LG Electronics Inc. G6 Smartphone (MDFPP30/WLANCEP10) Security Target

Version 0.3
2017/02/28

Prepared for:

LG Electronics Inc.

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

Prepared By:

www.gossamersec.com

1. SECURITY TARGET INTRODUCTION 4
1.1 SECURITY TARGET REFERENCE 5
1.2 TOE REFERENCE 5
1.3 TOE OVERVIEW 5
1.4 TOE DESCRIPTION 5
1.4.1 TOE Architecture 6
1.4.2 TOE Documentation 8
2. CONFORMANCE CLAIMS 9
2.1 CONFORMANCE RATIONALE 9
3. SECURITY OBJECTIVES 10
3.1 SECURITY OBJECTIVES FOR THE OPERATIONAL ENVIRONMENT 10
4. EXTENDED COMPONENTS DEFINITION 11
5. SECURITY REQUIREMENTS 13
5.1 TOE SECURITY FUNCTIONAL REQUIREMENTS 13
5.1.1 Security audit (FAU) 15
5.1.2 Cryptographic support (FCS) 16
5.1.3 User data protection (FDP) 21
5.1.4 Identification and authentication (FIA) 23
5.1.5 Security management (FMT) 26
5.1.6 Protection of the TSF (FPT) 32
5.1.7 TOE access (FTA) 35
5.1.8 Trusted path/channels (FTP) 35
5.2 TOE SECURITY ASSURANCE REQUIREMENTS 36
5.2.1 Development (ADV) 36
5.2.2 Guidance documents (AGD) 37
5.2.3 Life-cycle support (ALC) 38
5.2.4 Tests (ATE) 38
5.2.5 Vulnerability assessment (AVA) 39
6. TOE SUMMARY SPECIFICATION 40
6.1 SECURITY AUDIT 40
6.2 CRYPTOGRAPHIC SUPPORT 40
6.3 USER DATA PROTECTION 44
6.4 IDENTIFICATION AND AUTHENTICATION 47
6.5 SECURITY MANAGEMENT 48
6.6 PROTECTION OF THE TSF 49
6.7 TOE ACCESS 52
6.8 TRUSTED PATH/CHANNELS 52
7. TSF INVENTORY 53

LIST OF TABLES

Table 1 TOE Security Functional Components 15
Table 2 Security Management Functions 26
Table 3 WLAN Security Management Functions 31
Table 4 EAL 1 augmented with ALC_TSU_EXT.1 Assurance Components 36
Table 5 Asymmetric Key Generation 40
Table 6 Key Enumeration .... **Error! Bookmark not defined.**
Table 7 BoringSSL Cryptographic Algorithms 42
Table 8 Kernel Loadable Cryptographic Module Algorithms 42
Table 9 Application Processor Cryptographic Algorithms 43

Table 10 Key IV Source .... **Error! Bookmark not defined.**
Table 11 Power-up Cryptographic Algorithm Known Answer Tests..51

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 the G6 Smartphone 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)

Acronyms and Terminology

AA	Assurance Activity
CC	Common Criteria
CCEVS	Common Criteria Evaluation and Validation Scheme
EAR	Entropy Analysis Report (Entropy Documentation and Assessment)
GUI	Graphical User Interface
NFC	Near Field Communication
PCL	Product Compliant List
PP	Protection Profile
SAR	Security Assurance Requirement
SE	Secure Element
SFR	Security Functional Requirement
SOF	Strength of Function
ST	Security Target
TEE	Trusted Execution Environment (TrustZone)
TOE	Target of Evaluation
U.S.	United States
VR	Validation Report

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 MDFPP 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. G6 Smartphone (MDFPP30/WLANCEP10) Security Target

ST Version – Version 0.3

ST Date – 2017/02/28

1.2 TOE Reference

TOE Identification – LG Electronics Inc. G6 Smartphone

TOE Developer – LG Electronics Inc.

Evaluation Sponsor – LG Electronics Inc.

1.3 TOE Overview

The Target of Evaluation (TOE) is the G6 Smartphone featuring the following specifications:

Feature	G6
Display	5.7 Inch, QHD+ 2,880 x 1,440 resolution 18:9 aspect ratio display
Camera	Front 5MP with F2.2 Aperture (Wide Angle) / Rear 13MP with F1.8 Aperture OIS (Standard Angle) and 13MP with F2.4 Aperture (Wide Angle)
Communications	X12 LTE (up to 600 Mbps LTE Category 12/13 with 3x Carrier Aggregation) / Wi-Fi (802.11 a, b, g, n, ac) / USB Type-C / Bluetooth 4.2 BLE / NFC
Processor/ chipset	ARMv8 Qualcomm MSM8996 Pro Snapdragon 821 Application Processor
RAM	4 GB RAM
Storage	32GB Flash; microSD (up to 2TB)
Battery	Li-polymer 3,300 mAh

The TOE allows basic telephony features (make and receive phone calls, send and receive SMS/MMS messages) as well as advanced network connectivity (allowing connections to both 802.11 Wi-Fi and 2G/3G/4G LTE mobile data networks). The TOE 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 TOE 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 7.0 with a Linux 3.18 kernel and improved by LG (for example, adding NIST approved cryptographic algorithms, strengthening X5.09 certificate checking, bolstering keystore protection, and applying security patches) to meet the

MDFPP requirements, the TOE provides wireless connectivity and provides an execution environment for mobile applications.

The following models and versions are included in the evaluation:

Product	Carrier	Security SW Version	OS version	Build number	WFA Cert#
LG G6 H871	AT&T	MDF v3.0 Release 1	Android 7.0	NRD90U	WFA69452
LG G6 VS988	Verizon	MDF v3.0 Release 1	Android 7.0	NRD90U	WFA69251
LG G6 LS993	Sprint	MDF v3.0 Release 1	Android 7.0	NRD90U	WFA69457
LG G6 H872	T-Mobile	MDF v3.0 Release 1	Android 7.0	NRD90U	WFA69454
LG G6 US997	Open, U.S. Cellular, LRA	MDF v3.0 Release 1	Android 7.0	NRD90U	WFA69455

During the evaluation, Gossamer utilized a Verizon model of the phone for all testing.
Furthermore, one must configure the TOE into its Common Criteria Mode in order to utilize the TOE in its evaluated configuration. As described in the Admin Guide, one must select the following options to configure the TOE into Common Criteria Mode:

1. Enable the password on the lock-screen (done by an MDM Agent or by the user through the UI)
2. Enable SD card encryption (via MDM Agent or UI)
3. Set CC mode (using an MDM API either through an MDM Agent or through the standalone app referenced in the Admin Guide)

Doing this ensures that the phone complies with the MDFPP requirements (for example restricting TLS/HTTPS ciphersuites, disallowing 'Download Mode', disabling 'Smart Lock', disallowing VPN split-tunneling).

Please refer to the Admin Guide for more details on how to accomplish the necessary configuration steps.

1.4.1 TOE Architecture

The TOE provides a rich API 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 can interact with a Mobile Device Management system (not part of this evaluation) to allow enterprise control of the configuration and operation of the device so as to ensure adherence to enterprise-wide policies (for example, enabling CC mode, or restricting use of the device's camera, etc.).

The TOE includes several different levels of execution including (from lowest to highest) hardware, a Trusted Execution Environment, Android's Linux kernel, Android's user space, Android's Android Runtime (ART) environment for mobile applications, and the mobile applications themselves.

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 the TOE:

- 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 components (including its BoringSSL library, its Kernel Loadable Cryptographic module, and its Application Processor) with CAVP 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. The TOE supports Android for Work profiles to provide additional separation between application and application data belonging to the Android for Work profile. Please see the Admin Guide for additional details regarding how to set up and use Android for Work profiles.

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 both use X.509v3 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. LG Android 7 devices (G6) Guidance Documentation, Version 0.1, 2017/02/08 **[Admin Guide]**

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 4, September 2012.
    - Part 2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1 Revision 4, September 2012.
    - Part 3 Extended
- Package Claims:
    - Protection Profile for Mobile Device Fundamentals, Version 3.0, 10 June 2016 and General Purpose Operating Systems Protection Profile/Mobile Device Fundamentals Protection Profile Extended Package (EP) Wireless Local Area Network (WLAN) Clients, Version 1.0, 08 February 2016 (MDFPP30/WLANCEP10)
- Technical Decisions:
    - Applicable NIAP Technical decisions: TD0118, TD0120, TD0145, TD0147, TD0148, TD0158

2.1 Conformance Rationale

The ST conforms to the MDFPP30/WLANCEP10. 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 MDFPP30/WLANCEP10 and this section reproduces only the corresponding Security Objectives for operational environment for reader convenience. The MDFPP30/WLANCEP10 offers additional information about the identified security objectives, but that has not been reproduced here and the MDFPP30/WLANCEP10 should be consulted if there is interest in that material.

In general, the MDFPP30/WLANCEP10 has defined Security Objectives appropriate for mobile devices and as such are applicable to the G6 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.NO_TOE_BYPASS Information cannot flow between external and internal networks located in different enclaves without passing through the TOE.

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.

OE.TRUSTED_ADMIN TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.

4. Extended Components Definition

All of the extended requirements in this ST have been drawn from the MDFPP30/WLANCEP10. The MDFPP30/WLANCEP10 defines the following extended requirements and since they are not redefined in this ST the MDFPP30/WLANCEP10 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: Extended: 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: TLS Protocol
- FCS_TLSC_EXT.1(1): Extensible Authentication Protocol-Transport Layer Security - WLAN
- FDP_ACF_EXT.1: Extended: Security access control
- FDP_BCK_EXT.1: Extended: Application Backup
- FDP_DAR_EXT.1: Extended: Protected Data Encryption
- FDP_DAR_EXT.2: Extended: Sensitive Data Encryption
- FDP_IFC_EXT.1: Extended: Subset information flow control
- FDP_STG_EXT.1: Extended: User Data Storage
- FDP_UPC_EXT.1: Extended: Inter-TSF user data transfer protection
- FIA_AFL_EXT.1: Authentication failure handling
- FIA_BLT_EXT.1: Extended: Bluetooth User Authorization
- FIA_BLT_EXT.2: Extended: Bluetooth Mutual Authentication
- FIA_BLT_EXT.3: Extended: Rejection of Duplicate Bluetooth Connections
- FIA_BLT_EXT.4: Extended: Secure Simple Pairing
- FIA_PAE_EXT.1: Port Access Entity 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: Extended: Timing of Authentication
- FIA_X509_EXT.1: Extended: Validation of certificates
- FIA_X509_EXT.2: Extended: X509 certificate authentication
- FIA_X509_EXT.2(1): X.509 Certificate Authentication (EAP-TLS) - WLAN
- FIA_X509_EXT.3: Extended: Request Validation of certificates
- FIA_X509_EXT.4: Certificate Storage and Management
- FMT_MOF_EXT.1: Extended: Management of security functions behavior
- FMT_SMF_EXT.1: Extended: Specification of Management Functions
- FMT_SMF_EXT.1(1): Specification of Management Functions (Wireless LAN)
- FMT_SMF_EXT.2: Extended: Specification of Remediation Actions
- FMT_SMF_EXT.3: Extended: Current Administrator
- 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 (Overflow Protection)
- FPT_AEX_EXT.4: Extended: Domain Isolation
- FPT_BBD_EXT.1: Application Processor Mediation
- FPT_JTA_EXT.1: Extended: JTAG Disablement
- 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: Self-Test Notification
- FPT_TST_EXT.1: Extended: TSF Cryptographic Functionality Testing
- FPT_TST_EXT.1(1): TSF Cryptographic Functionality Testing (Wireless LAN)
- 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_WSE_EXT.1: Wireless Network Access
- FTP_ITC_EXT.1: Extended: Trusted channel Communication
- FTP_ITC_EXT.1(1): Trusted Channel Communication (Wireless LAN)

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 MDFPP30/WLANCEP10. The refinements and operations already performed in the MDFPP30/WLANCEP10 are not identified (e.g., highlighted) here, rather the requirements have been copied from the MDFPP30/WLANCEP10 and any residual operations have been completed herein. Of particular note, the MDFPP30/WLANCEP10 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 MDFPP30/WLANCEP10 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 MDFPP30/WLANCEP10 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 MDFPP30/WLANCEP10 should be consulted for the assurance activity definitions.

5.1 TOE Security Functional Requirements

The following table identifies the SFRs that are satisfied by G6 Smartphone TOE.

Requirement Class	Requirement Component
FAU: Security Audit	FAU_GEN.1: Audit Data Generation
	FAU_SAR.1: Audit Review
	FAU_STG.1: Audit Storage Protection
	FAU_STG.4: Prevention of Audit Data Loss
FCS: Cryptographic Support	FCS_CKM.1: Cryptographic key generation
	FCS_CKM.1(3): Cryptographic Key Generation (Symmetric Keys for WPA2 Connections) - WLAN
	FCS_CKM.2(1): Cryptographic key establishment
	FCS_CKM.2(2): Cryptographic key establishment (While device is locked)
	FCS_CKM.2(3): Cryptographic Key Distribution (GTK) - WLAN
	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): Cryptographic operation
	FCS_COP.1(4): Cryptographic operation
	FCS_COP.1(5): Cryptographic operation
	FCS_HTTPS_EXT.1: Extended: 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_TLSC_EXT.1: Extended: TLS Protocol
	FCS_TLSC_EXT.1(1): Extensible Authentication Protocol-Transport Layer Security - WLAN
FDP: User data Protection	FDP_ACF_EXT.1(*): Extended: Security access control
	FDP_BCK_EXT.1: Extended: Application Backup
	FDP_DAR_EXT.1: Extended: Protected Data Encryption
	FDP_DAR_EXT.2: Extended: Sensitive Data Encryption
	FDP_IFC_EXT.1: Extended: Subset information flow control
	FDP_STG_EXT.1: Extended: User Data Storage
	FDP_UPC_EXT.1: Extended: Inter-TSF user data transfer protection
FIA: Identification and authentication	FIA_AFL_EXT.1: Authentication failure handling
	FIA_BLT_EXT.1: Extended: Bluetooth User Authorization
	FIA_BLT_EXT.2: Extended: Bluetooth Mutual Authentication
	FIA_BLT_EXT.3: Extended: Rejection of Duplicate Bluetooth Connections
	FIA_BLT_EXT.4: Extended: Secure Simple Pairing
	FIA_PAE_EXT.1: Port Access Entity Authentication
	FIA_PMG_EXT.1: Extended: Password Management
	FIA_TRT_EXT.1: Extended: Authentication Throttling
	FIA_UAU.5: Multiple Authentication Mechanisms
	FIA_UAU.6(1): Re-Authentication
	FIA_UAU.6(2): Re-Authentication
	FIA_UAU.7: Protected authentication feedback
	FIA_UAU_EXT.1: Extended: Authentication for Cryptographic Operation
	FIA_UAU_EXT.2: Extended: Timing of Authentication
	FIA_X509_EXT.1: Extended: Validation of certificates
	FIA_X509_EXT.2: Extended: X509 certificate authentication
	FIA_X509_EXT.2(1): X.509 Certificate Authentication (EAP-TLS) - WLAN
	FIA_X509_EXT.3: Extended: Request Validation of certificates
	FIA_X509_EXT.4: Certificate Storage and Management
FMT: Security management	FMT_MOF_EXT.1: Extended: Management of security functions behavior
	FMT_SMF_EXT.1: Extended: Specification of Management Functions
	FMT_SMF_EXT.1(1): Specification of Management Functions (Wireless LAN)
	FMT_SMF_EXT.2: Extended: Specification of Remediation Actions
	FMT_SMF_EXT.3: Extended: Current Administrator
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 (Overflow Protection)
	FPT_AEX_EXT.4: Extended: Domain Isolation
	FPT_BBD_EXT.1: Application Processor Mediation
	FPT_JTA_EXT.1: Extended: JTAG Disablement
	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: Self-Test Notification
	FPT_STM.1: Reliable time stamps
	FPT_TST_EXT.1: Extended: TSF Cryptographic Functionality Testing
	FPT_TST_EXT.1(1): TSF Cryptographic Functionality Testing (Wireless LAN)
	FPT_TST_EXT.2: Extended: TSF Integrity Checking
	FPT_TUD_EXT.1: Extended: Trusted Update: TSF version query
	FPT_TUD_EXT.2: Extended: TSF 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: Wireless Network Access
FTP: Trusted path/channels	FTP_ITC_EXT.1: Extended: Trusted channel Communication
	FTP_ITC_EXT.1(1): Trusted Channel Communication (Wireless LAN)

Table 1 TOE Security Functional Components

5.1.1 Security audit (FAU)

5.1.1.1 Audit Data Generation (FAU_GEN.1)

FAU_GEN.1.1

The TSF shall be able to generate an audit record of the following auditable events:
1. Start-up and shutdown of the audit functions;
2. All auditable events for the not selected level of audit;
3. All administrative actions;
4. Start-up and shutdown of the Rich OS;
5. Insertion or removal of removable media;
6. Specifically defined auditable events in Table 1 (of the MDFPP30);
7. [***Audit records reaching [95%] percentage of audit capacity***].
8. [***Specifically defined auditable events in Table2 of MDFPP30/WLANCEP10***].

FAU_GEN.1.2

The TSF shall record within each audit record at least the following information:
1. Date and time of the event;
2. type of event;
3. subject identity;
4. the outcome (success or failure) of the event; and
5. additional information in Table 1 (of the MDFPP30).
6. [***additional information in Table2 of MDFPP30/WLANCEP10***].

5.1.1.2 Audit Review (FAU_SAR.1)

FAU_SAR.1.1

The TSF shall provide the administrator with the capability to read all audited events and record contents from the audit records.

FAU_SAR.1.2

The TSF shall provide the audit records in a manner suitable for the user to interpret the information.

5.1.1.3 Audit Storage Protection (FAU_STG.1)

FAU_STG.1.1

The TSF shall protect the stored audit records in the audit trail from unauthorized deletion.

FAU_STG.1.2

The TSF shall be able to prevent unauthorized modifications to the stored audit records in the audit trail.

5.1.1.4 Prevention of Audit Data Loss (FAU_STG.4)

FAU_STG.4.1

The TSF shall overwrite the oldest stored audit records if the audit trail is full.

5.1.2 Cryptographic support (FCS)

5.1.2.1 Cryptographic key generation (FCS_CKM.1)

FCS_CKM.1.1

The TSF shall generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm [
- RSA schemes using cryptographic key sizes of 2048-bit or greater that meet FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Appendix B.3,
- ECC schemes using [NIST curves' P-256, P-384 and [P-521] that meet the following: FIPS PUB 186-4, Digital Signature Standard (DSS)', Appendix B.4, Curve25519 schemes that meet the following: RFC 7748],
- FFC schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Appendix B.1].

5.1.2.2 Cryptographic Key Generation (Symmetric Keys for WPA2 Connections) - WLAN (FCS_CKM.1(3))

FCS_CKM.1(3).1

Refinement: The TSF shall generate symmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm PRF-384 and [***no other***] and specified cryptographic key sizes 128 bits and [***no other key sizes***] using a Random Bit Generator as specified in FCS_RBG_EXT.1 that meet the following: IEEE 802.11-2012 and [***no other standards***].

5.1.2.3 Cryptographic key establishment (FCS_CKM.2(1))

FCS_CKM.2(1).1

The TSF shall perform cryptographic key establishment in accordance with a specified cryptographic key establishment method:
- [RSA-based key establishment schemes] that meets the following: [NIST Special Publication 800-56B, 'Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography'];
and [- ***[Elliptic curve-based key establishment schemes] that meets the following: [NIST Special Publication 800-56A, 'Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography'];***
- [Finite field-based key establishment schemes] that meets the following: [NIST Special Publication 800-56A, 'Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography'];] for the purposes of encrypting sensitive data received while the device is locked..

5.1.2.4 Cryptographic key establishment (While device is locked) (FCS_CKM.2(2))

FCS_CKM.2(2).1

The TSF shall perform cryptographic key establishment in accordance with a specified cryptographic key establishment method:
[Elliptic curve-based key establishment schemes] that meets the following: [***NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"***];
for the purposes of encrypting sensitive data received while the device is locked.

5.1.2.5 Cryptographic Key Distribution (GTK) - WLAN (FCS_CKM.2(3))

FCS_CKM.2(3).1

Refinement: The TSF shall decrypt Group Temporal Key in accordance with a specified cryptographic key distribution method AES Key Wrap in an EAPOL-Key frame that meets the following: RFC 3394 for AES Key Wrap, 802.11-2012 for the packet format and timing considerations and does not expose the cryptographic keys.

5.1.2.6 Extended: Cryptographic Key Support (FCS_CKM_EXT.1)

FCS_CKM_EXT.1.1
The TSF shall support [***immutable hardware***] REK(s) with [***a symmetric***] key of strength [***256 bits***].

FCS_CKM_EXT.1.2
Each REK shall be hardware-isolated from Rich OS on the TSF in runtime.

FCS_CKM_EXT.1.3
A REK shall be generated by a RBG in accordance with FCS_RBG_EXT.1.

5.1.2.7 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.

5.1.2.8 Extended: Cryptographic Key Generation (FCS_CKM_EXT.3)

FCS_CKM_EXT.3.1
The TSF shall use [***asymmetric KEKs of [security strength greater than or equal to 256 bits] security strength, symmetric KEKs of [256-bit] security strength corresponding to at least the security strength of the keys encrypted by the KEK***]. (TD0145 Applied)

FCS_CKM_EXT.3.2
The TSF shall generate all KEKs using one or more of the following methods:
a) derive the KEK from a Password Authentication Factor using PBKDF and
[***b) generate the KEK using an RBG that meets this profile (as specified in FCS_RBG_EXT.1)***
c) generate the KEK using a key generation scheme that meets this profile (as specified in FCS_CKM.1),
d) Combine the KEK from other KEKs in a way that preserves the effective entropy of each factor by [encrypting one key with another]].

5.1.2.9 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 methods:
- by clearing the KEK encrypting the target key,
- in accordance with the following rules:

o For volatile memory, the destruction shall be executed by a single direct overwrite [***consisting of zeros***].
o For non-volatile EEPROM, the destruction shall be executed by a single direct overwrite consisting of a pseudo random pattern using the TSF's RBG (as specified in FCS_RBG_EXT.1), followed a read-verify.
o For non-volatile flash memory, that is not wear-leveled, the destruction shall be executed [***by a single direct overwrite consisting of zeros followed by a read-verify***].
o For non-volatile flash memory that is wear-leveled, the destruction shall be executed [***by a single direct overwrite consisting of zeros***].
o For non-volatile memory other than EEPROM and flash, the destruction shall be executed by overwriting with a random pattern that is changed before each write.

FCS_CKM_EXT.4.2
The TSF shall destroy all plaintext keying material and critical security parameters when no longer needed.

5.1.2.10 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.

5.1.2.11 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.

5.1.2.12 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 as defined in FIPS PUB 197, and NIST SP 800-38A) mode,
- AES-CCMP (as defined in FIPS PUB 197, NIST SP 800-38C and IEEE 802.11-2012), and
- [***AES Key Wrap (KW) (as defined in NIST SP 800-38F)***
AES-GCM (as defined in NIST SP 800-38D)
AES-CCM (as defined in NIST SP 800-38C) mode
AES-XTS (as defined in NIST SP 800-38E) mode]
and cryptographic key sizes 128-bit key sizes and [***256-bit key sizes***].

5.1.2.13 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].

5.1.2.14 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
- [RSA schemes] using cryptographic key sizes [of 2048-bit or greater] that meet the following: FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Section 4
and [***- [ECDSA schemes] using ['NIST curves' P-256, P-384 and [P-521]] that meet the following: FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Section 5;***].

5.1.2.15 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'.

5.1.2.16 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***]], with [***8192***] iterations, and output cryptographic key sizes [***256***] that meet the following: NIST SP 800-132.

5.1.2.17 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_TLSC_EXT.1).

FCS_HTTPS_EXT.1.3

The TSF shall notify the application and [***not establish the connection***] if the peer certificate is deemed invalid.

5.1.2.18 Extended: Initialization Vector Generation (FCS_IV_EXT.1)

FCS_IV_EXT.1.1

The TSF shall generate IVs in accordance with MDFPP30/WLANCEP10 Table 15: References and IV Requirements for NIST-approved Cipher Modes.

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

FCS_RBG_EXT.1.1

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

FCS_RBG_EXT.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.1.3

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

5.1.2.20 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:
- All mandatory and [***selected algorithms***] in FCS_CKM.2(2),
- The following algorithms in FCS_COP.1(1): AES-CBC, [***no other modes***]
- All mandatory and selected algorithms in FCS_COP.1(3)
- All mandatory and selected algorithms in FCS_COP.1(2)
- All mandatory and selected algorithms in FCS_COP.1(4)

[***- No other cryptographic operations***].

5.1.2.21 Extended: Cryptographic Key Storage (FCS_STG_EXT.1)

FCS_STG_EXT.1.1

The TSF shall provide [***software-based***] secure key storage for asymmetric private keys and [***symmetric keys, persistent secrets***].

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***].

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***].

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

FCS_STG_EXT.2.1
The TSF shall encrypt all DEKs, KEKs, [***Wi-Fi PSK, BT long term keys***] and [***all software-based key storage***] by KEKs that are [
1) Protected by the REK with [b. encryption by a KEK that is derived from a REK].
2) Protected by the REK and the password with [b. encryption by a KEK chaining to a REK and the password-derived or biometric-unlocked KEK]].

FCS_STG_EXT.2.2
DEKs and KEKs and [***long-term trusted channel key material, all software-based key storage***] shall be encrypted using one of the following methods: [***using AES in the [CCM,CBC mode]***].

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

FCS_STG_EXT.3.1
The TSF shall protect the integrity of any encrypted DEKs and KEKs and [***no other keys***] 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 [***MAC***] of the stored key prior to use of the key.

5.1.2.24 Extended: TLS Protocol (FCS_TLSC_EXT.1)

FCS_TLSC_EXT.1.1
The TSF shall implement TLS 1.2 (RFC 5246) supporting the following ciphersuites:
- Mandatory Ciphersuites:
TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246
- Optional Ciphersuites:
[***TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,***
TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246,
TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,
TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,
TLS_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246,
TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288 ,
TLS_DHE_RSA_WITH_AES_128_CBC_ SHA256 as defined in RFC 5246,
TLS_DHE_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289].

FCS_TLSC_EXT.1.2
The TSF shall verify that the presented identifier matches the reference identifier according to RFC 6125.

FCS_TLSC_EXT.1.3
The TSF shall not establish a trusted channel if the peer certificate is invalid.

FCS_TLSC_EXT.1.4
The TSF shall support mutual authentication using X.509v3 certificates.

FCS_TLSC_EXT.1.5
The TSF shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [***secp256r1, secp384r1, secp521r1***] and no other curves.

5.1.2.25 Extensible Authentication Protocol-Transport Layer Security - WLAN (FCS_TLSC_EXT.1(1))

FCS_TLSC_EXT.1(1).1
The TSF shall implement TLS 1.0 and [***TLS 1.1 (RFC 4346), TLS 1.2 (RFC 5246)***] in support of the EAP-TLS protocol as specified in RFC 5216 supporting the following ciphersuites:
Mandatory Ciphersuites in accordance with RFC 5246:
- TLS_RSA_WITH_AES_128_CBC_SHA

Optional Ciphersuites:
[***TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,***
TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246,
TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,].

FCS_TLSC_EXT.1(1).2
The TSF shall generate random values used in the EAP-TLS exchange using the RBG specified in FCS_RBG_EXT.1.

FCS_TLSC_EXT.1(1).3
The TSF shall use X509 v3 certificates as specified in FIA_X509_EXT.1.

FCS_TLSC_EXT.1(1).4
The TSF shall verify that the server certificate presented includes the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.

FCS_TLSC_EXT.1(1).5
The TSF shall allow an authorized administrator to configure the list of CAs that are allowed to sign authentication server certificates that are accepted by the TOE.

FCS_TLSC_EXT.1(1).6
The TSF shall allow an authorized administrator to configure the list of algorithm suites that may be proposed and accepted during the EAP-TLS exchanges.

5.1.3 User data protection (FDP)

5.1.3.1 Extended: Security access control (FDP_ACF_EXT.1(1))

FDP_ACF_EXT.1(1).1
The TSF shall provide a mechanism to restrict the system services that are accessible to an application.

FDP_ACF_EXT.1(1).2
The TSF shall provide an access control policy that prevents [***application***] from accessing [***all***] data stored by other [***application***]. Exceptions may only be explicitly authorized for such sharing by [***a common application developer***].

5.1.3.2 Extended: Security access control (FDP_ACF_EXT.1(2))

FDP_ACF_EXT.1(2).1
The TSF shall provide a mechanism to restrict the system services that are accessible to an application.

FDP_ACF_EXT.1(2).2
The TSF shall provide an access control policy that prevents [***groups of applications***] from accessing [***all***] data stored by other [***groups of applications***]. Exceptions may only be explicitly authorized for such sharing by [***the administrator***].

FDP_ACF_EXT.1(2).4
The TSF shall provide a separate [***address book, calendar, keystore***] for each application group and only allow applications within that process group to access the resource. Exceptions may only be explicitly authorized for such sharing by [***no one***].

5.1.3.3 Extended: Application Backup (FDP_BCK_EXT.1)

FDP_BCK_EXT.1.1
The TSF shall provide a mechanism for applications to mark [***all application data***] to be excluded from device backups.

5.1.3.4 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 [***XTS, CBC***] mode with key size [***128, 256***] bits.

5.1.3.5 Extended: Sensitive Data Encryption (FDP_DAR_EXT.2)

FDP_DAR_EXT.2.1
The TSF shall provide a mechanism for applications to mark data and keys as sensitive.

FDP_DAR_EXT.2.2
The TSF shall use an asymmetric key scheme to encrypt and store sensitive data received while the product is locked.

FDP_DAR_EXT.2.3
The TSF shall encrypt any stored symmetric key and any stored private key of the asymmetric key(s) used for the protection of sensitive data according to FCS_STG_EXT.2.1 selection 2.

FDP_DAR_EXT.2.4
The TSF shall decrypt the sensitive data that was received while in the locked state upon transitioning to the unlocked state using the asymmetric key scheme and shall re-encrypt that sensitive data using the symmetric key scheme.

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

FDP_IFC_EXT.1.1
The TSF shall [- ***provide an interface which allows a VPN client to protect all IP traffic using IPsec***] with the exception of IP traffic required to establish the VPN connection.

5.1.3.7 Extended: User Data Storage (FDP_STG_EXT.1)

FDP_STG_EXT.1.1
The TSF shall provide protected storage for the Trust Anchor Database.

5.1.3.8 Extended: Inter-TSF user data transfer protection (FDP_UPC_EXT.1)

FDP_UPC_EXT.1.1
The TSF shall provide a means for non-TSF applications executing on the TOE to use TLS, HTTPS, Bluetooth BR/EDR, and [***Bluetooth LE***] to provide a protected communication channel between the non-TSF application and another IT product that is logically distinct from other communication channels, provides assured identification of its end points, protects channel data from disclosure, and detects modification of the channel data.

FDP_UPC_EXT.1.2
The TSF shall permit the non-TSF applications to initiate communication via the trusted channel.

5.1.4 Identification and authentication (FIA)

5.1.4.1 Authentication failure handling (FIA_AFL_EXT.1)

FIA_AFL_EXT.1.1
The TSF shall consider password and [***no other***] as critical authentication mechanisms.

FIA_AFL_EXT.1.2
The TSF shall detect when a configurable positive integer within [***1-50***] of [***non-unique***] unsuccessful authentication attempts occur related to last successful authentication for each authentication mechanism.

FIA_AFL_EXT.1.3
The TSF shall maintain the number of unsuccessful authentication attempts that have occurred upon power off.

FIA_AFL_EXT.1.4
When the defined number of unsuccessful authentication attempts has exceeded the maximum allowed for a given authentication mechanism, all future authentication attempts will be limited to other available authentication mechanisms, unless the given mechanism is designated as a critical authentication mechanism.

FIA_AFL_EXT.1.5
When the defined number of unsuccessful authentication attempts for the last available authentication mechanism or single critical authentication mechanism has been surpassed, the TSF shall perform a wipe of all protected data.

FIA_AFL_EXT.1.6
The TSF shall increment the number of unsuccessful authentication attempts prior to notifying the user that the authentication was unsuccessful.

5.1.4.2 Extended: Bluetooth User Authorization (FIA_BLT_EXT.1)

FIA_BLT_EXT.1.1
The TSF shall require explicit user authorization before pairing with a remote Bluetooth device.

FIA_BLT_EXT.1.2
The TSF shall require explicit user authorization before granting trusted remote devices access to services associated with the following Bluetooth profiles: [***OPP, MAP***], and shall require explicit user authorization before granting untrusted remote devices access to services associated with the following Bluetooth profiles: [***all available Bluetooth profiles***].

5.1.4.3 Extended: Bluetooth Authentication (FIA_BLT_EXT.2)

FIA_BLT_EXT.2.1
The TSF shall require Bluetooth mutual authentication between devices prior to any data transfer over the Bluetooth link.

5.1.4.4 Extended: Rejection of Duplicate Bluetooth Connections (FIA_BLT_EXT.3)

FIA_BLT_EXT.3.1
The TSF shall discard connection attempts from a Bluetooth device address (BD_ADDR) to which a current connection already exists.

5.1.4.5 Extended: Secure Simple Pairing (FIA_BLT_EXT.4)

FIA_BLT_EXT.4.1
The TOE shall support Bluetooth Secure Simple Pairing, both in the host and the controller. Furthermore, Secure Simple Pairing shall be used during the pairing process if the remote device also supports it.

5.1.4.6 Port Access Entity 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.

5.1.4.7 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.

5.1.4.8 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***] for all authentication mechanisms selected in FIA_UAU.5.1. The minimum delay shall be such that no more than 10 attempts can be attempted per 500 milliseconds.

5.1.4.9 Multiple Authentication Mechanisms (FIA_UAU.5)

FIA_UAU.5.1
The TSF shall provide password and [***no other mechanism***] to support user authentication.

FIA_UAU.5.2
The TSF shall authenticate any user's claimed identity according to the [***validation of the user's password***]. (Per TD0147)

5.1.4.10 Re-Authentication (FIA_UAU.6(1))

FIA_UAU.6(1).1
The TSF shall re-authenticate the user via the Password Authentication Factor under the conditions attempted change to any supported authentication mechanisms.

5.1.4.11 Re-Authentication (FIA_UAU.6(2))

FIA_UAU.6(2).1
The TSF shall re-authenticate the user via an authentication factor defined in FIA_UAU.5.1 under the conditions TSF-initiated lock, user-initiated lock, [**change password, access Select screen lock menu**].

5.1.4.12 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.

5.1.4.13 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 encrypted DEKs, KEKs and [***all software-based key storage***] at startup.

5.1.4.14 Timing of Authentication (FIA_UAU_EXT.2)

FIA_UAU_EXT.2.1
The TSF shall allow [***[choose the keyboard input method, make emergency phone calls, take screen shots (stored internally), receive calls, turn TOE off, restart TOE, enable/disable airplane mode, 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.

5.1.4.15 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 [***the Online Certificate Status Protocol (OCSP) as specified in RFC 2560***].
- 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) in the extendedKeyUsage field.
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.
o (Conditional) Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) 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.

5.1.4.16 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 [***TLS, HTTPS***], and [***no additional uses***].

FIA_X509_EXT.2.2
When the TSF cannot establish a connection to determine the revocation status of a certificate, the TSF shall [***not accept the certificate***].

5.1.4.17 X.509 Certificate Authentication (EAP-TLS) - WLAN (FIA_X509_EXT.2(1))

FIA_X509_EXT.2(1).1
The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for EAP-TLS exchanges.

FIA_X509_EXT.2(1).2
When the TSF cannot establish a connection to determine the validity of a certificate, the TSF shall [***accept the certificate***].

5.1.4.18 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.

5.1.5 Security management (FMT)

5.1.5.1 Management of security functions behavior (FMT_MOF_EXT.1)

FMT_MOF_EXT.1.1
The TSF shall restrict the ability to perform the functions in column 3 of **Table 2 Security Management Functions** to the user.

FMT_MOF_EXT.1.2
The TSF shall restrict the ability to perform the functions in column 5 of **Table 2 Security Management Functions** to the administrator when the device is enrolled and according to the administrator-configured policy.

5.1.5.2 Specification of Management Functions (FMT_SMF_EXT.1)

FMT_SMF_EXT.1.1
The TSF shall be capable of performing the functions in column 2 of **Table 2 Security Management Functions**.

FMT_SMF_EXT.1.2
The TSF shall be capable of allowing the administrator to perform **the functions in column 4 Table 2 Security Management Functions.**

Table 2 Security Management Functions

Management Function **Status Markers:** M – Mandatory I – Implemented	FMT_SMF_EXT.1.1	FMT_MOF_EXT.1.1	FMT_SMF_EXT.1.2	FMT_MOF_EXT.1.2
1. configure password policy: a. minimum password length b. minimum password complexity c. maximum password lifetime The administrator can configure the required password characteristics (minimum length,	M		M	M

complexity, and lifetime) using the Android MDM APIs. Length: an integer value of characters (at least 4 and fewer than 16) Complexity: Unspecified, Something, Numeric, Alphabetic, Alphanumeric, Complex. Lifetime: an integer value of days (0 = no maximum)				
2. configure session locking policy: a. screen-lock enabled/disabled b. screen lock timeout c. number of authentication failures The administrator can configure the session locking policy using the Android MDM APIs. Screen lock timeout: an integer number of minutes before the TOE locks (0 = no lock timeout) Authentication failures: an integer number (0 = no limit and CC mode allows values of 1-50 inclusive)	M		M	M
3. enable/disable the VPN protection: a. across device [**c.** ***no other method***] Both users and administrator (using the TOE's MDM APIs) can configure a third party VPN client and then enable the VPN client to protect traffic.	M			
4. enable/disable [**Wi-Fi, GPS, cellular, NFC, Bluetooth BR/DR, Bluetooth LE**] The administrator can disable the radios using the TOE's MDM APIs. Once disabled, a user cannot enable the radio. The TOE's radios operate at frequencies of 2.4 GHz (NFC/Bluetooth), 2.4/5 GHz (Wi-Fi), and 850 MHz (4G/LTE).	M		I	I
5. enable/disable [**camera, microphone**]: a. across device [**c.** ***no other method***] An administrator may configure the TOE (through an MDM agent utilizing the TOE's MDM APIs) to turn off the camera and or microphones. If the administrator has disabled either the camera or the microphones, then the user cannot use those capture devices.	M		I	I
6. transition to the locked state Both users and administrators (using the TOE's MDM APIs) can transition the TOE into a locked state.	M		M	
7. full wipe of protected data Both users and administrators (using the TOE's MDM APIs) can force the TOE to perform a full wipe (factory reset) of data.	M		M	
8. configure application installation policy by: ***a. restricting the sources of applications,*** ***c. denying installation of applications***] The administrator using the TOE's MDM APIs can configure the TOE so that applications cannot be installed and can also block the use of the Google Market Place.	M		M	M
9. import keys/secrets into the secure key storage Both users and administrators (using the TOE's MDM APIs) can import secret keys into the secure key storage.	M		I	
10. destroy imported keys/secrets and [***no other keys/secrets***] in the secure key storage Both users and administrators (using the TOE's MDM APIs) can destroy secret keys in the secure key storage.	M		I	
11. import X.509v3 certificates into the Trust Anchor Database	M		M	

Both users and administrators (using the TOE's MDM APIs) can import X.509v3 certificates into the Trust Anchor Database.				
12. remove imported X.509v3 certificates and [***no other certificates***] in the Trust Anchor Database Both users and administrators (using the TOE's MDM APIs) can remove imported X.509v3 certificates from the Trust Anchor Database as well as disable any of the TOE's default Root CA certificates (in the latter case, the CA certificate still resides in the TOE's read-only system partition; however, the TOE will treat that Root CA certificate and any certificate chaining to it as untrusted).	M		I	
13. enroll the TOE in management TOE users can enroll the TOE in management according to the instructions specific to a given MDM. Presumably any enrollment would involve at least some user functions (e.g., install an MDM agent application) on the TOE prior to enrollment.	M	M		
14. remove applications Both users and administrators (using the TOE's MDM APIs) can uninstall user and administrator installed applications on the TOE.	M		M	
15. update system software Users can check for updates and cause the device to update if an update is available. An administrator can use MDM APIs to query the version of the TOE and query the installed applications and an MDM agent on the TOE could issue pop-ups, initiate updates, block communication, etc. until any necessary updates are completed.	M		M	
16. install applications Both users and administrators (using the TOE's MDM APIs) can install applications on the TOE.	M		M	
17. remove Enterprise applications An administrator (using the TOE's MDM APIs) can uninstall Android for Work installed applications on the TOE.	M		M	
18. configure the Bluetooth trusted channel: a. disable/enable the Discoverable mode (for BR/EDR) b. change the Bluetooth device name [**h.** ***disable/enable the Bluetooth services and/or profiles available on the device***] TOE users can enable Bluetooth discoverable mode for a short period of time and can also change the device name which is used for the Bluetooth name. Additional wireless technologies include Android Beam which utilizes NFC and Bluetooth, and can be enabled and disabled by the TOE user.	M		I	I
19. enable/disable display notification in the locked state of: [**f.** ***all notifications***] TOE users and administrator can configure the TOE to allow or disallow notifications while in a locked state.	M		I	I
20. enable data-at rest protection The administrator (using the TOE's MDM APIs) can configure a device encryption policy. Once enabled the TOE internal flash will be encrypted. Users can also encrypt the TOE internal flash even if not enforced by administrator policy. However, if the policy is set the user cannot decrypt the TOE.	M		I	I

21. enable removable media's data-at-rest protection The administrator (using the TOE's MDM APIs) can configure a device removable media encryption policy. Once enabled, files on external SD cards will be encrypted. Users can also encrypt the TOE internal flash even if not enforced by administrator policy. However, if the policy is set the user cannot decrypt the TOE.	M		I	I
22. enable/disable location services: a. across device [**d.** ***no other method***] The administrator (using the TOE's MDM APIs) can disable location services. Unless disabled by the administrator, TOE users can enable and disable location services.	M		I	I
23. Enable/disable the use of [***Biometric Fingerprint***]	M	I		
24. enable/disable all data signaling over [**USB, SD Card**] The administrator (using the TOE's MDM APIs) can disable USB communication features (tethering, debugging, mass storage access, etc.) to prevent data signaling via USB.	I		I	I
25. enable/disable [**Wi-Fi tethering, USB tethering, and Bluetooth tethering**] The administrator (using the TOE's MDM APIs) can individually disable each tethering method. Unless disabled by the administrator, TOE users can individually enable and disable tethering via a Wi-Fi hotspot, USB connection, and Bluetooth pairing. The TOE acts as a server (acting as an access point, a USB Ethernet adapter, and as a Bluetooth Ethernet adapter respectively) in order to share its network connection with another device.	I		I	I
26. enable/disable developer modes The administrator (using the TOE's MDM APIs) can disable Developer Mode. Unless disabled by the administrator, TOE users can enable and disable Developer Mode.	I		I	I
27. enable/disable bypass of local user authentication The administrator can prevent a user from recovering a lost password through their Google account	I		I	I
28. wipe Enterprise data An administrator can remove Android for Work applications and their data	I		I	
29. approve [***import***] by applications of X.509v3 certificates in the Trust Anchor Database The TOE will prompt the User to approve requests by an application to import an X.509v3 CA Certificate into the TOE's Trust Anchor Database. The User, when prompted, can allow or deny the import.	I			
30. configure whether to establish a trusted channel or disallow establishment if the TSF cannot establish a connection to determine the validity of a certificate				
31. enable/disable the cellular protocols used to connect to cellular network base stations				
32. read audit logs kept by the TSF An administrator can access and ready the logs created by the TOE	I		I	
33. configure [selection: certificate, public-key] used to validate digital signature on applications				
34. approve exceptions for shared use of keys/secrets by multiple applications				

35. approve exceptions for destruction of keys/secrets by applications that did not import the key/secret				
36. configure the unlock banner The administrator (using the TOE's MDM APIs) can defined a banner of a maximum of 1300 characters to be displayed while the TOE is locked. There is no method for the user to change the banner.	I		I	I
37. configure the auditable items				
38. retrieve TSF-software integrity verification values				
39. enable/disable [a. ***USB mass storage mode***] The administrator (using the TOE's MDM APIs) can disable USB mass storage mode.	I		I	I
40. enable/disable backup to [***all applications***] *to* [***locally connected system, remote system***] The administrator (using the TOE's MDM APIs) can disable use of Google backup as well as the user of LG backup features (which allow the user to backup application data). Additionally, unless disabled by the administrator, TOE users can configure and use Google and LG backup functionality to back up their application data.	I		I	I
41. enable/disable [**a.** ***Hotspot functionality authenticated by [pre-shared key],*** b. ***USB tethering authenticated by [no authentication]***] The administrator (using the TOE's MDM APIs) can disable the Wi-Fi hotspot and USB tethering. Unless disabled by the administrator, TOE users can configure the Wi-Fi hotspot with a pre-shared key and can configure USB tethering (with no authentication).	I		I	I
42. approve exceptions for sharing data between [*groups of application processes*]	I		I	
43. place applications into application process groups based on [***Android for Work applications and user applications***]	I		I	I
44. Unenroll the TOE from management	I		I	I
45. Enable/disable the Always On VPN protection				
46. Revoke Biometric template				
47. [**a. enable/disable automatic transfer of diagnostic data to an external device or service other than an MDM service with which the device has enrolled** **b. enable/disable automatic updates of system software (see function 17)** **c. wipe non-enterprise data** **d. enable/disable VPN split-tunneling** **e. enable/disable use of removable media** **f. configure implementation of FDP_ACF_EXT.1.2**]	I		I	I

5.1.5.3 Specification of Management Functions (Wireless LAN) (FMT_SMF_EXT.1(1))

FMT_SMF_EXT.1(1).1

The TSF shall be capable of performing the management functions:

Table 3 WLAN Security Management Functions

Management Function **Status Markers:** M – Mandatory I – Implemented	Function	Available to User	Available to Admin	Restricted to Admin
48. configure security policy for each wireless network: a. [***specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s)***] b. security type c. authentication protocol d. client credentials to be used for authentication	M		M	I
49. specify wireless networks (SSIDs) to which the TSF may connect; An administrator can specify a list of wireless network SSIDs to which the TOE may connect and can restrict the TOE to only allow a connection to the specified SSIDs.	I		I	I
50. enable/disable certificate revocation list checking;				
51. disable ad hoc wireless client-to-client connection capability,				
52. disable wireless network bridging capability (for example, bridging a connection between the WLAN and cellular radios on a smartphone so it can function as a hotspot);				
53. disable roaming capability;				
54. enable/disable IEEE 802.1X pre-authentication;				
55. enable/disable and configure PMK caching: a. set the amount of time (in minutes) PMK entries are cached; b. set the maximum number of PMK entries that can be cached.				

5.1.5.4 Extended: Specification of Remediation Actions (FMT_SMF_EXT.2(1))

FMT_SMF_EXT.2(1).1
The TSF shall offer [***wipe of protected data, wipe of sensitive data, remove Enterprise applications, remove all device-stored Enterprise resource data, remove Enterprise secondary authentication data, [remove MDM policies, disable CC mode]***] upon un-enrollment and [***no other triggers***].

5.1.5.5 Extended: Specification of Remediation Actions (FMT_SMF_EXT.2(2))

FMT_SMF_EXT.2(2).1
The TSF shall offer [***remove Enterprise applications, remove all device-stored Enterprise resource data, remove Enterprise secondary authentication data, [full wipe of managed profile data]***] upon **managed profile** un-enrollment and [***no other triggers***].

5.1.6 Protection of the TSF (FPT)

5.1.6.1 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.

5.1.6.2 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.

FPT_AEX_EXT.2.2
The TSF shall prevent write and execute permissions from being simultaneously granted to any page of physical memory [***excluding memory used for JIT (just-in-time) compilation***].

5.1.6.3 Extended: Anti-Exploitation Services (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.

5.1.6.4 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.

5.1.6.5 Application Processor Mediation (FPT_BBD_EXT.1)

FPT_BBD_EXT.1.1
The TSF shall prevent code executing on any baseband processor (BP) from accessing application processor (AP) resources except when mediated by the AP.

5.1.6.6 Extended: JTAG Disablement (FPT_JTA_EXT.1)

FPT_JTA_EXT.1.1
The TSF shall [***control access by a signing key***] to JTAG.

5.1.6.7 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.

5.1.6.8 Extended: No Key Transmission (FPT_KST_EXT.2)

FPT_KST_EXT.2.1
The TSF shall not transmit any plaintext key material outside the security boundary of the TOE.

5.1.6.9 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.

5.1.6.10 Extended: Event Notification (FPT_NOT_EXT.1)

FPT_NOT_EXT.1.1
The TSF shall transition to non-operational mode and [***log failures in the audit record***] when the following types of failures occur:
- failures of the self-tests
- TSF software integrity verification failures
- [***no other failures***].

5.1.6.11 Reliable time stamps (FPT_STM.1)

FPT_STM.1.1
The TSF shall be able to provide reliable time stamps for its own use.

5.1.6.12 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.

5.1.6.13 TSF Cryptographic Functionality Testing (Wireless LAN) (FPT_TST_EXT.1(1))

FPT_TST_EXT.1(1).1
The [***TOE platform***] shall run a suite of self-tests during initial start-up (on power on) to demonstrate the correct operation of the TSF.
FPT_TST_EXT.1(1).2
The [***TOE platform***] shall provide the capability to verify the integrity of stored TSF executable code when it is loaded for execution through the use of the TSF-provided cryptographic services.

5.1.6.14 Extended: TSF Integrity Testing (FPT_TST_EXT.2(1))

FPT_TST_EXT.2(1).1
The TSF shall verify the integrity of the bootchain up through the Application Processor OS kernel stored in mutable media prior to its execution through the use of of [***an immutable hardware hash of an asymmetric key***]. (TD0148 applied)

5.1.6.15 Extended: TSF Integrity Checking (FPT_TST_EXT.2(2))

FPT_TST_EXT.2(2).1
The TSF shall verify the integrity of [***[***
/system/app/FidoASM/FidoASM.apk
/system/app/FidoAuthenticator/FidoAuthenticator.apk
/system/app/FidoClient/FidoClient.apk
/system/app/LGATCMDService/LGATCMDService.apk
/system/app/LGSmartcardService.apk
/system/app/LGVZWStartupwizard/LGVZWStartupwizard.apk
/system/bin/atd
/system/bin/eut_init
/system/bin/fcheck
/system/bin/fidod
/system/bin/fingerprintd
/system/bin/lgkmd
/system/bin/sdc
/system/bin/sdpd
/system/etc/fota/data.dat
/system/etc/fota/public.crt
/system/framework/bouncycastle.jar

/system/framework/com.lge.services.jar
/system/framework/com.lge.smartcard.jar
/system/framework/framework.jar
/system/framework/org.simalliance.openmobileapi.jar
/system/framework/services.jar
/system/lib/hw/fingerprint.default.so
/system/lib/libatd_common.so
/system/lib/libatd_corelib.so
/system/lib/libatd_modeminterface.so
/system/lib/libccm.so
/system/lib/libcrypto.so
/system/lib/libdrmframework.so
/system/lib/libfidoservice_binder.s
/system/lib/liblgdrm.so
/system/lib/liblgkm.so
/system/lib/libsdpd.so
/system/lib/libsmm.so
/system/lib/libssl.so
/system/lib/libsurfaceflinger.so
/system/lib64/hw/fingerprint.default.so
/system/lib64/libatd_corelib.so
/system/lib64/libccm.so
/system/lib64/libcrypto.so
/system/lib64/libdrmframework.so
/system/lib64/libdrmframework_jni.so
/system/lib64/libfidoservice_binder.so
/system/lib64/liblgdrm_client.so
/system/lib64/liblgdrm_jni.so
/system/lib64/liblgkm.so
/system/lib64/libsdpd.so
/system/lib64/libssl.so
/system/lib64/libsurfaceflinger.so
/system/priv-app/HLC/HLC.apk
/system/priv-app/LGStartupwizard/LGStartupwizard.apk
/system/vendor/lib/drm/libdrmwvmplugin.so
/system/vendor/lib/libWVStreamControlAPI_L1.so
/system/vendor/lib/libWVStreamControlAPI_L3.so
/system/vendor/lib/libwvdrm_L3.so
/system/vendor/lib/libwvm.so
]], stored in mutable media prior to its execution through the use of [***an immutable hardware hash of an asymmetric key***]. (TD0148 applied)

5.1.6.16 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.

5.1.6.17 Extended: Trusted Update Verification (FPT_TUD_EXT.2)

FPT_TUD_EXT.2.1
The TSF shall verify software updates to the Application Processor system software and [***[baseband processor]***] using a digital signature by the manufacturer prior to installing those updates.

FPT_TUD_EXT.2.2
The TSF shall [***update only by verified software***] the TSF boot integrity [***key, hash***].

FPT_TUD_EXT.2.3
The TSF shall verify that the digital signature verification key used for TSF updates [***matches an immutable hardware public key***].

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

5.1.7 TOE access (FTA)

5.1.7.1 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.

FTA_SSL_EXT.1.2
The TSF shall transition to a locked state after initiation by either the user or the administrator.

FTA_SSL_EXT.1.3
The TSF shall, upon transitioning to the locked state, perform the following operations: a) clearing or overwriting display devices, obscuring the previous contents; b) [***lock sound is played***].

5.1.7.2 Default TOE Access Banners (FTA_TAB.1)

FTA_TAB.1.1
Before establishing a user session, the TSF shall display an advisory warning message regarding unauthorized use of the TOE.

5.1.7.3 Wireless Network Access (FTA_WSE_EXT.1)

FTA_WSE_EXT.1.1
The TSF shall be able to attempt connections only to wireless networks specified as acceptable networks as configured by the administrator in FMT_SMF_EXT.1.1/WLAN.

5.1.8 Trusted path/channels (FTP)

5.1.8.1 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, provides assured identification of its end points, protects channel data from disclosure and detection of modification of the channel data.

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

FTP_ITC_EXT.1.3
The TSF shall initiate communication via the trusted channel for wireless access point connections, administrative communication, configured enterprise connections, and [***no other connections***].

5.1.8.2 Trusted Channel Communication (Wireless LAN) (FTP_ITC_EXT.1(1))

FTP_ITC_EXT.1(1).1
The TSF shall use 802.11-2012, 802.1X, and EAP-TLS to provide a trusted communication channel between itself and a wireless access point that is logically distinct from other communication channels, provides assured identification of its end points, protects channel data from disclosure, and detects modification of the channel data.

FTP_ITC_EXT.1(1).2
The TSF shall initiate communication via the trusted channel for wireless access point connections.

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 4 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 categorization 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:

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

6.1 Security audit

FAU_GEN.1: The table in section 9 (Audit Logging) of the LG Android 7 devices (G6) guidance depicts the list of all auditable events (for MDFPP30/WLANCEP10) and shows the common format of audit records and a description of each field.

FAU_SAR.1: The TOE stores its audit records in the /data/ccaudit/ccaudit.log* and protects these logs by making them write inaccessible (file permissions) to any application other than the TSF itself.

FAU_STG.1: The TOE protects audit records as read-only to prevent modification or deletion of the records.

FAU_STG.4: The TOE maintains a set of five ccaudit.log files on Flash, and the TOE rotates logs (moving ccaudit.log to ccaudit.log.1 when a log has exceeded a 4MiB size). When the TOE rotates logs, it increases each log file's suffix, discarding the oldest log in order to make room for new audit events. The five ccaudit logs provide the TOE with 20MiB of audit storage.

6.2 Cryptographic support

FCS_CKM.1(1): The TOE provides generation of asymmetric keys including

Algorithm	Key/Curve Sizes	Usage
RSA, FIPS 186-4	1024/2048/3072	API/Application
ECDSA, FIPS 186-4	P-256/384/521	API/Application
ECDHE keys (not domain parameters)	P-256/384/521	TLS KeyEx (WPA2 & HTTPS) & Sensitive Data Protection (DAR.2)
DHE keys (not domain parameters)	1024/2048	TLS KeyEx (WPA2 & HTTPS)

Table 5 Asymmetric Key Generation

The TOE's cryptographic algorithm implementations have received NIST algorithm certificates (please see the table in FCS_COP.1 for all of the TOE's algorithm certificates). The TOE itself does not generate any RSA/ECDSA authentication key pairs for TOE functionality (the user or administrator must load certificates for use with WPA2 with EAP-TLS authentication); however, the TOE provides key generation APIs to mobile applications to allow them to generate such key pairs. The TOE generates DH/ECDH key pairs (but not domain parameters) for use in TLS Key Exchange and also generates ECDH keys to support storage of application sensitive data while the mobile device is locked.

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 derivation of AES 128-bit Transmission Key while the TOE uses CAVP validated algorithms (HMAC-SHA-1 and the SHA-256 Hash_DRBG) provided by the TOE's BoringSSL implementation. The TOE has been designed for compliance, and the Device has successfully completed certification (including WPA2 Enterprise) and received WiFi CERTIFIED Interoperability Certificates from the WiFi Alliance (see the table in section 1.4). The WiFi Alliance maintains a website providing further information about the testing program: http://www.wi-fi.org/certification

FCS_CKM.2(1): The TOE performs key establishment as part of EAP-TLS and TLS session establishment. **Table 5 Asymmetric Key Generation** enumerates the TOE's supported key establishment implementations (RSA/DH/ECDH for TLS/EAP-TLS).

FCS_CKM.2(2): The uses ECDH with a P-521 curve key establishment for protection of application sensitive data received while the device is locked.

FCS_CKM.2(3): The TOE adheres to RFC 3394, SP 800-38F, and 802.11-2012 standards and unwraps the Group Temporal Key (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 GTK.

FCS_CKM_EXT.1: The TOE includes a Root Encryption Key (REK) stored in a 256-bit fuse bank within the application processor. The TOE generates the REK/fuse value during manufacturing using its hardware DRBG. The application processor protects the REK by preventing any direct observation of the value and prohibiting any ability to modify or update the value. The application processor loads the fuse value into an internal hardware crypto register and the Trusted Execution Environment (TEE) provides trusted applications the ability to derive KEKs from the REK. The TEE does not allow trusted applications to use the REK for encryption or decryption, only the ability to derive a KEK from the REK. The TOE includes a TEE application that calls into the TEE in order to derive a KEK from the 256-bit REK/fuse value and then only permits use of the derived KEK for encryption and decryption as part of the TOE key hierarchy. 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 SD Card encryption keys, sensitive data encryption keys, or for the Secure Key Storage), the TOE utilizes the RAND_bytes() API call from its BoringSSL SHA-256 Hash_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 (SDC-FEKs), Sensitive Data Protection (SDP) files (SDP-FEKs), and the Data-at-Rest encryption key, the TOE utilizes its Application Processor Module's SHA-256 Hash_DRBG to generate the AES keys (using the qrng_get_random API for the FEKs and utilizing the Application Processor's Hash_DRBG directly in TrustZone).

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: 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 BoringSSL SHA-256 Hash_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 on when transitioning to the device's locked state (in the case of the Sensitive Data Protection keys). 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, the 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 secure direct overwrite (BLKSECDISCARD ioctl) of the wear-leveled 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 (either protected data or sensitive data) and upon encrypted files resident on the SD Card. Upon request, the TOE cryptographically erases the Data-At-Rest DEK protecting the user data partition, the SDPK protecting sensitive data files in the user data partition, and the SDEK protecting SD Card files/FEKs, clears those keys from memory, reformats the partition, and then reboots. The TOE's clearing of the keys follows the requirements of FCS_CKM_EXT.4.

FCS_CKM_EXT.6: The TOE generates salt nonces (which are just salt values used in WPA2) using its SHA-256 Hash_DRBG.

Salt value and size	RBG origin	Salt storage location
User password salt (128-bit)	BoringSSL's SHA-256 Hash_DRBG	Flash filesystem
TLS client_random (256-bit)	BoringSSL's SHA-256 Hash_DRBG	N/A (ephemeral)
TLS pre_master_secret (384-bit)	BoringSSL's SHA-256 Hash_DRBG	N/A (ephemeral)
TLS DHE/ECDHE private value (256, 384, 512)	BoringSSL's SHA-256 Hash_DRBG	N/A (ephemeral)
WPA2 4-way handshake supplicant nonce (SNonce)	BoringSSL's SHA-256 Hash_DRBG through wpa_supplicant	N/A (ephemeral)

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

The TOE's BoringSSL Library (version 1.0) provides the following algorithms:

Algorithm	NIST Standard	SFR Reference	Cert#
AES 128/256 CBC, GCM, KW	FIPS 197, SP 800-38A/D/F	FCS_COP.1(1)	4331
CVL KAS ECC/FFC	SP 800-56A	FCS_CKM.2(1)	1037
DRBG Hash SHA-256	SP 800-90A	FCS_RBG_EXT.1(1)	1381

DSA KeyPair 2048/3072	SP 800-56A	FCS_CKM.1(1)
		FCS_CKM.2(1)
			1151

ECDSA PKG/PKV/SigGen/SigVer	FIPS 186-4	FCS_CKM.1(1)
		FCS_CKM.1(2)
		FCS_COP.1(3)
			1024

HMAC SHA-1/256/384/512	FIPS 198-1 & 180-4	FCS_COP.1(4)	2868

RSA SIG(gen)/SIG(ver)/Key(gen)	FIPS 186-4	FCS_CKM.1(1)
		FCS_CKM.1(2)
		FCS_COP.1(3)
			2337

SHS SHA-1/256/384/512	FIPS 180-4	FCS_COP.1(2)	3572

Table 6 BoringSSL Cryptographic Algorithms

The TOE's Kernel Loadable Cryptographic Module (version 1.0 compiled for A64+CE) provides the following cryptographic algorithms:

Algorithm	NIST Standard	SFR Reference	Cert#
AES 128/256 CBC	FIPS 197, SP 800-38A	FCS_COP.1(1)	3975
HMAC SHA-1/256	FIPS 198-1 & 180-4	FCS_COP.1(4)	2593
SHS SHA-1/256	FIPS 180-4	FCS_COP.1(2)	3280

Table 7 Kernel Loadable Cryptographic Module Algorithms

Please note that the above CAVP certificates resulted from testing on the Qualcomm Snapdragon 820 (MSM8996) ARMv8 w/ Linux Kernel 3.18, and per NIAP Policy #5, the TOE matches the software version (kernel 3.18) and while the CPU is different, the Qualcomm Snapdragon 821 is in the same family as the 820. Hence the certificates remain valid for the TOE.

The TOE's Wi-Fi chipset provides an AES-CCMP implementation, and the TOE's application processor (Snapdragon 821[8996 Pro], based upon the Snapdragon 820) provides additional cryptographic algorithms. Please note that even though the algorithm certificates are for the Snapdragon 820, the Snapdragon 821 does not change the underlying cryptographic engines, and thus the algorithm certificates remain valid for the Snapdragon 821.

Algorithm	NIST Standard	SFR Reference	Cert#

Algorithm	NIST Standard	SFR Reference	Cert#
AES 128 CCM	FIPS 197, SP 800-38C	FCS_COP.1(1)	3678
AES 128/256 CBC	FIPS 197, SP 800-38A	FCS_COP.1(1)	3526
AES 128/256 XTS	FIPS 197, SP 800-38E	FCS_COP.1(1)	3556, 3557
HMAC SHA-1/256	FIPS 198-1 & 180-4	FCS_COP.1(4)	2254
DRBG SHA-256 Hash_DRBG	SP 800-90A	FCS_RBG_EXT.1(2)	885
SHS SHA-256	FIPS 180-4	FCS_RBG_EXT.1(2)	2908/2930
SHS SHA-1/256	FIPS 180-4	FCS_COP.1(2)	2909

Table 8 Hardware 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’s BoringSSL library supports the TOE’s cryptographic Android Runtime (ART) methods (through Android’s conscrypt JNI provider) afforded to mobile applications and also supports Android user-space processes and daemons (e.g., wpa_supplicant). The TOE”s Kernel Loadable Cryptographic Module provides cryptography to the kernel and other kernel loadable modules (for example, eCryptfs used for SD Card encryption). The TOE”s Application Processor provides hardware accelerated cryptography utilized in Data-At-Rest (DAR) encryption of the user data partition.

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. The time needed to derive keying material does not impact or lessen the difficulty faced by an attacker’s exhaustive guessing as the combination of the password derived KEK with REK value entirely prevents offline attacks and the TOE’s maximum incorrect login attempts (between 1 and 50 incorrect attempts with 4 character, minimum, passwords) prevents exhaustive online attacks.

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/output MAC lengths of 512/160, 512/160, and 1024/384-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 BoringSSL SHA-256 Hash_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 Application Processor provided SHA-256 Hash_DRBG. In all cases, the TOE uses AES-CBC mode and generates the IVs in compliance with the requirements of table 14 of MDFPP30/WLANCEP10.

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. An SHA-256 Hash_DRBG provided by BoringSSL. Note that while the TOE includes implementations of Hash_DRBG (using any size SHA) and HMAC_DRBG (again using size SHA), the TOE (and its current system level applications) makes use of only an SHA-256 Hash_DRBG. Furthermore the TOE provides mobile applications access (through an Android Java API) to random data drawn from its SHA-256 Hash_DRBG. However, future (system-level) applications could utilize other DRBG variants supported by the TOE’s BoringSSL module.

The TOE initializes each RBG with sufficient entropy ultimately accumulated from a TOE-hardware-based noise source. The TOE uses its hardware-based noise source to continuously fill the Linux Kernel Random Number Generator input pool, and in turn, the TOE draws from /dev/random to seed both its SHA-256 Hash_DRBGs. The TOE seeds its SHA-256 Hash_DRBG using 440-bits of data from /dev/random, thus ensuring at least 256-bits of entropy. Similarly, the TOE seeds its SHA-256 Hash_DRBG with 440-bits of data from /dev/hw_random, also ensuring that it contains at least 256-bits of entropy. The TOE uses its Application Processor DRBG when generating FEKs for SDP and SD card encryption, and uses its BoringSSL DRBG for all other uses.

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 BoringSSL API, and through the application processor module (user and kernel) APIs.

FCS_STG_EXT.1: The TOE provides the user, administrator, 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 (and which explicitly declare a shared UUID in their application manifest) 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.

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_TLSC_EXT.1: The TOE provides mobile applications (through its Android API) the use of TLS version 1.2 including support for all sixteen selectable ciphersuites (see the selections in section 5.1.2.24).

When an application uses the APIs provided in the Admin Guide to attempt to establish a trusted channel connection based on EAP-TLS, TLS or HTTPS, the TOE supports Common Name (CN) and Subject Alternative Name (SAN) (DNS and IP address) as reference identifiers. The TOE supports client (mutual) authentication. The TOE in its evaluated configuration and by design, supports only evaluated elliptic curves for TLS (P-256, P-384, & P-521 and no others) and requires/allows no configuration of the supported curves. While the TOE supports the use of wildcards in X.509 reference identifiers (CN and SAN), the TOE does not support certificate pinning. If the TOE cannot determine the revocation status of a peer certificate, the TOE will not establish the connection.

FCS_TLSC_EXT.1(1): The TSF supports TLS versions 1.0, 1.1, and 1.2 and also supports the selected ciphersuites utilizing SHA (see the selections in section 5.1.2.25) for use with EAP-TLS as part of WPA2. The TOE in its evaluated configuration and by design, supports only evaluated elliptic curves (P-256, P-384, & P-521 and no others) and requires/allows no configuration of the supported curves.

The TOE, allows the user to load and utilize authentication certificates for EAP-TLS used with WPA. The Android UI (Settings->Security->Credential storage: Install from device storage) allows the user to import an RSA or ECDSA certificate and designate its use as WiFi.

6.3 User data protection

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

1. normal – A lower-risk permission that gives an application access to isolated application-level features, with minimal risk to other applications, the system, or the user. The system automatically grants this type of permission 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).
2. dangerous – A higher-risk permission that would give a requesting application access to private user data or control over the device that can negatively impact the user. Because this type of permission introduces potential risk, the system may not automatically grant it to the requesting application. For example, any dangerous permissions requested by an application may be displayed to the user and require confirmation

before proceeding, or some other approach may be taken to avoid the user automatically allowing the use of such facilities.

3. signature – A permission that the system is to grant 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.
4. signatureOrSystem – A permission that the system is to grant only to packages in the Android system image *or* that are signed with the same certificates. Please avoid using this option, as the signature protection level should be sufficient for most needs and works regardless of exactly where applications are installed. This permission is used for certain special situations where multiple vendors have applications built in to a system image which need to share specific features explicitly because they are being built together.

An example of a normal permission is the ability to vibrate the device: `android.permission.VIBRATE`. This permission allows an application to make the device vibrate, and an application that does not declare this permission would have its vibration requests ignored.

An example of a dangerous privilege would be access to location services to determine the location of the mobile device: `android.permission.ACCESS_FINE_LOCATION`. The TOE controls access to Dangerous permissions during the installation of the application. The TOE prompts the user to review the application's requested permissions (by displaying a description of each permission group, into which individual permissions map, that an application requested access to). If the user approves, then the mobile device continues with the installation of the application. Thereafter, the mobile device grants that application during execution access to the set of permissions declared in its Manifest file.

An example of a signature permission is the `android.permission.BIND_VPN_SERVICE`, that an application must declare in order to utilize the VpnService APIs of the device. Because the permission is a Signature permission, the mobile device only grants this permission to an application that requests this permission *and* that has been signed with the same developer key used to sign the application declaring the permission (in the case of the example, the Android Framework itself).

An example of a systemOrSignature permission is the `android.permission.LOCATION_HARDWARE`, which allows an application to use location features in hardware (such as the geofencing API). The device grants this permission to requesting applications that either have been signed with the same developer key used to sign the android application declaring the permissions or that reside in the "system" directory within Android, which for Android 4.4 and above, are applications residing in the /system/priv-app/ directory on the read-only system partition). Put another way, the device grants systemOrSignature permissions by Signature or by virtue of the requesting application being part of the "system image."

Additionally, Android includes the following flags that layer atop the base categories.

1. privileged – this permission can also be granted to any applications installed as privileged apps on the system image. Please avoid using this option, as the signature protection level should be sufficient for most needs and works regardless of exactly where applications are installed. This permission flag is used for certain special situations where multiple vendors have applications built in to a system image which need to share specific features explicitly because they are being built together.
2. system – Old synonym for "privileged".
3. development – this permission can also (optionally) be granted to development applications (e.g., to allow additional location reporting during beta testing).
4. appop – this permission is closely associated with an app op for controlling access.
5. pre23 - this permission can be automatically granted to apps that target API levels below API level 23 (Marshmallow/6.0).
6. installer - this permission can be automatically granted to system apps that install packages.
7. verifier- this permission can be automatically granted to system apps that verify packages.
8. preinstalled – this permission can be automatically granted any application pre-installed on the system image (not just privileged apps) (the TOE does not prompt the user to approve the permission).

For older applications (those targeting Android's pre-23 API level, i.e., API level 22 [lollipop] and below), the TOE will prompt a user at the time of application installation whether they agree to grant the application access to the requested services. Thereafter (each time the application is run), the TOE will grant the application access to the services specified during install.

For newer applications (those targeting API level 23 or later), the TOE grants individual permissions at application run-time by prompting the user for confirmation of each permissions category requested by the application (and only granting the permission if the user chooses to grant it).

Finally, the TOE supports Android for Work profiles to provide additional separation between application and application data belonging to the Android for Work profile. Please see the Admin Guide for additional details regarding how to set up and use Android for Work profiles.

FDP_DAR_EXT.1: The TOE provides Data-At-Rest AES-128 XTS hardware 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-128 bit DEK with XTS feedback mode to encrypt the entire partition using dedicated application processor hardware. The TOE also provides AES-256 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. The TOE uses its application processor AES CBC hardware for SD Card file encryption.

FDP_DAR_EXT.2: The TOE provides APIs to allow mobile applications to designate data files as sensitive. The TOE then protects that data file such that when the device is locked, the sensitive data cannot be read. The TOE accommodates additional sensitive data received while the device is locked by using an ECDH key exchange between an application wishing to store sensitive data and the TSF's ECDH key. An application (by calling the API) will trigger the generation of a new ECDH key for the application, use. The TSF encrypts the file data and prepends a header to indicate that the data is encrypted.

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 and any additional user or admin/MDM loaded certificates. The built-in certs are individually stored in device's read-only system image in the `/system/etc/security/cacerts` directory, and the user can individually disable certs through Android's user interface [Settings->Security-> Trusted Credentials]. Because the built-in CA certificates reside on the read-only system partition, the TOE places a copy of any disabled built-in certificate into the `/data/misc/user/X/cacerts-removed/` directory, where "X" represents the user's number (which starts at 0). The TOE stores added CA certificates in the corresponding `/data/misc/user/X/cacerts-added/` directory and also stores a copy of the CA certificate in the user's Secure Key Storage (residing in the `/data/misc/keystore/user_X/` directory).

FDP_UPC_EXT.1: The TOE provides APIs allowing non-TSF applications (mobile applications) the ability to establish a secure channel using TLS, HTTPS, and Bluetooth DR/EDR and LE. Mobile applications can use the following Android APIs for TLS, HTTPS, and Bluetooth respectively:

`javax.net.ssl.SSLContext`:

http://developer.android.com/reference/javax/net/ssl/SSLContext.html

`javax.net.ssl.HttpsURLConnection`:

http://developer.android.com/reference/javax/net/ssl/HttpsURLConnection.html

`android.bluetooth`:

http://developer.android.com/reference/android/bluetooth/package-summary.html

6.4 Identification and authentication

FIA_AFL_EXT.1: The TOE maintains in persistent storage, for each user, the number of failed password logins since the last successful login (the phone, in CC mode, only supports password authentication), 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 for the cryptlock screen from the default of ten failed logins to a value between one and 50 through an MDM. The TOE validates passwords by providing them to Android's Gatekeeper (which runs in the Trusted Execution Environment), if the presented password fails to validate, the TOE increments the incorrect password counter before displaying a visual error to the user.

FIA_BLT_EXT.1: The TOE requires explicit user authorization before it will pair with a remote Bluetooth device. When pairing with another device, the TOE requires that the user either confirm that a displayed numeric passcode matches between the two devices or that the user enter (or choose) a numeric passcode that the peer device generates (or must enter). The TOE requires this authorization (via manual input) for mobile application use of the Bluetooth trusted channel and in situations where temporary (non-bonded) connections are formed.

FIA_BLT_EXT.2: The TOE prevents data transfer of any type until Bluetooth pairing has completed. Additionally, the TOE supports OBEX (OBject Exchange) through L2CAP (Logical Link Control and Adaptation Protocol).

FIA_BLT_EXT.3: The TOE rejects duplicate Bluetooth connections by only allowing a single connection per paired device.

FIA_BLT_EXT.4: The TOE's Bluetooh host and controller supports Bluetooth Secure Simple Pairing and the TOE utilizes this pairing method when the remote host also supports it.

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 the user through a password consisting of basic Latin characters (upper and lower case, numbers, and the special characters noted in the selection (see section 5.1.4.7)). 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. Note that while the Smart Lock feature allows additional authentication mechanisms, these features have not been evaluated and LG disables Smart Lock in CC mode.

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.5: The TOE, in CC mode, allows the user to authenticate using only a password. The TOE prohibits other authentication mechanisms, such as pattern, PIN, fingerprint, or Smart Lock mechanisms (on-body detection, trusted places, trusted devices, trusted face, trusted voice).

FIA_UAU.6: 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. LG disables Smart Lock while in CC mode. Only after entering their current user password can the user then elect to change their password.

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), allows a user to perform the actions assigned in FIA_UAU_EXT.2.1 (see section 5.1.4.14) without first successfully authenticating. Choosing the input method allows the user to select between different keyboard devices (say, for example, if the user has installed addition keyboards). Note that the TOE automatically names and saves (to the internal Flash) any screen shots or photos taken from the lock screen, and the TOE provides the user no opportunity to name them or change where they are stored.

Beyond those actions, a user cannot perform any other actions other than observing notifications displayed on the lock screen until after successfully authenticating.

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/FIA_X509_EXT.2(1)/WLAN: The TOE uses X.509v3 certificates during EAP-TLS, TLS, and HTTPS. The TOE comes with a built-in set of default 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 Wi-Fi connections and HTTPS used to download OTA updates), 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 OCSP]). If, during the process of certificate verification, the TOE cannot establish a connection with the server acting as the OCSP Responder, the TOE will deem the server's certificate as invalid and not establish a TLS connection with the server.

The TOE, when acting as a WPA2 supplicant uses X.509 certificates for EAP-TLS authentication. Because the TOE may not have network connectivity to a revocation server prior to being admitted to the WPA2 network and because the TOE cannot determine the IP address or hostname of the authentication server (the Wi-Fi access point proxies the supplicant's authentication request to the server), the TOE will accept the certificate of 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. http://developer.android.com/reference/java/security/cert/CertPathValidator.html

6.5 Security management

FMT_MOF_EXT.1/FMT_SMF_EXT.1: The TOE provides the management functions described in **Table 2 Security Management Functions** in section 5.1.5.2. The table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted. 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_EXT.1(1): The TOE provides the management functions described in **Table 3 WLAN Security Management Functions** in section 5.1.5.3. As with **Table 2 Security Management Functions**, the table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted. 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_EXT.2: The TOE offers MDM agents the ability to wipe protected data, to remove Enterprise applications, to remove MDM policies, and to disable CC mode upon unenrollment. The TOE offers MDM agents the ability to wipe protected data (effectively wiping the device) at any time. Similarly, the TOE also offers the ability to remove Enterprise applications and a full wipe of managed profile data of the TOE's Android for Work data/applications at any time.

6.6 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 7.0 operating system utilizes a 3.18 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 ARMv8 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 D.5 of the ARMv8 Architecture Reference Manual contains additional details about the MMU of ARM-based processors: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0487a.f/index.html

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 hardware-based eXecute-Never 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).

The TOE, in its evaluated CC mode, prevents a user from installing a new software image via USB, as CC mode prevents the phone from entering its download mode during boot. Thus, in CC mode, the phone can only update via FOTA.

FPT_BBD_EXT.1: The TOE's hardware and software architecture ensures separation of the application processor (AP) from the baseband or communications processor (CP) through internal controls of the TOE's SoC, which contains both the AP and the CP.

FPT_JTA_EXT.1: The TOE's prevents access to its processor's JTAG interface by requiring use of a signing key to authenticate prior to gaining JTAG access. Only a JTAG image with the accompanying device serial number (which is different for each mobile device) that has been signed by LG's private key can be used to access a devie's JTAG interface. The LG private key corresponds to the LG RSA 2048-bit public key (a SHA-256 hash of which is fused into the TOE's application processor).

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 6.2 of 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 BoringSSL, application processor cryptography (which leverages AP hardware), and the following system-level executables that utilize KEKs: dm-crypt, eCryptfs, wpa_supplicant, and the Secure Key Store.

1. dm-crypt provides Data-At-Rest encryption of the user data partition in Flash
2. eCryptfs provides encryption of sensitive data and files residing on external SD Cards.
3. wpa_supplicant provides 802.11-2014/WPA2 services
4. the Secure Key Store application provides key generation, storage, deletion services to mobile applications and to user through the UI.

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.

Furthermore, the components of the device are designed to prevent transmission of key material outside the device. Each internal system component requiring access to a plaintext key (for example the Wi-Fi driver) must have the necessary precursor(s), whether that be a password from the user or file access to key in Flash (for example the encrypted AES key used for encryption of the Flash data partition). With those appropriate precursors, the internal system-level component may call directly to the system-level library to obtain the plaintext key value. The system library in turn requests decryption from a component executing inside the trusted execution environment and then directly returns the plaintext key value (assuming that it can successfully decrypt the requested key, as confirmed by the HMAC-SHA-256 integrity check) to the calling system component. That system component will then utilize that key (in the example, the dm-crypt daemon that holds the key in order to encrypt and decrypt reads and writes to the encrypted user data partition in Flash). In this way, only the internal system components responsible for a given activity have access to the plaintext key needed for the activity, and that component receives the plaintext key value directly from the system library.

For a user's mobile applications, those applications do not have any access to any system-level components and only have access to keys that the application has imported into the Secure Key Store. Upon requesting access to a key, the mobile application receives the plaintext key value back from the system library through the Android API. Mobile applications do not have access to the memory space of any other mobile application so it is not possible for a malicious application to intercept the plaintext key value to then log or transmit the value off the device.

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 (which installs and verifies APK signatures and certificates), 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 (unless a system application is residing in /system/priv-app or signed by the vendor) 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 (application processor and BoringSSL) performs 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	BoringSSL	Comparison of known answer to calculated valued
ECDH key agreement	BoringSSL	Comparison of known answer to calculated valued
DRBG random bit generation	BoringSSL	Comparison of known answer to calculated valued
ECDSA sign/verify	BoringSSL	Comparison of known answer to calculated valued
HMAC-SHA	BoringSSL	Comparison of known answer to calculated valued
RSA sign/verify	BoringSSL	Comparison of known answer to calculated valued
SHA hashing	BoringSSL	Comparison of known answer to calculated valued
AES encryption/decryption	Kernel Cryptographic	Comparison of known answer to calculated valued
HMAC-SHA	Kernel Cryptographic	Comparison of known answer to calculated valued
SHA hashing	Kernel Cryptographic	Comparison of known answer to calculated valued
AES encryption/decryption	Application Processor	Comparison of known answer to calculated valued
HMAC-SHA	Application Processor	Comparison of known answer to calculated valued
DRBG random bit generation	Application Processor	Comparison of known answer to calculated valued
SHA hashing	Application Processor	Comparison of known answer to calculated valued
AES-XTS encrypt/decrypt	Application Processor	Comparison of known answer to calculated valued

Table 9 Power-up Cryptographic Algorithm Known Answer Tests

FPT_TST_EXT.1(1): The TOE platform performs the previously mentioned self-tests to ensure the integrity of the WLAN client (wpa_supplicant) and the cryptographic libraries that it uses.

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. The TOE also checks the integrity of the system libraries and applications described in section 5.1.6.14. Additionally, the TOE performs checking of the entire /system partition through use of Android's dm_verity mechanism (and while the TOE will still operate, it will log any blocks/executables that have been modified).

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 FOTA (Firmware Over The Air) updates to the TOE software (which includes baseband processor updates) 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.

The TOE also provides roll-back protection to prevent a user from installing a prior/previous version of software.

ALC_TSU_EXT: LG provides its customers with an email address (support-enterprise-mobility@lge.com) and PGP public key to allow researchers to securely report any potential issues. LG analyzes the feedback from its customers, and takes appropriate action 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 resolve reported issues as quickly as possible, and makes these updates available to the wireless carriers fielding LG phones. LG seeks to resolve issues in approximately 90 days. Issues reported to Google directly are handled through Google's notification processes.

6.7 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 5.1.4.14).

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 and unlock the screen. After successfully authenticating at the initial power-up Data-At-Rest login screen, the user can access the TOE and upon (re)locking the TOE, the TOE will present the user the normal lock screen (which displays and allows the actions described in FIA_UAU_EXT.2.1 (see section 5.1.4.14).

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, the security type, authentication protocol, and the client credentials to be used for authentication. 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.

6.8 Trusted path/channels

FTP_BLT_EXT.1: The TOE requires the Bluetooth channel be encrypted and will terminate the connection if the remote device stops encryption while connected (as the TOE perceived the data stream to have been corrupted).

FTP_ITC_EXT.1/FTP_ITC_EXT.1(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 communications via the WPA2 (802.11-2012, 802.1X with EAP-TLS) trusted channel for connection to a wireless access point. The TOE provides mobile applications access to HTTPS and 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	Security Function
dalvikvm	/system/bin	Virtual Machine
keystore	/system/bin	Key Store
qseecomd	/system/bin	Trustzone Daemon
time_daemon	/system/bin	Time
vold	/system/bin	DAR
wpa_supplicant	/system/bin	WPA2
ccmd	/system/bin	CC Mode Daemon
libcrypto.so	/system/lib	Crypto
libcrypto.so	/system/lib64	Crypto
libfotajni.so	/system/lib	FOTA JNI
libfotajni.so	/system/lib64	FOTA JNI
libjavacrypto.so	/system/lib	Crypto JNI
libjavacrypto.so	/system/lib64	Crypto JNI
libkeystore_binder.so	/system/lib	KeyStore
libkeystore_binder.so	/system/lib64	KeyStore
libkeyutils.so	/system/lib64	DAR
libecryptfs.so	/system/lib64	DAR
libsoftkeymaster.so	/system/lib	Key Store
libsoftkeymaster.so	/system/lib64	Key Store
libssl.so	/system/lib	SSL/TLS
libssl.so	/system/lib64	SSL/TLS
liblgkm.so	/system/lib	Key Management
liblgkm.so	/system/lib64	Key Management
lgkm.msm8992, lgkm.msm8996	/system/lib/hw	H/W Key Storage HAL
lgkm.msm8992, lgkm.msm8996	/system/lib64/hw	H/W Key Storage HAL
libccm.so	/system/lib	CC Mode
libccm.so	/system/lib64	CC Mode
libsecureks.so	/system/lib	H/W Key Storage
libsecureks.so	/system/lib64	H/W Key Storage