist),
c. denying installation of applications],
11. import keys/secrets into the secure key storage,
12. destroy imported keys/secrets and [***no other keys/secrets***] in the secure key storage,
13. import X.509v3 certificates into the Trust Anchor Database,
14. remove imported X.509v3 certificates and [***no other X.509v3 certificates***] in the Trust Anchor Database,
15. enroll the TOE in management
16. remove applications
17. update system software
18. install applications
[***19. enable/disable data transfer capabilities over [USB, SD Card],***
20. enable/disable [wireless remote access connections except for personal Hotspot service (Bluetooth), personal Hotspot connections (Wi-Fi), tethered connections (USB)],
21. enable/disable developer modes,
22. enable data-at rest protection,
23. enable removable media's data-at-rest protection,
24. enable/disable local authentication bypass,
30. remove Enterprise applications,
33. enable/disable cellular voice functionality,
34. enable/disable device messaging capabilities,
36. enable/disable voice command control of device functions,
41. configure the unlock banner,
42. [a. enable/disable Location services
b. authenticate wired connections to the device via [passcode],
c. authenticate personal Hotspot connections to the device via [pre-shared key, passcode]
d. enable/disable USB mass storage mode]].

[2] The TOE can specify an authorized application repository of Enterprise applications installed through an MDM Agent.

5.1.4.4 Extended: Specification of Remediation Actions (FMT_SMF_EXT.1)

FMT_SMF_EXT.1.1

The TSF shall offer [***[remove Enterprise applications, remove MDM policies, and disable CC mode]***] upon unenrollment and [***no other triggers***].

5.1.5 Protection of the TSF (FPT)

5.1.5.1 Extended: Anti-Exploitation Services (ASLR) (FPT_AEX_EXT.1)

FPT_AEX_EXT.1.1

The TSF shall provide address space layout randomization (ASLR) to applications.

FPT_AEX_EXT.1.2

The base address of any user-space memory mapping will consist of at least 8 unpredictable bits.

5.1.5.2 Extended: Anti-Exploitation Services (Memory Page Permissions) (FPT_AEX_EXT.2)

FPT_AEX_EXT.2.1

The TSF shall be able to enforce read, write, and execute permissions on every page of physical memory.

5.1.5.3 Extended: Anti-Exploitation Services (Stack Overflow Protection) (FPT_AEX_EXT.3)

FPT_AEX_EXT.3.1

TSF processes that execute in a non-privileged execution domain on the application processor shall implement stack-based buffer overflow protection.

5.1.5.4 Extended: Domain Isolation (FPT_AEX_EXT.4)

FPT_AEX_EXT.4.1

The TSF shall protect itself from modification by untrusted subjects.

FPT_AEX_EXT.4.2

The TSF shall enforce isolation of address space between applications.

5.1.5.5 Extended: Key Storage (FPT_KST_EXT.1)

FPT_KST_EXT.1.1

The TSF shall not store any plaintext key material in readable non-volatile memory.

5.1.5.6 Extended: No Key Transmission (FPT_KST_EXT.2)

FPT_KST_EXT.2.1

The TSF shall not transmit any plaintext key material from the cryptographic module.

5.1.5.7 Extended: No Plaintext Key Export (FPT_KST_EXT.3)

FPT_KST_EXT.3.1

The TSF shall ensure it is not possible for the TOE user(s) to export plaintext keys.

5.1.5.8 Extended: Event Notification (FPT_NOT_EXT.1)

FPT_NOT_EXT.1.1

The TSF shall transition to non-operational mode and [***no other actions***] when the following types of failures occur:
- failures of the self tests
- TSF software integrity verification failures
- [***no other failures***].

5.1.5.9 Reliable time stamps (FPT_STM.1)

FPT_STM.1.1

The TSF shall be able to provide reliable time stamps for its own use.

5.1.5.10 Extended: TSF Cryptographic Functionality Testing (FPT_TST_EXT.1)

FPT_TST_EXT.1.1

The TSF shall run a suite of self-tests during initial start-up (on power on) to demonstrate the correct operation of all cryptographic functionality.

5.1.5.11 Extended: TSF Integrity Testing (FPT_TST_EXT.2)

FPT_TST_EXT.2.1

The TSF shall verify the integrity of the Application Processor bootloader software, Application Processor OS kernel, and [***[/system/lib/liblgdrm.so***
/system/lib/libdrmframework.so
/system/lib/libssl.so
/system/lib/libcrypto.so
/system/lib/libsurfaceflinger.so
/system/vendor/lib/drm/libdrmwvmplugin.so
/system/vendor/lib/libwvm.so
/system/vendor/lib/libwvdrm_L3.so
/system/vendor/lib/libWVStreamControlAPI_L1.so
/system/vendor/lib/libWVStreamControlAPI_L3.so
/system/app/LGApduService.apk
/system/app/LGSmartcardService.apk
/system/framework/com.lge.smartcard.jar
/system/framework/org.simalliance.openmobileapi.jar
/system/framework/bouncycastle.jar/system/etc/fota/data.dat
/system/lib/liblgkm.so
/system/lib/libccm.so
/system/lib/libsecureks.so]], stored in mutable media prior to its execution through the use of [***a digital signature using a hardware-protected asymmetric key***].

5.1.5.12 Extended: Trusted Update: TSF version query (FPT_TUD_EXT.1)

FPT_TUD_EXT.1.1

The TSF shall provide authorized users the ability to query the current version of the TOE firmware/software.

FPT_TUD_EXT.1.2

The TSF shall provide authorized users the ability to query the current version of the hardware model of device.

FPT_TUD_EXT.1.3

The TSF shall provide authorized users the ability to query the current version of installed mobile applications.

5.1.5.13 Extended: Trusted Update Verification (FPT_TUD_EXT.2)

FPT_TUD_EXT.2.1

The TSF shall verify software updates to the TSF using a digital signature by the manufacturer prior to installing those updates.

FPT_TUD_EXT.2.2

The boot integrity [***key***]] shall only be updated by verified software.

FPT_TUD_EXT.2.3
The digital signature verification key shall [***match a hardware-protected public key***].

FPT_TUD_EXT.2.4
The TSF shall verify mobile application software using a digital signature mechanism prior to installation.

5.1.6 TOE access (FTA)

5.1.6.1 Extended: TSF- and User-initiated locked state (FTA_SSL_EXT.1)

FTA_SSL_EXT.1.1
The TSF shall transition to a locked state after a time interval of inactivity and a user initiated lock, and upon transitioning to the locked state, the TSF shall perform the following operations:
a) clearing or overwriting display devices, obscuring the previous contents;
b) [**no other actions**].

5.1.6.2 Default TOE Access Banners (FTA_TAB.1)

FTA_TAB.1.1
Before establishing a user session, the TSF shall display an Administrator-specified advisory notice and consent warning message regarding use of the TOE.

5.1.6.3 Extended: Wireless Network Access (FTA_WSE_EXT.1)

FTA_WSE_EXT.1.1
The TSF shall be able to attempt connections to wireless networks specified as acceptable networks as configured by the administrator in FMT_SMF.1.

5.1.7 Trusted path/channels (FTP)

5.1.7.1 Extended: Trusted channel Communication (FTP_ITC_EXT.1)

FTP_ITC_EXT.1.1
The TSF shall use 802.11-2012, 802.1X, and EAP-TLS and [***TLS, HTTPS protocol***] to provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure and detection of modification of the channel data.

FTP_ITC_EXT.1.2
The TSF shall permit the TSF and applications to initiate communication via the trusted channel.

FTP_ITC_EXT.1.3
The TSF shall initiate communication via the trusted channel for connection to a wireless access point and [**no other communications**].

5.2 TOE Security Assurance Requirements

The SARs for the TOE are the EAL 1 augmented with ALC_TSU_EXT.1 components as specified in Part 3 of the Common Criteria. Note that the SARs have effectively been refined with the assurance activities explicitly defined in association with both the SFRs and SARs.

Requirement Class	Requirement Component
ADV: Development	ADV_FSP.1: Basic functional specification
AGD: Guidance documents	AGD_OPE.1: Operational user guidance
	AGD_PRE.1: Preparative procedures

ALC: Life-cycle support	ALC_CMC.1: Labelling of the TOE
	ALC_CMS.1: TOE CM coverage
	ALC_TSU_EXT.1: Timely Security Updates
ATE: Tests	ATE_IND.1: Independent testing - conformance
AVA: Vulnerability assessment	AVA_VAN.1: Vulnerability survey

Table 2 EAL 1 augmented with ALC_TSU_EXT.1 Assurance Components

5.2.1 Development (ADV)

5.2.1.1 Basic functional specification (ADV_FSP.1)

ADV_FSP.1.1d
The developer shall provide a functional specification.

ADV_FSP.1.2d
The developer shall provide a tracing from the functional specification to the SFRs.

ADV_FSP.1.1c
The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.2c
The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.3c
The functional specification shall provide rationale for the implicit categorisation of interfaces as SFR-non-interfering.

ADV_FSP.1.4c
The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

ADV_FSP.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.1.2e
The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

5.2.2 Guidance documents (AGD)

5.2.2.1 Operational user guidance (AGD_OPE.1)

AGD_OPE.1.1d
The developer shall provide operational user guidance.

AGD_OPE.1.1c
The operational user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

AGD_OPE.1.2c
The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.

AGD_OPE.1.3c
The operational user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.

AGD_OPE.1.4c
The operational user guidance shall, for each user role, clearly present each type of security-

relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_OPE.1.5c

The operational user guidance shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

AGD_OPE.1.6c

The operational user guidance shall, for each user role, describe the security measures to be followed in order to fulfil the security objectives for the operational environment as described in the ST.

AGD_OPE.1.7c

The operational user guidance shall be clear and reasonable.

AGD_OPE.1.1e

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.2.2 Preparative procedures (AGD_PRE.1)

AGD_PRE.1.1d

The developer shall provide the TOE including its preparative procedures.

AGD_PRE.1.1c

The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.

AGD_PRE.1.2c

The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

AGD_PRE.1.1e

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_PRE.1.2e

The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared securely for operation.

5.2.3 Life-cycle support (ALC)

5.2.3.1 Labelling of the TOE (ALC_CMC.1)

ALC_CMC.1.1d

The developer shall provide the TOE and a reference for the TOE.

ALC_CMC.1.1c

The TOE shall be labelled with its unique reference.

ALC_CMC.1.1e

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3.2 TOE CM coverage (ALC_CMS.1)

ALC_CMS.2.1d

The developer shall provide a configuration list for the TOE.

ALC_CMS.2.1c

The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.

ALC_CMS.2.2c

The configuration list shall uniquely identify the configuration items.

ALC_CMS.2.1e

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3.3 Timely Security Updates (ALC_TSU_EXT.1)

ALC_TSU_EXT.1.1d

The developer shall provide a description in the TSS of how timely security updates are made to the TOE.

ALC_TSU_EXT.1.1c

The description shall include the process for creating and deploying security updates for the TOE software/firmware.

ALC_TSU_EXT.1.2c

The description shall express the time window as the length of time, in days, between public disclosure of a vulnerability and the public availability of security updates to the TOE.

ALC_TSU_EXT.1.3c

The description shall include the mechanisms publicly available for reporting security issues pertaining to the TOE.

ALC_TSU_EXT.1.1e

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4 Tests (ATE)

5.2.4.1 Independent testing - conformance (ATE_IND.1)

ATE_IND.1.1d

The developer shall provide the TOE for testing.

ATE_IND.1.1c

The TOE shall be suitable for testing.

ATE_IND.1.1e

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.1.2e

The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

5.2.5 Vulnerability assessment (AVA)

5.2.5.1 Vulnerability survey (AVA_VAN.1)

AVA_VAN.1.1d

The developer shall provide the TOE for testing.

AVA_VAN.1.1c

The TOE shall be suitable for testing.

AVA_VAN.1.1e

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VAN.1.2e

The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.

AVA_VAN.1.3e

The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.

6. TOE Summary Specification

This chapter describes the security functions:

- Cryptographic support
- User data protection
- Identification and authentication
- Security management
- Protection of the TSF
- TOE access
- Trusted path/channels

6.1 Cryptographic support

FCS_CKM.1(1): The TOE provides generation of asymmetric keys used for key establishment as part of the TLS protocol (utilized by the TOE during WPA2 with EAP-TLS as well as provided to mobile applications utilizing Android APIs to establish secure TLS and HTTPS connections). The TOE supports generation of RSA, Diffie-Hellman, and Elliptic Curve keys for use in the TLS cipher suites requiring use of those keys. The TOE's cryptographic algorithm implementations have received NIST algorithm certificates (please see the table in FCS_COP.1) and generally fulfills all of the NIST SP 800-56A and 800-56B requirements without extensions.

FCS_CKM.1(2): The TOE supports generation of asymmetric keys used for authentication. The TOE itself does not generate any public/private key pairs for TOE functionality (the user or administrator must load certificates for use with WPA2); however, the TOE provides key generation APIs to mobile applications to allow them to generate keys. The TOE's cryptographic algorithm implementations have received NIST algorithm certificates (please see the table in FCS_COP.1).

FCS_CKM.1(3): The TOE adheres to 802.11-2012 with regards to 802.11i key generation. The TOE's wpa_supplicant provides the PRF384 for WPA2 generation of AES 128-bit. The TOE has been designed for compliance, and the Device has successfully completed certification (including WPA2 Enterprise) and received a WiFi CERTIFIED Interoperability Certificate from the WiFi Alliance. The WiFi Alliance maintains a website providing further information about the testing program: http://www.wi-fi.org/certification

FCS_CKM.2: The TOE adheres to RFC 3394, SP 800-38F, and 802.11-2012 standards and unwraps the GTK (sent encrypted with the WPA2 KEK using AES Key Wrap in an EAPOL-Key frame). The TOE, upon receiving an EAPOL frame, will subject the frame to a number of checks (frame length, EAPOL version, frame payload size, EAPOL-Key type, key data length, EAPOL-Key CCMP descriptor version, and replay counter) to ensure a proper EAPOL message and then decrypt the GTK using the KEK, thus ensuring that it does not expose the Group Temporal Key (GTK).

FCS_CKM_EXT.1: The TOE includes a Root Encryption Key (REK) derived from a 256-bit fuse value residing in the application processor. The application processor protects the fuse value by preventing any direct observation of the value and any ability to modify or update the value. The application processor stores the fuse value only in a hardware register and allows only the Trusted Execution Environment (TEE) to request use of the value for encryption, decryption or key derivation. Furthermore, only signed trusted applications executing in the TEE can call into the TEE in order to derive a key from the REK.

The TOE derives this REK from the 256-bit fuse value and then allows use of the REK for encryption and decryption as part of the TOE key hierarchy. The TOE generates the fuse value during manufacturing using its hardware DRBG. More information regarding Trusted Execution Environments may be found here: http://www.globalplatform.org/mediaguidetee.asp

FCS_CKM_EXT.2: The TOE utilizes its approved RBGs to generate DEKs. When generating AES keys for itself (for example, the TOE's own Data-At-Rest encryption key or for Secure Key Storage), the TOE utilizes the

RAND_bytes() API call from its OpenSSL AES-256 CTR_DRBG to generate a 256-bit AES key. The TOE also utilizes that same DRBG when servicing API requests from mobile applications wishing to generate AES keys (either 128 or 256-bit).

When generating keys used to encrypt files stored on the SD Card, the TOE utilizes the fips_cprng_get_random API from its kernel implementation of an ANSI X9.31 RBG to generate 128-bit AES keys.

In all cases, the TOE generates DEKs using a compliant RBG seeded with sufficient entropy so as to ensure that the generated key cannot be recovered with less work than a full exhaustive search of the key space.

FCS_CKM_EXT.3: Refer to the Key Hierarchy table. The TOE derives all password-based KEKs by combining the user's password with an RBG generated salt value using PBKDFv2 (in accordance with FCS_COP.1(5)). The TOE generates all non-derived KEK using the RAND_bytes() API call from its OpenSSL AES-256 CTR_DRBG to ensure a full 256-bits of strength. And the TOE combines KEKs by encrypting one KEK with the other so as to preserve entropy.

FCS_CKM_EXT.4: The TOE clears sensitive cryptographic material (plaintext keys, authentication data, other security parameters) from memory when no longer needed. No plaintext cryptographic material resides in the TOE's Flash as the TOE encrypts all keys stored in Flash. When performing a full wipe of protected data, the TOE cryptographically erases the protected data by clearing the Data-At-Rest DEK. Because the TOE's keystore resides within the user data partition, TOE effectively cryptographically erases those keys when clearing the Data-At-Rest DEK. In turn, the TOE clears the Data-At-Rest DEK, SD Card SDEK, and Secure Key Storage SEK through a direct overwrite (BLKDISCARD ioctl) of the Flash memory containing the key followed by a read-verify.

FCS_CKM_EXT.5: The TOE stores all protected data in encrypted form within the user data partition. Upon request, the TOE cryptographically erases the Data-At-Rest DEK protecting the user data partition, clears that key from memory, reformats the partition, and then reboots. The TOE's clearing of the Data-At-Rest DEK follows the requirements of FCS_CKM_EXT.4.

FCS_CKM_EXT.6: The TOE generates salt values using is AES-256 CTR_DRBG.

FCS_COP.1: The TOE performs cryptographic algorithms in accordance with the following NIST standards and has received the following CAVP algorithm certificates.

The OpenSSL FIPS Object Module provides the following algorithms

Algorithm	NIST Standard	SFR Reference	Cert#
AES 128/256 CBC, CCM, GCM, KW	FIPS 197, SP 800-38A/C/D/F	FCS_COP.1(1)	3011
CVL ECC CDH P-224/256/384/521	SP 800-56A	FCS_CKM.1(1)	366
DRBG Hash/HMAC/CTR	SP 800-90A	FCS_RBG_EXT.1(1)	573
ECDSA PKG/PKV/SigGen/SigVer	FIPS 186-4	FCS_CKM.1(1) FCS_CKM.1(2) FCS_COP.1(3)	551
HMAC SHA-1/256/384/512	FIPS 198-1 & 180-4	FCS_COP.1(4)	1903
RSA SIG(gen)/SIG(ver)/Key(gen)	FIPS 186-4	FCS_CKM.1(1) FCS_CKM.1(2) FCS_COP.1(3)	1571
SHS SHA-1/256/384/512	FIPS 180-4	FCS_COP.1(2)	2519

Table 3 OpenSSL Cryptographic Algorithms

The Bouncy Castle JCE provider provides the following algorithms

Algorithm	NIST Standard	SFR Reference	Cert#
AES 128/256 CBC	FIPS 197, SP 800-38A/C/D/F	FCS_COP.1(1)	3013
HMAC SHA-1/256/384/512	FIPS 198-1 & 180-4	FCS_COP.1(4)	1905
RNG ANSI X9.31 AES-128	ANSI X9.31 & 931rngext.pdf	FCS_RBG_EXT.1(2)	1308
RSA SIG(gen)/SIG(ver)/Key(gen)	FIPS 186-4	FCS_CKM.1(1) FCS_CKM.1(2)	1572

Algorithm	NIST Standard	SFR Reference	Cert#
		FCS_COP.1(3)	
SHS SHA-1/256/384/512	FIPS 180-4	FCS_COP.1(2)	2521

Table 4 Bouncy Castle Cryptographic Algorithms

The Linux kernel provides the following cryptographic algorithms

Algorithm	NIST Standard	SFR Reference	Cert#
AES 128/256 CBC	FIPS 197, SP 800-38A	FCS_COP.1(1)	3012
HMAC SHA-1/256/384/512	FIPS 198-1 & 180-4	FCS_COP.1(4)	1904
RNG ANSI X9.31 AES-128	ANSI X9.31 & 931rngext.pdf	FCS_RBG_EXT.1(2)	1307
SHS SHA-1/256/384/512	FIPS 180-4	FCS_COP.1(2)	2520

Table 5 Kernel Cryptographic Algorithms

The TOE's application processor includes a source of hardware entropy that the TOE distributes throughout, and the TOE's RBGs make use of that entropy when seeding/instantiating themselves.

The TOE conditions the user's password exactly as per SP 800-132 (and thus as per SP 800-197-1) with no deviations by using PBKDF2 with 8192 HMAC-SHA-256 iterations to combine a 128-bit salt with the user's password.

In addition to utilizing HMAC as part of PBKDF2, the TOE uses HMAC as part of the TLS ciphersuites. For TLS, the TOE uses HMAC using SHA-1 (with a 160-bit key) to generate a 160-bit MAC, using SHA-256 (with a 256-bit key) to generate a 256-bit MAC, and using SHA-384 (with a 384-bit key) to generate a 384-bit MAC. FIPS 198-1 & 180-4 dictate the block size used, and they specify block sizes of 512, 512, and 1024-bits for HMAC-SHA-1, HMAC-SHA-256, and HMAC_SHA-384 respectively.

FCS_HTTPS_EXT.1: The TOE support the HTTPS protocol (compliant with RFC 2818) so that (mobile and system) applications executing on the TOE can act as HTTPS clients and securely connect to external servers using HTTPS. Administrators have no credentials and cannot use HTTPS or TLS to establish administrative sessions with the TOE as the TOE does not provide any such capabilities.

FCS_IV_EXT.1: The TOE generates IVs using its AES-256 CTR_DRBG for use with all keys other than the FEK keys used to encrypt files located on the SD Card. For FEKs, the TOE generates the IVs using its X9.31 AES-128 RBG. In all cases, the TOE uses AES-CBC mode and generates the IVs in compliance with the requirements of table H of MDFPP11.

FCS_RBG_EXT.1(*): The TOE provides a number of different RBGs including

1. A SHA-256 Hash_DRBG provided in the hardware of the Application Processor.
2. RBGs provided by OpenSSL: Hash_DRBG (using any size SHA), HMAC_DRBG (again using size SHA), and CTR_DRBG (using AES). Note that while the TOE includes implementations of three DRBG variants (and supports all options within each variant), the TOE (and its current system level applications) make use of only an AES-256 CTR_DRBG. Furthermore The TOE provides mobile applications access (through an Android Java API) to random data drawn from its AES-256 CTR_DRBG. However, future (system-level) applications could utilize other DRBG variants supported by the TOE's OpenSSL module.
3. And an ANSI X9.31 AES-128 RBG provided by the Linux kernel

The TOE initializes each RBG with sufficient entropy ultimately accumulated from a TOE-hardware-based noise source (see the Entropy Assessment Report for more details). The TOE uses its hardware-based noise source to continuously fill /dev/random, and in turn, the TOE draws from /dev/random to seed both its AES-256 CTR_DRBG and ANSI X9.31 AES-128 RBG. The TOE seeds its AES-256 CTR_DRBG using 384-bits of data from /dev/random, thus ensuring at least 256-bits of entropy. Finally the TOE seeds its X9.31 AES-128 with 256-bits of data from /dev/random, also ensuring that it contains at least 128-bits of entropy.

FCS_SRV_EXT.1: The TOE provides applications access to the cryptographic operations including encryption (AES), hashing (SHA), signing and verification (RSA & ECDSA), key hashing (HMAC), password-based key-derivation functions (PKBDFv2 HMAC-SHA-256), generation of asymmetric keys for key establishment (RSA, DH, and ECDH), and generation of asymmetric keys for signature generation and verification (RSA, ECDSA). The TOE provides access through the Android operating system's Java API, through the native OpenSSL API, and through the kernel.

FCS_STG_EXT.1: The TOE provides the user, administration, and mobile applications the ability to import and use asymmetric public and private keys into the TOE's software-based Secure Key Storage. Additionally, the user and administrator can request the TOE to destroy the keys stored in the Secure Key Storage. While normally mobile applications cannot use or destroy the keys of another application, applications that share a common application developer (and are thus signed by the same developer key) may do so. In other words applications with a common developer may use and destroy each other's keys located within the Secure Key Storage.

FCS_STG_EXT.2: The TOE employs a key hierarchy that protects all DEKs and KEKs by encryption with either the REK or by the REK and password derived KEK.

For the DAR_KEK, SD_KEK, and SKS_KEK, the TOE uses the REK to directly encrypt these keys.

When encrypting the DEKs for Data-At-Rest, SD Card Data-At-Rest encryption, and Secure Key Storage, the TOE uses KEKs formed by combining KEKs encrypted by REK (the aforementioned DAR_KEK, SD_KEK, and SKS_KEK) with KEKs derived from the user's password (KEK_1, KEK_3, and KEK_5).

The FEKs used for SD Card Data-At-Rest encryption are the last type of DEK, and the TOE uses the SD Card Data-At-Rest DEK to encrypt the FEKs. Thus, the TOE encrypts the FEKs with a KEK that is encrypted by a KEK chaining to a REK and the password-derived KEK.

All keys other than the FEKs (which are 128-bit keys) are 256-bits in size. All non-derived keys are generated using the TOE's AES-256 CTR_DRBG, while the FEKs are generated using the TOE's X9.31 AES-128 RBG, and the TOE encrypts its keys using AES-CBC. By utilizing only 256-bit KEKs, the TOE ensures that all keys are encrypted by an equal or larger sized key (put another way, the 128-bit FEKs are encrypted by the 256-bit SD Data-At-Rest DEK, SDEK, and all other DEKs and KEKs are encrypted with 256-bit KEKs).

In the case of WiFi, the TOE utilizes the 802.11-2012 KCK and KEK keys to unwrap (decrypt) the WPA2 Group Temporal Key received from the access point. Additionally, the TOE protects persistent Wifi keys (user certificates and private keys) by storing them in the Secure Key Store.

FCS_STG_EXT.3: The TOE protects the integrity of KEK stored in Flash by verifying the HMAC-SHA-256 checksum of the decrypted key. The TOE utilizes the encryption key as the HMAC secret key, and stores the resulting HMAC with the encrypted key in Flash. Thus, the TOE protects the HMAC key as required by FCS_STG_EXT.2

FCS_TLS_EXT.1: The TSF supports TLS versions 1.0 and supports the four selected ciphersuites utilizing SHA (see the selections in section 5.1.1.24) for use with EAP-TLS as part of WPA2. The TOE does not support any other ciphersuites with EAP-TLS, as the remainder of selectable cipher suites are based on SHA256, which requires TLS v1.2.

FCS_TLS_EXT.2: The TOE provides mobile applications (through its Android API) the use of TLS versions 1.0, 1.1, and 1.2 including support for all twelve selectable ciphersuites (see the selections in section 5.1.1.25).

When an application uses the APIs provided in the User Guidance to attempt to establish a trusted channel connection based on TLS or HTTPS, if the distinguished name (DN) contained within the peer certificate does not match the expected DN for the peer, then the TOE will not establish the connection.

6.2 User data protection

FDP_ACF_EXT.1: The TOE provides the following categories of system services to applications.

1. Signature or System – these are privileged services that the system grants only to applications that are in the Android system image or that are signed with the same certificate as the application that declared the permission.

2. Signature – these are privileged services that the system grants only if the requesting application is signed with the same certificate as the application that declared the permission. If the certificates match, the system automatically grants the permission without notifying the user or asking for the user's explicit approval.
3. Dangerous – these are higher-risk services that would give a requesting application access to private user data or control over the device that can negatively impact the user.
4. Normal – these are lower-risk services that the system automatically grants to a requesting application at installation, without asking for the user's explicit approval (though the user always has the option to review these permissions before installing)

An example of a System or Signature privilege would be access to the management APIs, for example the ability require device SD Card storage encryption (com.lge.permission.SDENCRYPTION). Such permissions are conferred based upon the applications signature (of by virtue of it being part of the system image). An example of a Manifest Permission privilege would be access to location services (android.permission.ACCESS_FINE_LOCATION). An example of a Normal privilege is the ability to change the background image shown to the user.

The TOE controls access to Manifest Permission privileges during the installation of the application. The TOE prompts the user to review the application's requested privileges and if the user approved, to continue with installation of the application. Thereafter, the TOE associates a set of permissions with that application to govern its subsequent execution.

FDP_DAR_EXT.1: The TOE provides Data-At-Rest AES-256 CBC encryption for all data stored on the TOE in the user data partition (which includes both user data and TSF data). The TOE also has TSF data relating to key storage for TSF keys not stored in the system's Secure Key Store. The TOE separately encrypts those TSF keys and data. Additionally, the TOE includes a read-only filesystem in which the TOE's system executables, libraries, and their configuration data reside. For its Data-At-Rest encryption of the data partition on the internal Flash (where the TOE stores all user data and all application data), the TOE uses an AES-256 bit DEK with CBC feedback mode to encrypt the entire partition. The TOE also provides AES-128 CBC encryption of protected data stored on the external SD Card using FEKs. The TOE encrypts each individual file stored on the SD Card, generating a unique FEK for each file.

FDP_IFC_EXT.1: The TOE will route all traffic other than traffic necessary to establish the VPN connection to the VPN gateway (when the gateway's configuration specifies so). The TOE includes an interceptor kernel module that controls inbound and output packets. When a VPN is active, the interceptor will route all incoming packets to the VPN and conversely route all outbound packets to the VPN before they are output.

FDP_STG_EXT.1: The TOE's Trusted Anchor Database consists of the built-in certs (individually stored in /system/etc/security/ and which can be disabled through the TOE's Android user interface [Settings->Security-> Trusted Credentials]) and any additional user or admin/MDM loaded certificates . The built-in ones are protected as they are part of the TSF's read only system partition, and the TOE protects user-loaded certs by storing them encrypted in the Secure Key Storage.

6.3 Identification and authentication

FIA_AFL_EXT.1: The TOE maintains, for each user, the number of failed logins since the last successful login, and upon reaching the maximum number of incorrect logins, the TOE performs a full wipe of all protected data (and in fact, wipes all user data). An administrator can adjust the number of failed logins from the default of ten failed logins to a value between one and one hundred through an MDM.

FIA_PAE_EXT.1: The TOE can join WPA2-802.1X (802.11i) wireless networks requiring EAP-TLS authentication, acting as a client/supplicant (and in that role connect to the 802.11 access point and communicate with the 802.1X authentication server).

FIA_PMG_EXT.1: The TOE authenticates user through a password consisting of basic Latin characters (upper and lower case, numbers, and the special characters noted in the selection (see section 5.1.3.3). The TOE defaults to requiring passwords to have a minimum of four characters but no more than sixteen, contain at least one letter; However, an MDM application can change these defaults.

FIA_TRT_EXT.1: The TOE allows users to authenticate using a USB or Bluetooth keyboard (and thus allows users to authenticate through an external port). However, whether using an external keyboard or the standard User Interface (the TOE's touchscreen), the TOE limits the number of authentication attempts through the UI to no more than five attempts within 30 seconds. Thus if the current [the n^{th}] and prior four authentication attempts have failed and the n-4^{th} attempt was less than 30 second ago, the TOE will prevent any further authentication attempts until the user has typed the phrase "life is good" and hit enter. Note as well that the TOE will wipe itself when it reaches the maximum number of unsuccessful authentication attempts (as described in FIA_AFL_EXT.1 above). For example, when the maximum failed authentication attempts is set to 10, after reaching five failed authentication attempts, the TOE displays a warning dialog requiring the user to type 'Life is good' in order to continue authentication attempts, and then, if the maximum count (10) of failed attempts is reached, the TOE will wipe itself.

FIA_UAU.7: The TOE allows the user to enter the user's password from the lock screen. The TOE will, by default, display the most recently entered character of the password briefly or until the user enters the next character in the password, at which point the TOE obscures the character by replacing the character with a dot symbol.

FIA_UAU_EXT.1: As described before, the TOE's key hierarchy requires the user's password in order to derive the KEK_* keys in order to decrypt other KEKs and DEKs. Thus, until it has the user's password, the TOE cannot decrypt the DEK utilized for Data-At-Rest encryption, and thus cannot decrypt the user's protected data.

FIA_UAU_EXT.2: The TOE, when configured to require a user password (as is the case in CC mode), the TOE allows a user to make an emergency call, take screen shots (stored internally), turn the TOE off, restart the TOE, enable or disable airplane mode, configure sound/vibrate/mute, view user configured widgets, receive an incoming phone calls, take photos, or start the Quick Memo Application without first successfully authenticating. Beyond those actions, a user cannot perform any other actions other than observing notifications displayed on the lock screen until after successfully authenticating. Note that the TOE automatically names and saves (to the internal Flash) any screen shots or photos taken from the lock screen. The TOE provides the user no opportunity to name them or change where they are stored.

FIA_UAU_EXT.3: The TOE requires the user to enter their password in order to unlock the TOE. Additionally the TOE requires the user to confirm their current password when accessing the "Settings->Display->LockScreen->Screen Security->Select screen lock" menu in the TOE's user interface. Only after entering their current user password can the user then elect to change their password.

FIA_X509_EXT.1: The TOE checks the validity of all imported CA certificates by checking for the presence of the basicConstraints extension and that the CA flag is set to TRUE as the TOE imports the certificate. Additionally the TOE verifies the extendedKeyUsage Server Authentication purpose during WPA2/EAP-TLS negotiation. The TOE's certificate validation algorithm examines each certificate in the path (starting with the peer's certificate) and first checks for validity of that certificate (e.g., has the certificate expired or it not yet valid, whether the certificate contains the appropriate X.509 extensions [e.g., the CA flag in the basic constraints extension for a CA certificate, or that a server certificate contains the Server Authentication purpose in the ExtendedKeyUsagefield], then verifies each certificate in the chain (applying the same rules as above, but also ensuring that the Issuer of each certificate matches the Subject in the next rung "up" in the chain and that the chain ends in a self-signed certificate present in either the TOE's trusted anchor database or matches a specified Root CA), and finally the TOE performs revocation checking for all certificates in the chain..

FIA_X509_EXT.2: The TOE uses X.509v3 certificates as part of EAP-TLS, TLS and IPsec authentication. The TOE comes with a built-in set of default of Trusted Credentials (Android's set of trusted CA certificates). And while the user cannot remove any of the built-in default CA certificates, the user can disable any of those certificates through the user interface so that certificates issued by disabled CA's cannot validate successfully. In addition, a user and an administrator/MDM can import a new trusted CA certificate into the Trust Anchor Database (the TOE stores the new CA certificate in the Security Key Store).

The TOE does not establish TLS connections itself (beyond EAP-TLS used for WPA2 Wifi connections), but provides a series of APIs that mobile applications can use to check the validity of a peer certificate. The mobile application, after correctly using the specified APIs can be assured as to the validity of the peer certificate and be assured that the TOE will not establish the trusted connection if the peer certificate cannot be verified (including validity, certification path, and revocation [through CRL]). If, during the process of certificate verification, the TOE

cannot establish a connection with the server acting as the CRL Distribution Point (CDP), the TOE will deem the server's certificate as invalid and not establish a TLS connection with the server.

FIA_X509_EXT.3: The TOE's Android operating system provides applications the CertPathValidator Java API Class of methods validating certification paths (certificate chains) establishing a trust chain from a certificate to a trust anchor.

6.4 Security management

FMT_MOF.1(1): The TOE grants the user the exclusive functions to enroll the TOE in a management (effectively granting Device Administration permissions to a mobile application) and to enable and disable tethering (Bluetooth, Wi-Fi, and USB).

FMT_MOF.1(2): When the TOE had enrolled in Device Management, the TOE grants the administrator the exclusive ability to perform the management functions specified in the SFR selections (see section 5.1.4.2). It is worth noting that the TOE's ability to specify authorized application repositories takes the form of allowing enterprise applications (i.e., restricting applications to only those applications installed by an MDM Agent).

FMT_SMF.1: The TOE provides the capability to perform all the management functions specified in the SFR selections (see section 5.1.4.3). The TOE provides a configurable banner that it will display to the user at power-up. After being unlocked, the TOE will no longer display any banner while in the lock state.

FMT_SMF_EXT.1: The TOE offers MDM agents the ability to remove Enterprise applications, to remove MDM policies, and to disable CC mode upon unenrollment..

6.5 Protection of the TSF

FPT_AEX_EXT.1: The Linux kernel of the TOE's Android operating system provides address space layout randomization utilizing the get_random_int(void) kernel random function to provide eight unpredictable bits to the base address of any user-space memory mapping. The random function, though not cryptographic, ensures that one cannot predict the value of the bits.

FPT_AEX_EXT.2: The TOE's Android 4.4 operating system utilizes a 3.4 Linux kernel, whose memory management unit (MMU) enforces read, write, and execute permissions on all pages of virtual memory. The Android operating system (as of Android 2.3) sets the ARM No eXecute (XN) bit on memory pages and the TOE's ARMv7 Application Processor's Memory Management Unit (MMU) circuitry enforces the XN bits. From Android's documentation (https://source.android.com/devices/tech/security/index.html), Android 2.3 forward supports "Hardware-based No eXecute (NX) to prevent code execution on the stack and heap". Section B4.2 of the ARM v7 Architecture Reference Manual contains additional details about the MMU of ARM-based processors: http://www.club.cc.cmu.edu/~mjrosenb/ARM%20v7%20Architecture%20Reference%20Manual.pdf

FPT_AEX_EXT.3: The TOE's Android operating system provides explicit mechanisms to prevent stack buffer overruns in addition to taking advantage of hardware-based No eXecute to prevent code execution on the stack and heap. Specifically, the vendor builds the TOE (Android and support libraries) using gcc–fstack-protector compile option to enable stack overflow protection and Android takes advantage of ARM v6 eXecute-Neve to make the stack and heap non-executable. The vendor applies these protections to all TSF executable binaries and libraries.

FPT_AEX_EXT.4: The TOE protects itself from modification by untrusted subjects using a variety of methods. The first protection employed by the TOE is a Secure Boot process that uses cryptographic signatures to ensure the authenticity and integrity of the bootloader and kernels using data fused into the device processor.

The TOE protects its REK by limiting access to only trusted applications within the TEE (Trusted Execution Environment). The TOE key manager includes a TEE module which utilizes the REK to protect all other keys in the key hierarchy. All TEE applications are cryptographically signed, and when invoked at runtime (at the behest of an untrusted application), the TEE will only load the trusted application after successfully verifying its cryptographic signature.

Additionally, the TOE's Android operating system provides "sandboxing" that ensures that each third-party mobile application executes with the file permissions of a unique Linux user ID, in a different virtual memory space. This

ensures that applications cannot access each other's memory space or files and cannot access the memory space or files of other applications (notwithstanding access between applications with a common application developer).

FPT_KST_EXT.1: The TOE does not store any plaintext key material in its internal Flash or on external SD Cards; instead, the TOE encrypts all keys before storing them. This ensures that irrespective of how the TOE powers down (e.g., a user commands the TOE to power down, the TOE reboots itself, or battery depletes or is removed), all keys in internal Flash or on an external SD Card are wrapped with a KEK. Please refer to section 6.1of the TSS for further information (including the KEK used) regarding the encryption of keys stored in the internal Flash and on external SD Cards. As the TOE encrypts all keys stored in Flash, upon boot-up, the TOE must first decrypt any keys in order to utilize them.

FPT_KST_EXT.2: The TOE itself (i.e., the mobile device) comprises a cryptographic module that utilizes cryptographic libraries including OpenSSL, kernel cryptography, Bouncy Castle JCE, and the following system-level executables that utilize KEKs: dm-crypt, eCryptfs, wpa_supplicant, and the Secure Key Store.

The TOE ensures that plaintext key material does not leave this cryptographic module by only allowing the system-level executables access to the plaintext KEK values that protect all other keys in the TOE. The TSF software (the system-level executables) protects the plaintext KEKs and any plaintext DEK values in memory both by not providing any access to these values and by clearing them when no longer needed (in compliance with FCS_CKM.4).

FPT_KST_EXT.3: The TOE does not provide any way to export plaintext DEKs or KEKs (including all keys stored in the Secure Key Store) as the TOE chains or directly encrypts all KEKs to the REK.

FPT_NOT_EXT.1: When the TOE encounters a critical failure (either a self-test failure or TOE software integrity verification failure), the TOE transitions to a non-operational mode. The user may attempt to power-cycle the TOE to see if the failure condition persists, and if it does persist, the user may attempt to boot to the recovery mode/kernel to wipe data and perform a factory reset in order to recover the device.

FPT_STM.1: The TOE requires time for the Package Manager, image verifier, wpa_supplicant, and Secure Key Store applications. These TOE components obtain time from the TOE using system API calls [e.g., time() or gettimeofday()]. An application cannot modify the system time as mobile applications need the android "SET_TIME" permission to do so. Likewise, only a process with root privileges can directly modify the system time using system-level APIs. The TOE uses the Cellular Carrier time (obtained through the Carrier's network time server) as a trusted source; however, the user can also manually set the time through the TOE's user interface.

FPT_TST_EXT.1: The TOE automatically performs known answer power on self-tests (POST) on its cryptographic algorithms to ensure that they are functioning correctly. Each component providing cryptography (the kernel, OpenSSL, and Bouncy Castle JCE) perform known answer tests on their cryptographic algorithms to ensure they are working correctly. Should any of the tests fail, the TOE displays an error message stating "Boot Failure" and halts the boot process, at which point a user can attempt to power cycle the phone to see if that will clear the error.

Algorithm	Implemented in	Description
AES encryption/decryption	OpenSSL	Comparison of known answer to calculated valued
ECDH key agreement	OpenSSL	Comparison of known answer to calculated valued
DRBG random bit generation	OpenSSL	Comparison of known answer to calculated valued
ECDSA sign/verify	OpenSSL	Comparison of known answer to calculated valued
HMAC-SHA	OpenSSL	Comparison of known answer to calculated valued
RSA sign/verify	OpenSSL	Comparison of known answer to calculated valued
SHA hashing	OpenSSL	Comparison of known answer to calculated valued
AES encryption/decryption	Bouncy Castle JCE	Comparison of known answer to calculated valued
HMAC-SHA	Bouncy Castle JCE	Comparison of known answer to calculated valued
RSA sign/verify	Bouncy Castle JCE	Comparison of known answer to calculated valued
SHA hashing	Bouncy Castle JCE	Comparison of known answer to calculated valued
AES encryption/decryption	Linux kernel	Comparison of known answer to calculated valued
HMAC-SHA	Linux kernel	Comparison of known answer to calculated valued
SHRNG random bit generation	Linux kernel	Comparison of known answer to calculated valued

Algorithm	Implemented in	Description
SHA hashing	Linux kernel	Comparison of known answer to calculated valued

Table 6 Power-up Cryptographic Algorithm Known Answer Tests

FPT_TST_EXT.2: The TOE ensures a secure boot process in which the TOE verifies the digital signature of the bootloader software for the Application Processor (using a public key whose hash resides in the processor's internal fuses) before transferring control. The bootloader, in turn, verifies the signature of the Linux kernel (either the primary or the recovery kernel) it loads.

FPT_TUD_EXT.1: The TOE's user interface provides a method to query the current version of the TOE software/firmware (Android version, baseband version, kernel version, build number, and software version) and hardware (model and version). Additionally, the TOE provides users the ability to review the currently installed apps (including 3rd party "built-in" applications) and their version.

FPT_TUD_EXT.2: When in CC mode, the TOE verifies all updates to the TOE software using a public key chaining ultimately to the Root Public Key, a hardware protected key whose SHA-256 hash resides inside the application processor.

The application processor verifies the bootloader's authenticity and integrity (thus tying the bootloader and subsequent stages to a hardware root of trust: the SHA-256 hash of the Root Public Key, which cannot be reprogrammed after the "write-enable" fuse has been blown).

The Android OS on the TOE requires that all applications bear a valid signature before Android will install the application.

ALC_TSU_EXT: LG provides its customers with an email address (support-enterprise-mobility@lge.com) in order to report any potential issues. LG analyzes the feedback from its customers, and takes action appropriate depending upon where the bug may lie (whether in code completely maintained by LG or working with Google for Android code issues). LG creates updates and patches to revolved reported issues as quickly as possible, and makes these updates available to the wireless carriers fielding LG phones. Issues reported to Google directly are handled through Google's notification processes.

6.6 TOE access

FTA_SSL_EXT.1: The TOE transitions to its locked state either immediately after a User initiates a lock by pressing the power button (if configured) or after a (also configurable) period of inactivity, and as part of that transition, the TOE will display a lock screen to obscure the previous contents; however, the TOE's lock screen still displays email notifications, calendar appointments, user configured widgets, text message notifications, the time, date, call notifications, battery life, signal strength, and carrier network. But without authenticating first, a user cannot perform any related actions based upon these notifications (they cannot respond to emails, calendar appointments, or text messages) other than the actions assigned in FIA_UAU_EXT.2.1 (see section 5.1.3.7).

Note that during power up, the TOE presents the user with an initial power-up Data-At-Rest lock screen, where the user can only make an emergency call or enter the user password in order to allow the TOE to decrypt the Data-At-Rest key so as to be able to access the data partition. After successfully authenticating at the power-up login screen, the TOE presents the user with the lock screen.

FTA_TAB.1: The TOE can be configured to display a user-specified message (maximum of 23 characters) on the Lock screen, and additionally an administrator can also set a Lock screen message using an MDM.

FTA_WSE_EXT.1: The TOE allows an administrator to specify (through the use of an MDM) a list of wireless networks (SSIDs) to which the user may direct the TOE to connect to. When not enrolled with an MDM, the TOE allows the user to control to which wireless networks the TOE should connect, but does not provide an explicit list of such networks, rather the use may scan for available wireless network (or directly enter a specific wireless network), and then connect. Once a user has connected to a wireless network, the TOE will automatically reconnect to that network when in range and the user has enabled the TOE's WiFi radio.

6.7 Trusted path/channels

FTP_ITC_EXT.1: The TOE provides secured (encrypted and mutually authenticated) communication channels between itself and other trusted IT products through the use of 802.11-2012, 802.1X, and EAP-TLS and TLS. The TOE permits itself and applications to initiate communicate via the trusted channel, and the TOE initiates communicate via the trusted channel for connection to a wireless access point. The TOE provides access to TLS via published APIs which are accessible to any application that needs an encrypted end-to-end trusted channel.

7. TSF Inventory

Below is a list of user-mode TSF binaries and libraries that are built with the -fstack-protector option set. For each binary/library, the name, path and security function is provided.

Name	Path	Securuty Function
dalvikvm	/system/bin	Virtual Machine
keystore	/system/bin	Key Store
qrngd	/system/bin	RNG
qseecomd	/system/bin	Trustzone Daemon
time_daemon	/system/bin	Time
vold	/system/bin	DAR
wpa_supplicant	/system/bin	DAR
libcrypto.so	/system/lib	Crypto
libfotajni.so	/system/lib	FOTA JNI
libjavacrypto.so	/system/lib	Crypto JNI
libkeystore_binder.so	/system/lib	KeyStore
libkeyutils.so	/system/lib	DAR
libecryptfs.so	/system/lib	DAR
libsoftkeymaster.so	/system/lib	Key Store
libssl.so	/system/lib	SSL/TLS