HTC A9 Secured by Cog Systems D4 (MDFPP20) Security Target

Version 0.5
May 12, 2017

Prepared for:

Cog Systems
Level 1, 277 King Street
Newtown NSW 2042
Australia

Prepared By:

Gossamer Laboratories
www.gossamersec.com

LIST OF TABLES

1. Security Target Introduction

This section identifies the Security Target (ST) and Target of Evaluation (TOE) identification, ST conventions, ST conformance claims, and the ST organization. The TOE is Secure Communications Mobile provided by Cog Systems. The TOE is being evaluated as a mobile device.

The Security Target contains the following additional sections:

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

Conventions

The following conventions have been applied in this document:

- Security Functional Requirements – Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.
  - Iteration: allows a component to be used more than once with varying operations. In the ST, iteration is indicated by a parenthetical number placed at the end of the component. For example FDP_ACC.1(1) and FDP_ACC.1(2) indicate that the ST includes two iterations of the FDP_ACC.1 requirement.
  - 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 …").
- 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 – D4 Secure Mobile (MDFPP20) Security Target

ST Version – Version 0.5

ST Date – May 12, 2017

1.2 TOE Reference

TOE Identification – HTC A9 Secured by Cog Systems D4 Secure Mobile

TOE Developer – Cog Systems

Evaluation Sponsor – Cog Systems

1.3 TOE Overview

The Target of Evaluation (TOE) is the HTC A9 Secured by Cog Systems D4 Secure Mobile device.

1.4 TOE Description

The Target of Evaluation is a smartphone based upon the HTC A9 hardware which uses Qualcomm SoCs (Snapdragon 617, MSM8952) and runs custom Cog Systems D4 Secure images. This is a custom built smartphone intended to support military and civil service users.

1.4.1 TOE Architecture

The TOE utilizes an OKL4 separation kernel (hypervisor), to provide 'cells' (i.e., a virtualized environment) that virtualize and isolate different aspects of the phone's hardware. The TOE includes the Qualcomm Secure Execution Environment (QSEE) Trustzone, the separation kernel/hypervisor, custom Cog Systems D4 Secure Mobile 'cells' and a high-level operating system, along with the HTC One A9 hardware. While the D4 Secure Mobile can support any operating system, the high level OS included in this evaluation is Android version 6.0.1.

The HTC One A9 hardware is based upon the Qualcomm Snapdragon 617, MSM 8952 System on a Chip (SoC). The SoC includes a Qualcomm Integrated Cryptographic Engine (ICE) to perform encryption and decryption operations with hardware implemented AES algorithm and software configured keys.

The custom D4 Secure Mobile ‘cells’ are shown in Figure 1-1 and described below. The OKL4 separation kernel is also referred to as the Hypervisor. The Hypervisor create the virtualized execution environments referred to as ‘cells’ and enforces strict, pre-defined relationships for the interactions between cells. The QSEE Trustzone provides a hardware isolated environment for cryptographic operations invoked by the Android cell and the Block Cell (see below).

The following figure depicts the “Cells” in the running TOE and identifies the scope of this evaluation. The TOE is packaged to include all of the pieces shown in Figure 1-1 except the 3rd party VPN client. The “blue” boxes in Figure 1-1 represent software that is part of the TOE. The “black” box represent the mobile device hardware. Finally, the “white” boxes are TOE software providing functionality not included in this evaluation, which is subject to its own separate evaluations (they do not support nor affect any of the claimed security functions).

In Figure 1-1 TOE Architecture, several cells are denoted as “Linux Cell”. This indicates that a Linux kernel is running within the cell, providing an environment for the other functionality offered by that cell. Those boxes labeled “C Cell” represent a cell running customized C language code.

The TOE’s Android cell includes a high-level OS. This high-level OS is a full implementation of Android 6.0.1, modified as necessary to satisfy MDFPP20 requirements. The Android cell provides an Application Programming Interface to mobile applications and provides users installing an application with the ability to either approve or reject an application based upon the API access that the application requires. The Android cell provides many (but not all) of the TOE security functions required by the MDFPP20.

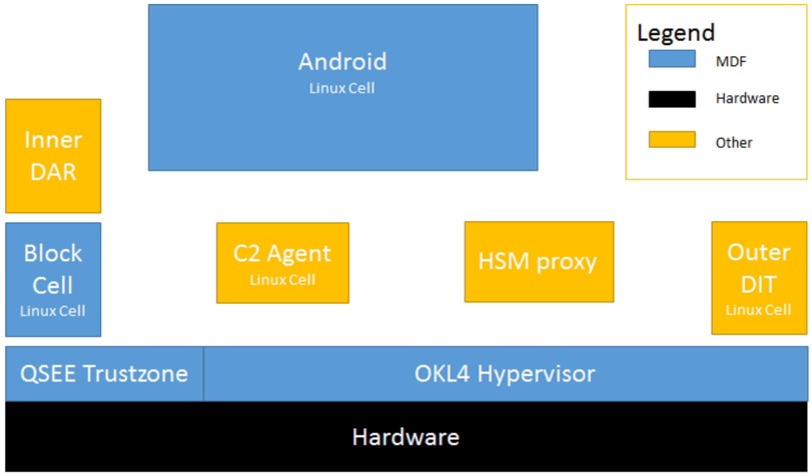

Figure 1-1 TOE Architecture

The Block Cell, is a Linux cell that provides functionality supporting the TOE security functions for Data-at-rest protections and secure key storage. The Block Cell provides users with the ability to protect Data-At-Rest (DAR) with AES encryption. This encompasses all user and mobile application data stored in the user's data partition. The DAR encryption that satisfies the Data-At-Rest protection required by the MDFPP20 is provided by the Block Cell and is referred to as the "Outer DAR" capability.

The TOE provides special protection to all user and application cryptographic keys stored in the TOE. To achieve this, the TOE includes an Android key store within the Android cell. Android and Android applications utilize the key store in the Android cell, which is the stock Qualcomm key master implementation.

The TOE includes the C2-Agent cell, the HSM Proxy cell, the Inner DAR cell and the Outer DIT cell that are not in the scope of this evaluation.

Finally, like most other mobile devices, the TOE interacts with commercial Mobile Device Management systems to allow enterprise control of the configuration and operation of the device so as to ensure adherence to enterprise-wide policies.

1.4.1.1 Physical Boundaries

The TOE's physical boundary is the physical perimeter of the mobile device enclosure.

1.4.1.2 Logical Boundaries

This section summarizes the security functions provided by D4 Secure Mobile:

- 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 multiple instances of the OpenSSL cryptographic library with CAVP validated 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 the media (including the generation and protection of data, 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 controls access to system services by hosted applications, including protection of the Trust Anchor Database. Additionally, the TOE protects user and other sensitive data using encryption so that even if a device is physically lost, the data remains protected.

1.4.1.2.3 Identification and authentication

The TOE supports a number of features related to identification and authentication. From a user perspective, except for limited functions such as making phone calls to an emergency number and receiving notifications, 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. The TOE limits the frequency of password entry and when a configured number of failures occurs, the TOE performs a full wipe of protected content. Passwords constructed using upper and lower cases characters, numbers, and special characters and up to 14 characters are supported.

The TOE serves as an 802.1X supplicant and to use X509v3 certificates and perform certificate validation for a number of functions when applicable such as 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 claimed in the corresponding Security Target (and conforming to the MDFPP requirements) as well as other functions commonly found in mobile devices. Some of the available functions are available only to the mobile device users 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 performs functions such as performing a full wipe of protected data to complete the unenrollment.

1.4.1.2.5 Protection of the TSF

The TOE implements a number of features 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 has access to a timing mechanism to ensure that reliable time information is available (e.g., for cryptographic operations and perhaps user accountability). It enforces read, write, and execute memory page protections, use address space layout randomization, and use stack-based buffer overflow protections to minimize the potential to exploit application flaws. It is 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 employs a Secure Boot process that uses cryptographic signatures to ensure the authenticity and integrity of the bootloader, and the secure boot partition produced by Cog. The cryptographic signatures utilize data fused into the device processor.

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 self-test fails, the TOE does 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.

1.4.1.2.6 TOE access

The TOE can be locked, obscuring its display, by a user or after a configured interval of inactivity. The TOE has the capability to display an advisory message (banner) when users unlock the TOE for use.

The TOE is 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, HTTPS/TLS and EAP-TLS to secure communications channels between itself and other trusted network devices.

1.4.2 TOE Documentation

- HTC One A9, Secured by D4, Administrator guide Instructions, 6 March 2017, Version 0.34

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 2.0, 17 September 2014 (MDFPP20)

2.1 Conformance Rationale

The ST conforms to the MDFPP20. As explained previously, the security problem definition, security objectives, and security requirements have been drawn from the PP.

3. Security Objectives

The Security Problem Definition may be found in the MDFPP20 and this section reproduces only the corresponding Security Objectives for operational environment for reader convenience. The MDFPP20 offers additional information about the identified security objectives, but that has not been reproduced here and the MDFPP20 should be consulted if there is interest in that material.

In general, the MDFPP20 has defined Security Objectives appropriate for mobile device and as such are applicable to the Secure Communications Mobile TOE.

3.1 Security Objectives for the Operational Environment

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

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

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

4. Extended Components Definition

All of the extended requirements in this ST have been drawn from the MDFPP20. The MDFPP20 defines the following extended requirements and since they are not redefined in this ST the MDFPP20 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_TLSC_EXT.1: Extended: EAP TLS Protocol
- FCS_TLSC_EXT.2: Extended: TLS Protocol
- FDP_ACF_EXT.1: Extended: Security access control
- FDP_DAR_EXT.1: Extended: Protected 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_PAE_EXT.1: Extended: PAE Authentication
- FIA_PMG_EXT.1: Extended: Password Management
- FIA_TRT_EXT.1: Extended: Authentication Throttling
- FIA_UAU_EXT.1: Extended: Authentication for Cryptographic Operation
- FIA_UAU_EXT.2: Extended: Timing of Authentication
- FIA_UAU_EXT.3: Extended: Re-Authentication
- FIA_X509_EXT.1: Extended: Validation of certificates
- FIA_X509_EXT.2: Extended: X509 certificate authentication
- FIA_X509_EXT.3: Extended: Request Validation of certificates

- FMT_MOF_EXT.1: Extended: Management of security functions behavior
- FMT_SMF_EXT.1: Extended: Specification of Management Functions
- FMT_SMF_EXT.2: Extended: Specification of Remediation Actions
- FPT_AEX_EXT.1: Extended: Anti-Exploitation Services (ASLR)
- FPT_AEX_EXT.2: Extended: Anti-Exploitation Services (Memory Page Permissions)
- FPT_AEX_EXT.3: Extended: Anti-Exploitation Services (Overflow Protection)
- FPT_AEX_EXT.4: Extended: Domain Isolation
- FPT_KST_EXT.1: Extended: Key Storage
- FPT_KST_EXT.2: Extended: No Key Transmission
- FPT_KST_EXT.3: Extended: No Plaintext Key Export
- FPT_NOT_EXT.1: Extended: Self-Test Notification
- FPT_TST_EXT.1: Extended: TSF Cryptographic Functionality Testing
- FPT_TST_EXT.2: Extended: TSF Integrity Testing
- FPT_TUD_EXT.1: Extended: Trusted Update: TSF version query
- FPT_TUD_EXT.2: Extended: Trusted Update Verification
- FTA_SSL_EXT.1: Extended: TSF- and User-initiated locked state
- FTA_WSE_EXT.1: Extended: Wireless Network Access
- FTP_ITC_EXT.1: Extended: Trusted channel Communication

Extended SARs:

- ALC_TSU_EXT.1: Timely Security Updates

5. Security Requirements

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

The SFRs have all been drawn from the MDFPP20. The refinements and operations already performed in the MDFPP20 are not identified (e.g., highlighted) here, rather the requirements have been copied from the MDFPP20 and any residual operations have been completed herein. Of particular note, the MDFPP20 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 MDFPP20 which includes all the SARs for EAL 1. However, the SARs are effectively refined since requirement-specific 'Assurance Activities' are defined in the MDFPP20 that serve to ensure corresponding evaluations will yield more practical and consistent assurance than the EAL 1 assurance requirements alone. The MDFPP20 should be consulted for the assurance activity definitions.

5.1 TOE Security Functional Requirements

The following table identifies the SFRs that are satisfied by Secure Communications Mobile TOE.

Table 5-1 TOE Security Functional Components

Requirement Class	Requirement Component
FCS: Cryptographic support	FCS_CKM.1(1): Cryptographic key generation
	FCS_CKM.1(2): Cryptographic key generation
	FCS_CKM.2(1): Cryptographic key establishment
	FCS_CKM.2(2): Cryptographic key distribution
	FCS_CKM_EXT.1: Extended: Cryptographic Key Support
	FCS_CKM_EXT.2: Extended: Cryptographic Key Random Generation
	FCS_CKM_EXT.3: Extended: Cryptographic Key Generation
	FCS_CKM_EXT.4: Extended: Key Destruction
	FCS_CKM_EXT.5: Extended: TSF Wipe
	FCS_CKM_EXT.6: Extended: Salt Generation
	FCS_COP.1(1): Cryptographic operation
	FCS_COP.1(2): Cryptographic operation
	FCS_COP.1(3): 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: EAP TLS Protocol
	FCS_TLSC_EXT.2: Extended: TLS Protocol
FDP: User data protection	FDP_ACF_EXT.1: Extended: Security access control
	FDP_DAR_EXT.1: Extended: Protected 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_PAE_EXT.1: Extended: PAE Authentication
	FIA_PMG_EXT.1: Extended: Password Management
	FIA_TRT_EXT.1: Extended: Authentication Throttling
	FIA_UAU.7: Protected authentication feedback
	FIA_UAU_EXT.1: Extended: Authentication for Cryptographic Operation
	FIA_UAU_EXT.2: Extended: Timing of Authentication
	FIA_UAU_EXT.3: Extended: Re-Authentication
	FIA_X509_EXT.1: Extended: Validation of certificates
	FIA_X509_EXT.2: Extended: X509 certificate authentication
	FIA_X509_EXT.3: Extended: Request Validation of certificates
FMT: Security management	FMT_MOF_EXT.1: Extended: Management of security functions behavior
	FMT_SMF_EXT.1: Extended: Specification of Management Functions
	FMT_SMF_EXT.2: Extended: Specification of Remediation Actions
FPT: Protection of the TSF	FPT_AEX_EXT.1: Extended: Anti-Exploitation Services (ASLR)
	FPT_AEX_EXT.2: Extended: Anti-Exploitation Services (Memory Page Permissions)
	FPT_AEX_EXT.3: Extended: Anti-Exploitation Services (Overflow Protection)
	FPT_AEX_EXT.4: Extended: Domain Isolation
	FPT_BBD_EXT.1: Application Processor Mediation
	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.2: Extended: TSF Integrity Testing
	FPT_TUD_EXT.1: Extended: Trusted Update: TSF version query
	FPT_TUD_EXT.2: Extended: Trusted Update Verification
FTA: TOE access	FTA_SSL_EXT.1: Extended: TSF- and User-initiated locked state
	FTA_TAB.1: TOE Access Banner
	FTA_WSE_EXT.1: Extended: Wireless Network Access
FTP: Trusted path/channels	FTP_ITC_EXT.1: Extended: Trusted channel Communication

5.1.1 Cryptographic support (FCS)

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

FCS_CKM.1(1).1

The application shall generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm [

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,

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]

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

FCS_CKM.1(2).1

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

5.1.1.3 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',

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. (TD0048 applied)

5.1.1.4 Cryptographic key distribution (FCS_CKM.2(2))

FCS_CKM.2(2).1

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

5.1.1.5 Extended: Cryptographic Key Support (FCS_CKM_EXT.1)

FCS_CKM_EXT.1.1

The TSF shall support a [***hardware-isolated***] REK with a [***symmetric***] key of strength [***256 bits***]. (TD0038 applied)

FCS_CKM_EXT.1.2

System software on the TSF shall be able only to request [***NIST SP 800-108 key derivation***] by the key and shall not be able to read, import, or export a REK. (TD0038 applied)

FCS_CKM_EXT.1.3

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

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

FCS_CKM_EXT.2.1

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

5.1.1.7 Extended: Cryptographic Key Generation (FCS_CKM_EXT.3)

FCS_CKM_EXT.3.1
The TSF shall use [***[256-bit] symmetric KEKs***] corresponding to at least the security strength of the keys encrypted by the KEK. (TD0038 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(1)) (Per TD0038),

d) Combine the KEK from other KEKs in a way that preserves the effective entropy of each factor by [encrypting one key with another] (Per TD0038)].

5.1.1.8 Extended: Key Destruction (FCS_CKM_EXT.4)

FCS_CKM_EXT.4.1
The TSF shall destroy cryptographic keys in accordance with the specified cryptographic key destruction methods:

- by clearing the KEK encrypting the target key,
- in accordance with the following rules:
  - For volatile memory, the destruction shall be executed by a single direct overwrite [***consisting of zeroes***]. (TD0028 applied)
  - 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.
  - For non-volatile flash memory that is not wear-leveled, the destruction shall be executed [***by a block erase that erases the reference to memory that stores data as well as the data itself***]. (TD0057 applied)
  - For non-volatile flash memory that is wear-leveled, the destruction shall be executed [***by a block erase***].
  - For non-volatile memory other than EEPROM and flash, the destruction shall be executed by overwriting three or more times with a random pattern that is changed before each write. (TD0047 applied)

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

5.1.1.9 Extended: TSF Wipe (FCS_CKM_EXT.5)

FCS_CKM_EXT.5.1
The TSF shall wipe all protected data by [

- ***Cryptographically erasing the encrypted DEKs and/or the KEKs in non-volatile memory by following the requirements in FCS_CKM_EXT.4.1;***].

FCS_CKM_EXT.5.2
The TSF shall perform a power cycle on conclusion of the wipe procedure.

5.1.1.10 Extended: Salt Generation (FCS_CKM_EXT.6)

FCS_CKM_EXT.6.1
The TSF shall generate all salts using a RBG that meets FCS_RBG_EXT.1.

5.1.1.11 Cryptographic operation (FCS_COP.1(1))

FCS_COP.1(1).1

The TSF shall perform [encryption/decryption] in accordance with a specified cryptographic algorithm

- AES-CBC (as defined in 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-CCM (as defined in NIST SP 800-38C),***
- ***AES-XTS (as defined in NIST SP 800-38E) mode***]

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

5.1.1.12 Cryptographic operation (FCS_COP.1(2))

FCS_COP.1(2).1

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

5.1.1.13 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];***].

5.1.1.14 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***] and message digest sizes 160 and [***256, 384, 512***] bits that meet the following: [FIPS Pub 198-1, 'The Keyed-Hash Message Authentication Code', and FIPS Pub 180-4, 'Secure Hash Standard'].

5.1.1.15 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 [***10000***] iterations, and output cryptographic key sizes [***256***] that meet the following: [NIST SP 800-132].

5.1.1.16 Extended: 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.2).

FCS_HTTPS_EXT.1.3

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

5.1.1.17 Extended: Initialization Vector Generation (FCS_IV_EXT.1)

FCS_IV_EXT.1.1
The TSF shall generate IVs in accordance with **MDFPP20** Table 14: References and IV Requirements for NIST-approved Cipher Modes.

5.1.1.18 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 [***CTR_DRBG (AES)***]. (TD0079 applied)

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.

5.1.1.19 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 [***(per TD0059) selected algorithms***] in FCS_CKM.2(1),
- 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) [
- ***All mandatory and [(per TD0059) selected algorithms] in FCS_CKM.1(1),***
- ***The selected algorithms in FCS_COP.1(5)***].

5.1.1.20 Extended: Cryptographic Key Storage (FCS_STG_EXT.1)

FCS_STG_EXT.1.1
The TSF shall provide [***hardware-isolated***] secure key storage for asymmetric private keys and [***symmetric keys***].

FCS_STG_EXT.1.2
The TSF shall be capable of importing keys/secrets into the secure key storage upon request of [***the user***] 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***].

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

FCS_STG_EXT.2.1

The TSF shall encrypt all DEKs and KEKs and [***long-term trusted channel key material***] by KEKs that are [

1) Protected by the REK with [

b. encryption by a KEK chaining to a REK],

2) Protected by the REK and the password with [

b. encryption by a KEK chaining to a REK and the password-derived KEK]].

FCS_STG_EXT.2.2

DEKs and KEKs and [***long-term trusted channel key material***] shall be encrypted using one of the following methods: [

using a SP800-56B key establishment scheme,

using AES in the [GCM, CBC mode]]. (TD0038 applied)

5.1.1.22 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 [

- ***[GCM] cipher mode for encryption according to FCS_STG_EXT.2***
- ***a keyed hash (FCS_COP.1(4)) using a key protected by a key protected by FCS_STG_EXT.2***].

FCS_STG_EXT.3.2

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

5.1.1.23 Extended: EAP TLS Protocol (FCS_TLSC_EXT.1)

FCS_TLSC_EXT.1.1

The TSF shall implement TLS 1.0 and [***TLS 1.1 (RFC 4346)***] 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_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 server certificate presented for EAP-TLS [***chains to one of the specified CAs***].

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.

5.1.1.24 Extended: TLS Protocol (FCS_TLSC_EXT.2)

FCS_TLSC_EXT.2.1
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_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.2.2
The TSF shall verify that the presented identifier matches the reference identifier according to RFC 6125.

FCS_TLSC_EXT.2.3
The TSF shall not establish a trusted channel if the peer certificate is invalid.

FCS_TLSC_EXT.2.4
The TSF shall support mutual authentication using X.509v3 certificates.

FCS_TLSC_EXT.2.5
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.

ST Author Note: FCS_TLSC_EXT.2.6 is no longer required to meet Use Case #2 ("Enterprise-owned device for specialized, high-security use"), as detailed in section G.2 of the Protection Profile for Mobile Devices v2.0. [TD0091]

5.1.2 User data protection (FDP)

5.1.2.1 Extended: Security access control (FDP_ACF_EXT.1)

FDP_ACF_EXT.1.1
The TSF shall provide a mechanism to restrict the system services that are accessible to an application.

FDP_ACF_EXT.1.2
The TSF shall provide an access control policy that prevents [***groups of application processes***] from accessing [***all***] data stored by other [***groups of application processes***]. Exceptions may only be explicitly authorized for such sharing by [***a common application developer***].

5.1.2.2 Extended: Protected Data Encryption (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***] mode with key size [***128***] bits.

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

FDP_IFC_EXT.1.1
The TSF shall [***provide an interface to VPN clients to enable all IP traffic (other than IP traffic required to establish the VPN connection) to flow through the IPsec VPN client***].

5.1.2.4 Extended: User Data Storage (FDP_STG_EXT.1)

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

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

FDP_UPC_EXT.1.1
The TSF provide a means for non-TSF applications executing on the TOE to use TLS, HTTPS, Bluetooth BR/EDR, and [***no other protocol***] 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.

5.1.3 Identification and authentication (FIA)

5.1.3.1 Authentication failure handling (FIA_AFL_EXT.1)

FIA_AFL_EXT.1.1
The TSF shall detect when a configurable positive integer within [***1 and 10***] of unsuccessful authentication attempts occur related to last successful authentication by that user.

FIA_AFL_EXT.1.2
When the defined number of unsuccessful authentication attempts has been surpassed, the TSF shall perform wipe of all protected data.

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

5.1.3.2 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.

5.1.3.3 Extended: PAE Authentication (FIA_PAE_EXT.1)

FIA_PAE_EXT.1.1
The TSF shall conform to IEEE Standard 802.1X for a Port Access Entity (PAE) in the 'Supplicant' role.

5.1.3.4 Extended: Password Management (FIA_PMG_EXT.1)

FIA_PMG_EXT.1.1
The TSF shall support the following for the Password Authentication Factor: 1. Passwords shall be able to be composed of any combination of [***upper and lower case letters***], numbers, and special characters: [***[“!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, “)”, “+”, “=”, “_”, “/”, “-“, “’”, “”",***]

":", ";", ",", "?", "`", "~", "\", "|", "<", ">", "{", "}", "[", "]"]; 2. Password length up to [***16***] characters shall be supported.

5.1.3.5 Extended: Authentication Throttling (FIA_TRT_EXT.1)

FIA_TRT_EXT.1.1

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

5.1.3.6 Protected authentication feedback (FIA_UAU.7)

FIA_UAU.7.1

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

5.1.3.7 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 [***long-term trusted channel key material***] at startup.

5.1.3.8 Extended: Timing of Authentication (FIA_UAU_EXT.2)

FIA_UAU_EXT.2.1

The TSF shall allow [

- ***enter password to unlock,***
- ***turn off or Restart the TOE,***
- ***see the configured banner,***
- ***toggle the Do Not Disturb and Auto Rotate function,***
- ***make/receive an emergency call,***
- ***receive an incoming phone calls,***
- ***take screen shots,***
- ***enable or disable airplane mode,***
- ***configure sound/vibrate/mute,***
- ***use a flashlight,***
- ***use Voice assist ,***
- ***access widget functions, and***
- ***view notifications***

] 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.

Application Note: Some notifications identify actions, for example to view a screenshot; however, selecting those notifications highlights the password prompt and require the password to access that data.

5.1.3.9 Extended: Re-Authentication (FIA_UAU_EXT.3)

FIA_UAU_EXT.3.1

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

5.1.3.10 Extended: Validation of certificates (FIA_X509_EXT.1)

FIA_X509_EXT.1.1

The TSF shall validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a certificate in the Trust Anchor Database.
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates.
- The TSF shall validate the revocation status of the certificate using [***a Certificate Revocation List (CRL) as specified in RFC 5759***].
- The TSF shall validate the extendedKeyUsage field according to the following rules:
  - 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.
  - 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.
  - (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.

5.1.3.11 Extended: X509 certificate authentication (FIA_X509_EXT.2)

FIA_X509_EXT.2.1

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

FIA_X509_EXT.2.2

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

Application Note: The TOE EAP-TLS implementation accepts the certificate when the revocation server cannot be reached, and the TOE TLS and HTTPS implementations reject the certificate when the revocation server cannot be reached.

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

FIA_X509_EXT.3.1

The TSF shall provide a certificate validation service to applications.

FIA_X509_EXT.3.2

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

5.1.4 Security management (FMT)

5.1.4.1 Extended: 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 1~~ Table 5-2 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 1~~ Table 5-2 Security Management Functions to the administrator when the device is enrolled and according to the administrator-configured policy.

5.1.4.2 Extended: Specification of Management Functions (FMT_SMF_EXT.1)

FMT_SMF_EXT.1.1

The TSF shall be capable of performing the ~~following management~~ functions~~:~~ **in column 2 of** Table 5-2 Security Management Functions.

FMT_SMF_EXT.1.2

The TSF shall be capable of allowing the administrator to perform the functions in column 4 of Table 5-2 Security Management Functions**.**

Table 5-2 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, complexity, and lifetime) using the Android MDM APIs.	M	-	M	M
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. The user can also adjust each of the session locking policies; however, if set by the administrator, the user can only set a more strict policy (e.g., setting the device to allow fewer authentication failures than configured by the administrator).	M	-	M	M
3. enable/disable the VPN protection: a. across device c. [***no other method***]	M	-	I	I
4. enable/disable [***Wifi, Bluetooth, and Cellular radios***]	M	-	-	-
5. enable/disable [camera, microphone]: (TD0044 applied) a. across device c. [***no other method***]	M	-	~~M~~ ~~I~~	~~M~~ I
6. specify wireless networks (SSIDs) to which the TSF may connect (TD0064 applied)	~~M~~ I	-	~~M~~ I	-
7. configure security policy for each wireless network: a. [***specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s)***] b. security type c. authentication protocol d. client credentials to be used for authentication	M	-	M	-
8. transition to the locked state	M	-	M	-
9. full wipe of protected data	M	-	M	-
10. configure application installation policy by [	M	-	M	M

a. specifying authorized application repository(s), ***b. denying installation of applications***]				
11. import keys/secrets into the secure key storage	M	I	-	-
12. destroy imported keys/secrets and [***no other keys/secrets***] in the secure key storage	M	I	-	-
13. import X.509v3 certificates into the Trust Anchor Database	M	-	M	-
14. remove imported X.509v3 certificates and [**all other X.509v3 certificates**] in the Trust Anchor Database	M	-	-	-
15. enroll the TOE in management (TD0058 applied)	M	M	~~O~~ -	-
16. remove applications	M	-	M	-
17. update system software	M	-	M	-
18. install applications	M	-	M	-
19. remove Enterprise applications	M	-	M	-
20. configure the Bluetooth trusted channel: a. disable/enable the Discoverable mode (for BR/EDR) b. change the Bluetooth device name c. [***no other Bluetooth configuration***]	M	-	-	-
21. enable/disable display notification in the locked state of: [***all notifications***]	M	-	I	I
22. enable/disable all data signaling over [assignment: list of externally accessible hardware ports]	-	-	-	-
23. enable/disable [assignment: *list of protocols where the device acts as a server* = Protocols supporting wireless remote access = all protocols where the device acts as a server]	I	-	-	-
24. enable/disable developer modes	-	-	-	-
25. enable data-at rest protection	-	-	-	-
26. enable removable media's data-at-rest protection	-	-	-	-
27. enable/disable bypass of local user authentication	-	-	-	-
28. wipe Enterprise data	-	-	-	-
29. approve [selection: *import*, *removal*] by applications of X.509v3 certificates in the Trust Anchor Database	-	-	-	-
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	-	-	-	-
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	I	-	-	-
37. configure the auditable items	-	-	-	-
38. retrieve TSF-software integrity verification values	-	-	-	-
39. enable/disable [selection: a. *USB mass storage mode,* b. *USB data transfer without user authentication,* c. *USB data transfer without authentication of the connecting system*]	-	-	-	-
40. enable/disable backup to [selection: *locally connected system, remote system*]	-	-	-	-
41. enable/disable [selection: a. *Hotspot functionality authenticated by [selection: pre-shared key, ~~passcode, no authentication~~],* b. *USB tethering authenticated by [selection: pre-shared key, passcode, ~~no authentication~~]*]	-	-	-	-
42. approve exceptions for sharing data between [selection: *application processes, groups of application processes*]	-	-	-	-

43. place applications into application process groups based on [assignment: *application characteristics*]	-	-	-	-
44. enable/disable location services: a. across device b. [***no other method***]	M	-	-	-
45. [assignment: *list of other management functions to be provided by the TSF*]	-	-	-	-

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

FMT_SMF_EXT.2.1
The TSF shall offer [***alert the administrator and performs a factory reset of the device***] upon unenrollment and [***no other triggers***].

5.1.5 Protection of the TSF (FPT)

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

FPT_AEX_EXT.1.1
The TSF shall provide address space layout randomization (ASLR) to applications.

FPT_AEX_EXT.1.2
The base address of any user-space memory mapping will consist of at least 8 unpredictable bits.

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

FPT_AEX_EXT.2.1
The TSF shall be able to enforce read, write, and execute permissions on every page of physical memory.

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

FPT_AEX_EXT.3.1
TSF processes that execute in a non-privileged execution domain on the application processor shall implement stack-based buffer overflow protection.

5.1.5.4 Extended: Domain Isolation (FPT_AEX_EXT.4)

FPT_AEX_EXT.4.1
The TSF shall protect itself from modification by untrusted subjects.

FPT_AEX_EXT.4.2
The TSF shall enforce isolation of address space between applications.

5.1.5.5 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.

5.1.5.6 Extended: Key Storage (FPT_KST_EXT.1)

FPT_KST_EXT.1.1
The TSF shall not store any plaintext key material in readable non-volatile memory.

5.1.5.7 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.

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

FPT_KST_EXT.3.1
The TSF shall ensure it is not possible for the TOE user(s) to export plaintext keys.

5.1.5.9 Extended: Self-Test Notification (FPT_NOT_EXT.1)

FPT_NOT_EXT.1.1
The TSF shall transition to non-operational mode and [***no other actions***] when the following types of failures occur:

- failures of the self-test(s)
- TSF software integrity verification failures
- [***no other failures***].

5.1.5.10 Reliable time stamps (FPT_STM.1)

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

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

FPT_TST_EXT.1.1
The TSF shall run a suite of self-tests during initial start-up (on power on) to demonstrate the correct operation of all cryptographic functionality.

5.1.5.12 Extended: TSF Integrity Testing (FPT_TST_EXT.2)

FPT_TST_EXT.2.1
The TSF shall verify the integrity of the bootchain up through the Application Processor OS kernel, and [***all executable code stored in mutable media***], stored in mutable media prior to its execution through the use of [***a digital signature using a hardware-protected asymmetric key***].

5.1.5.13 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 the device.

FPT_TUD_EXT.1.3
The TSF shall provide authorized users the ability to query the current version of installed mobile applications.

5.1.5.14 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***].

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

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

5.1.6 TOE access (FTA)

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

FTA_SSL_EXT.1.1
The TSF shall transition to a locked state after a time interval of inactivity.

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) [***no other actions***].

5.1.6.2 Default TOE Access Banners (FTA_TAB.1)

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

5.1.6.3 Extended: Wireless Network Access (FTA_WSE_EXT.1)

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

5.1.7 Trusted path/channels (FTP)

5.1.7.1 Extended: Trusted channel Communication (FTP_ITC_EXT.1)

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

5.2 TOE Security Assurance Requirements

The SARs for the TOE are the 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.

Table 5-3 Assurance Components

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

5.2.1 Development (ADV)

5.2.1.1 Basic functional specification (ADV_FSP.1)

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

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

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

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

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

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

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

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

5.2.2 Guidance documents (AGD)

5.2.2.1 Operational user guidance (AGD_OPE.1)

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

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

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

AGD_OPE.1.3c
The operational user guidance shall describe, for each user role, the available functions and

interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.

AGD_OPE.1.4c

The operational user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_OPE.1.5c

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

AGD_OPE.1.6c

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

AGD_OPE.1.7c

The operational user guidance shall be clear and reasonable.

AGD_OPE.1.1e

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

5.2.2.2 Preparative procedures (AGD_PRE.1)

AGD_PRE.1.1d

The developer shall provide the TOE including its preparative procedures.

AGD_PRE.1.1c

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

AGD_PRE.1.2c

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

AGD_PRE.1.1e

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

AGD_PRE.1.2e

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

5.2.3 Life-cycle support (ALC)

5.2.3.1 Labelling of the TOE (ALC_CMC.1)

ALC_CMC.1.1d

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

ALC_CMC.1.1c

The TOE shall be labelled with its unique reference.

ALC_CMC.1.1e

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

5.2.3.2 TOE CM coverage (ALC_CMS.1)

ALC_CMS.1.1d

The developer shall provide a configuration list for the TOE.

ALC_CMS.1.1c
The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.
ALC_CMS.1.2c
The configuration list shall uniquely identify the configuration items.
ALC_CMS.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3.3 Timely Security Updates (ALC_TSU_EXT.1)

ALC_TSU_EXT.1.1d
The developer shall provide a description in the TSS of how timely security updates are made to the TOE.
ALC_TSU_EXT.1.1c
The description shall include the process for creating and deploying security updates for the TOE software/firmware.
ALC_TSU_EXT.1.2c
The description shall express the time window as the length of time, in days, between public disclosure of a vulnerability and the public availability of security updates to the TOE.
ALC_TSU_EXT.1.3c
The description shall include the mechanisms publicly available for reporting security issues pertaining to the TOE.
ALC_TSU_EXT.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4 Tests (ATE)

5.2.4.1 Independent testing - conformance (ATE_IND.1)

ATE_IND.1.1d
The developer shall provide the TOE for testing.
ATE_IND.1.1c
The TOE shall be suitable for testing.
ATE_IND.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ATE_IND.1.2e
The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

5.2.5 Vulnerability assessment (AVA)

5.2.5.1 Vulnerability survey (AVA_VAN.1)

AVA_VAN.1.1d
The developer shall provide the TOE for testing.
AVA_VAN.1.1c
The TOE shall be suitable for testing.
AVA_VAN.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
AVA_VAN.1.2e
The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.

AVA_VAN.1.3e

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

6. TOE Summary Specification

This chapter describes the security functions:

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

6.1 Cryptographic support

The TOE includes cryptographic libraries in each of three OKL4 cells. Each provides cryptographic functions in support of various security features offered by the given cell. The TOE implements cryptographic algorithms in accordance with the following NIST standards and has received CAVP algorithm certificates as shown in the following table.

The TOE includes the OpenSSL library version 1.0.2j, using FIPS OpenSSL version 2.0.14, running as different instances in the Block Cell OKL4 cell, and the Android OKL4 cell.

Table 6-1 Cryptographic Providers and CAVP Certificates

Algorithm	NIST Standard	SFR Reference	Provider	Cert#
AES 128/256 bit CBC, CCM, and GCM modes	FIPS 197, SP 800-38A/C/E	FCS_COP.1(1)	OpenSSL	AES: #4476
AES Key Wrap	SP 800-38F IEEE 802.11-2012	FCS_CKM.2(2)	OpenSSL	AES: #4476
SHS SHA-1/256/384/512	FIPS 180-4	FCS_COP.1(2)	OpenSSL	SHA: #3686
RSA SIG(gen), SIG(ver), 2048-bits	FIPS 186-4	FCS_CKM.2(1) FCS_COP.1(3)	OpenSSL	RSA: #2446
ECDSA PKG, PKV, SigGen, SigVer, P-256, P-384, P-521	FIPS 186-4	FCS_CKM.1(1) FCS_CKM.2(1) FCS_COP.1(3)	OpenSSL	ECDSA: #1093
DSA Sig(gen), SIG(ver), Key(GEN)	FIPS 186-4	FCS_CKM.1(1)	OpenSSL	DSA: #1200
HMAC SHA-1 HMAC SHA-256/384/512	FIPS 198-1 & 180-4	FCS_COP.1(4)	OpenSSL	HMAC: #2969
PBKDF2	NIST SP 800-132 **No NIST CAVP test available**	FCS_COP.1(5)	OpenSSL	No Cert Possible
DRBG	SP 800-90A	FCS_RBG_EXT.1	OpenSSL	DRBG: #1456
CVL FCC	SP 800-56A	FCS_CKM.2(1)	OpenSSL	CVL: #1191

Algorithm	NIST Standard	SFR Reference	Provider	Cert#
ECC				

The TOE derives all DEKs and KEKs as defined by the following table. The TOE uses a NIST 800-108 (CMAC) key derivation function to derive keys used for the DAR functionality and the key store functionality which chain from the Root Encryption Key (REK).

Table 6-2 D4 Secure CSP Information

Key Name	Key Type	Usage	Algorithm / Size	Generated From / Derived from
ICE_RDK[1]	KEK	Encrypt / decrypt Intermediate value for FDE_DKE	256-bit AES CBC	Generated from SHK using SP 800-108 (CMAC)
KeyMaster RDK	KEK	Encrypt / decrypt private keys and symmetric keys	256-bit AES CBC	Generated from SHK using SP 800-108 (CMAC)
FDE_DEK	DEK	Encrypt / Decrypt user data (FDE_DEK)	AES-128 XTS	Generated using its approved DRBG
WPA2 keys	Ephemeral session key	Encrypt/decryption Wi-Fi/802.11 communications	nonces	Generated from OpenSSL DRBG in Android Cell
TLS/HTTPS session keys	Ephemeral session key	Encryption/decryption of TLS/HTTPS sessions	128-bit / 256-bit AES CBC	OpenSSL – Derived as specified in TLS/HTTPS protocol
Bluetooth BDR	Ephemeral session key	Encryption/decryption of Bluetooth communications	128-bit AES	Generated within Bluetooth

The TOE provides an SP800-90A AES-256 CTR DRBG provided by FIPS OpenSSL library. Three instances of this DRBG are present, one each in the various Linux / Android OKL4 cells requiring random data. User applications can obtain random data from the AES-256 CTR_DRBG running in the Android OKL4 Cell.

The TOE initializes the RBG with sufficient entropy ultimately accumulated from a TOE-hardware-based noise source (detailed information was provided to NIAP). The hardware based DRBG is seeded from the conditioned, hardware noise source. The hardware-based DRBG is fed into the Linux kernels entropy pool which is reflected through /dev/random. The FIPS OpenSSL library is seeded from /dev/random.

These sources for RBG all provide a security strength of at least 256-bits.

The Cryptographic support function is designed to satisfy the following security functional requirements:

- FCS_CKM.1(1): The TOE provides asymmetric key generation for DH, DHE/DSA and ECDH/ECDSA. The TOE generates DH/ECDH, and ECDSA (including P-256, P384 and P-521) keys in its OpenSSL software library (which generates the keys in accordance with FIPS 186-4). The TOE supports generating keys with a security strength of 112-bits and larger, thus supports 2048-bit DH keys, and 256-bit ECDH keys. The TOE utilizes generated DH/ECDH when acting as a TLS Client and the TOE provides mobile applications HTTP/TLS APIs as well as general cryptographic APIs (to generate ECDSA key pairs).

[1] RDK is a root derived key. The ICE_RDK is a key derived from the REK, and used to protect the ICE-based data-at-rest protections.

- FCS_CKM.1(2): The TOE adheres to 802.11-2012 for key generation. The TOE's wpa_supplicant provides the PRF384 for WPA2 derivation of 128-bit AES Temporal Key and also employs its OpenSSL AES-256 DRBG when generating random values use in the EAP-TLS and 802.11 4-way handshake. The TOE has been designed for compliance, and the Device has successfully completed certification (including WPA2 Personal) and received a Wi-Fi CERTIFIED Interoperability Certificate from the Wi-Fi Alliance (Certification ID: Cert # WFA61756). The Wi-Fi Alliance maintains a website providing further information about the testing program: http://www.wi-fi.org/certification.
- FCS_CKM.2(1): The TOE supports RSA (800-56B), DHE (FFC 800-56A), and ECDHE (ECC 800-56A) methods in TLS key establishment/exchange (the sole secure channel the TOE provides). The user and administrator need take no special configuration of the TOE as the TOE automatically generates the keys needed for negotiated TLS ciphersuite. Because the TOE only acts as a TLS client, the TOE only performs 800-56B encryption (specifically the encryption of the Pre-Master Secret using the Server's RSA public key) when participating in TLS_RSA_* based TLS handshakes. Thus, the TOE does not perform 800-56B decryption during TLS handshakes. However, the TOE's TLS client correctly handles other cryptographic errors (for example, invalid checksums, incorrect certificate types, corrupted certificates) by sending a TLS fatal alert.
- FCS_CKM.2(2): The TOE adheres to RFC 3394, SP 800-38F, and 802.11-2012 standards and unwraps the GTK (sent encrypted with the WPA2 KEK using AES Key Wrap in an EAPOL-Key frame). The TOE, upon receiving an EAPOL frame, will subject the frame to a number of checks (frame length, EAPOL version, frame payload size, EAPOL-Key type, key data length, EAPOL-Key CCMP descriptor version, and replay counter) to ensure a proper EAPOL message and then decrypt the GTK using the KEK, thus ensuring that it does not expose the Group Temporal Key (GTK).
- FCS_CKM_EXT.1: The TOE includes a hardware-based, 256-bit Secondary Hardware Key (SHK) that is provided by fuses on the processor. The SHK is Qualcomm terminology. Common Criteria requirements refer to this as a Root Encryption Key (REK). The fuses making up the SHK are written once when the device is first initialized using data from the hardware DRBG provided by the processor. The REK is used to derive keys (w/ SP 800-108 CMAC) for Data-At-Rest (DAR) protection, AuthToken HMAC validation and the KeyMaster keystore. The REK is never entered, read, exported, or updated after it is initialized. The only operations allowed by or performed on the REK is key derivation using SP 800-108 (CMAC).
- FCS_CKM_EXT.2: The TOE generates all DEKs as described by Table 6-2 D4 Secure CSP Information.
- FCS_CKM_EXT.3: The TOE derives all DEKs and KEKs as defined by Table 6-2 D4 Secure CSP Information.
- FCS_CKM_EXT.4: The TOE destroys cryptographic material (plaintext keys, authentication data, other security parameters) when they are no longer in use by the system. No plaintext cryptographic keying material resides in the TOE's flash, because 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 FDE_DEK that is used to encrypt the user data partition. Because the TOE's keystore stores encrypted keys within the user data partition, TOE effectively cryptographically erases those keys when clearing the Data-At-Rest DEK (FDE_DEK). In turn, the TOE clears the FDE_DEK FEKs through a secure direct overwrite (BLKSECDISCARD ioctl) of the Flash memory containing the key followed by a read-verify. The TOE uses Flash memory with wear-leveling.
- FCS_CKM_EXT.5: The TOE stores all protected data in encrypted form within the user data partition. Upon request, the TOE cryptographically erases the FDE_DEK that is protecting the user data partition, clears that key from memory, reformats the partition, and then reboots. The TOE's clearing of the FDE_DEK follows the requirements of FCS_CKM_EXT.4.
- FCS_CKM_EXT.6: All salt value used by Android or Linux OKL4 cells are generated from an OpenSSL DRBG.
- FCS_COP.1(1): The TOE uses multiple cryptographic providers to perform encryption and decryption using AES in accordance with the standards, key sizes and modes referenced in Table 6-1 Cryptographic Providers and CAVP Certificates.

- FCS_COP.1(2): The TOE uses multiple cryptographic providers to perform cryptographic hashing using SHA algorithms in accordance with the standards and message digest sizes referenced in Table 6-1 Cryptographic Providers and CAVP Certificates.
- FCS_COP.1(3): The TOE uses multiple cryptographic providers to perform cryptographic signature generation and verification using either an ECDSA or RSA scheme that is in accordance with standards and key-sizes referenced in Table 6-1 Cryptographic Providers and CAVP Certificates.
- FCS_COP.1(4): The TOE uses multiple cryptographic providers to perform keyed-hash message authentication using the HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384 and HMAC-SHA-512 algorithms, meeting the standards referenced in the tables above. These include key sizes and message digest sizes of 160, 256, 384 and 512. Refer to Table 6-1 for associated CAVP certificate for these algorithms.
- FCS_COP.1(5): The TOE conditions a user's password per SP 800-132 using PBKDF2 using HMAC-SHA-256 with 10000 iterations, to combine a 128-bit salt with the user's password to obtain a 256-bit master key which is provided to the Qualcomm Integrated Cryptographic Engine (ICE) for use by the full disk encryption capability. The TOE utilizes several mechanism to increase the computational complexity of the protections afforded by keys derived from passwords. First, the TOE performs 10000 HMAC-SHA-256 iterations during PBKDF2 key derivation. The TOE also combines the password derived key with a KEK chained to the REK. Finally, the TOE enforces a maximum number of incorrect login attempts prior to wiping all user data.

  The time needed to derive keying material does not impact or lessen the difficulty faced by an attacker's exhaustive guessing matter as the combination of the password derived KEK with REK value entirely prevents offline attacks. The TOE's maximum incorrect login attempts (less than 99), password length (between 4 and 16 characters) and password complexity rules prevent exhaustive online attacks because the number of combinations of passwords that an adversary can attempt is much higher than the maximum incorrect login attempts required before a device wipe is triggered.
- FCS_HTTPS_EXT.1: The TOE supports the HTTPS protocol compliant with RFC 2818. Applications on the TOE can act as an HTTPS client to connect to external servers using HTTPS over TLS (which is compliant with FCS_TLSC_EXT.2). The TOE also offers TLS APIs that ensure the certificate checking performed by the TOE as part of an HTTPS client's communications will satisfy FIA_X509_EXT not establish the connection to the targeted server.
- FCS_IV_EXT.1: The TOE generates IVs for data storage encryption and for key storage encryption. The TOE uses AES-CBC mode for data encryption.
- FCS_RBG_EXT.1: The TOE provides the DRBGs identified above having the CAVP certificates shown in Table 6-1 Cryptographic Algorithms, Providers and CAVP certificates. The TOE initializes each RBG with sufficient entropy ultimately accumulated from a TOE-hardware-based noise source (detailed information was provided to NIAP). Once initialized the DRBGs provide a security strength of 256-bits.
- FCS_SRV_EXT.1: The TOE provides applications access to the cryptographic operations including encryption (AES-CBC), hashing (SHA), signing and verification (RSA), 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). The TOE provides access through the Android operating system's Java API, through the native OpenSSL API, and through the kernel.
- FCS_STG_EXT.1: The TOE provides the Android Key Store that is used by the Android Cell, and is accessible to Android applications. The Android key stores is a hardware isolated protected keystores.

  The Android Key Store is provided to the user and mobile applications. This keystore can generate, import and securely store symmetric and asymmetric keys. Keys can also be destroyed. While normally mobile applications cannot use or destroy the keys of another application, applications that share a common application developer (and are thus signed by the same developer key) may do so. In other words applications with a common developer may use and destroy each other's keys located within the Secure

Key Storage. The TOE utilizes a TrustZone implementation of KeyMaster key storage service to provide hardware-isolated secure key storage, for symmetric keys and asymmetric private keys.

- FCS_STG_EXT.2: The REK derived keys used in the key hierarchy for Outer DAR protections and Android's KeyMaster are never stored in non-volatile memory.

  The Outer DAR DEK (a.k.a., FDE_DEK) is stored in non-volatile memory only when it is protected by a KEK derived from the REK and by the user's password. Before being stored the FDE_DEK is encrypted by a KEK derived from the user's password using AES-256 in GCM mode, and the result is encrypted by a KEK derived from the REK using AES-256 in GCM mode. The encrypted key and IV is then stored in the crypto footer of the partition.

  The Keymaster_RDK is used to provide an integrity hash of asymmetric private keys and symmetric keys belonging to the user through the use of HMAC-SHA-256. These keys are protected by encrypting them with the SHK using AES-256-CBC.

  Wi-fi keys are protected through the use of 802.11-2012 Key Confirmation Key (KCK) and Key Encryption Key (KEK) keys. These keys unwrap the WPA2 GTK (Group Temporal Key) send by an access point. Persistent wi-fi keys such as pre-shared keys or certificates are protected by storing them on an encrypted user data partition which is protected by a KEK chained to both a user's password and the device REK.

- FCS_STG_EXT.3: The integrity of the FDE_DEK is provided through the use of AES-256 in GCM mode when the key is encrypted and decrypted. Stored asymmetric private keys and symmetric keys have integrity protections through the use of HMAC-SHA-256 (using the KeyMaster RDK).

- FCS_TLSC_EXT.1: The TSF supports TLSv1.1, and TLSv1.0 using the following pre-configured ciphersuites. These are used with EAP-TLS as part of WPA2.

  - TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246
  - 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_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,

  The TOE supports mutual authentication by providing a certificate in response to a server's certificate request message received during TLS negotiation. The TOE verifies X.509v3 certificates as per FIA_X509_EXT.1, FIA_X509_EXT.2 and FIA_X509_EXT.3. When a certificate presented by a server during TLS negotiation is deemed invalid, the TOE rejects the TLS channel negotiation (i.e., no connection is established). The TOE supports authentication using only elliptic curves P-256, P-384 and P-521 within EAP/TLS negotiation (corresponding to secp256r1, secp384r1, and secp521r1).

- FCS_TLSC_EXT.2: The TOE provides mobile applications (through its Android API) the use of TLS versions 1.2 including support for the following pre-configured cipher suites.

  - TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246
  - 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_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,

The TOE supports mutual authentication by providing a X.509v3 certificate in response to a server's certificate request message received during TLS negotiation. The TOE verifies X.509v3 certificates as per FIA_X509_EXT.1, FIA_X509_EXT.2 and FIA_X509_EXT.3. In addition to this verification, the TOE implements identity verification that is consistent with RFC 6125. This includes checks of the Subject Alternative Name fields, checks of the Common Name field, and check for acceptable use of wildcards in names. When a certificate presented by a server during TLS negotiation is deemed invalid, the TOE rejects the TLS channel negotiation (i.e., no connection is established). The TOE supports authentication using only elliptic curves P-256, P-384 and P-521 within TLS negotiation (corresponding to secp256r1, secp384r1, and secp521r1).

6.2 User data protection

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. 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.

a) privileged and signature- The privileged permission is granted to any applications installed as a privileged app on the system image. The signature protection level is sufficient for most needs and works regardless of exactly where applications are installed. These permission flags are 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.

b) system - Old synonym for 'privileged'.

c) development - this permission can also (optionally) be granted to development applications (e.g., to allow additional location reporting during beta testing).

d) appop - this permission is closely associated with an app op for controlling access.

e) pre23 - this permission can be automatically granted to apps that target API levels below API level 23 (Marshmallow/6.0).

f) installer - this permission can be automatically granted to system apps that install packages.

g) verifier- this permission can be automatically granted to system apps that verify packages.

h) 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).

Refer to Table 6-3 Categories of System Services for specific permissions associated with protection levels applicable to the TOE.

Table 6-3 Categories of System Services

Permission Table.docx

The User data protection function is designed to satisfy the following security functional requirements:

- FDP_ACF_EXT.1: The mechanism to restrict system services that are accessible to an application are shown in the text immediately above.
- FDP_DAR_EXT.1: The TOE utilizes the Qualcomm Integrated Cryptographic Engine (ICE) to provide AES-128 XTS mode 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 Android Key Store. The TOE separately encrypts those TSF keys and data. Additionally, the TOE includes a read-only file system in which the TOE's system executables, libraries, and their configuration data reside. For its Data-At-Rest encryption of the user data partition on the internal Flash

(where the TOE stores all user data and all application data), the TOE uses the AES-128 bit DEK in XTS mode to encrypt the entire partition. The SD Card reader is not accessible to the Android cell.

- FDP_IFC_EXT.1: The D4 Secure provides a 3rd party VPN client within the Android cell with an API to ensure IP routes are configured to direct IP traffic through a 3rd party VPN client. The API ensures that all traffic is directed through the VPN client when a VPN client connection exists. The API configures kernel packet filtering rules to ensure the TOE transmits all outbound traffic through the VPN Client and the TOE accepts inbound traffic only if that traffic is part of the established VPN connection. All traffic that is not part of the VPN connection is discarded by the TOE.
- FDP_STG_EXT.1: The TOE's Android Cell provides a Trusted Anchor Database that consists of built-in certificates and any additional user or admin/MDM loaded certificates. The built-in certs along with user added certs are stored in user specific directories within the user data partition. Neither the Block Cell, OKL4 hypervisor nor QSEE Trustzone access certificates in the Trust Anchor Database.
- FDP_UPC_EXT.1: The TOE provides APIs allowing non-TSF applications (mobile applications) the ability to establish a secure channel using TLS, HTTPS, Bluetooth DR/EDR and Bluetooth BR/EDR. 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

6.3 Identification and authentication

The Identification and authentication function is designed to satisfy the following security functional requirements:

- FIA_AFL_EXT.1: The TOE maintains, for each user, the number of failed logins since the last successful login, and upon reaching the maximum number of incorrect logins, the TOE performs a full wipe of all protected data (and in fact, wipes all user data). An administrator can adjust the number of failed logins from the default of ten failed logins to a value between one and one hundred using an MDM. Turning power to the device off, does not affect the count of failed logins maintained by the TOE, because this count is saved in non-volatile storage. The lock screen failure count is maintained through a successful DAR authentication.
- FIA_BLT_EXT.1: The TOE requires explicit user authorization before it will pair with a remote Bluetooth device. The user can make the TOE visible to other Bluetooth enabled devices and can attempt, via explicit user action, to pair with a visible device. Furthermore, the user must explicitly accept pairing attempts from other devices.
- 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 allows password for accounts to be composed of upper or lower case letters, numbers, and special characters including !, @, #, $, %, ^, &, *, (,), +, =, _, /, -, ', ", :, ;, comma, ?, `, ~, \, |, <, >, {, }, [, and]. The TOE defaults to requiring passwords to have a minimum of four characters but no more than sixteen. However, an MDM application can change these defaults.
- FIA_TRT_EXT.1: The TOE allows users to authenticate at the touchscreen. A user must authenticate through the standard User Interface (using the TOE touchscreen). The TOE limits the number of authentication attempts through the UI to no more than ten attempts within 30 seconds. Thus if the current [the nth] and prior nine authentication attempts have failed, and the n-9th attempt was less than 30 second

ago, the TOE will prevent any further authentication attempts until 30 seconds has elapsed. 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).

- FIA_UAU.7: The TOE allows the user to enter the user's password from the lock screen or from the DAR 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: The TOE requires that a user enter their password in order for the TOE to derive a master key that can be combined with a KEK chaining to the REK; thus allowing the TOE to decrypt the DEK that protects the data stored in the user data partition (including the long-term trusted channel key material which is stored in the user data partition).
- FIA_UAU_EXT.2: The TOE, when configured to require a user password, the TOE will allow a user to do the following things before successfully authenticating:
  - make an emergency call,
  - receive an incoming phone calls,
  - take screen shots (automatically named and stored internally by the TOE),
  - turn the TOE off,
  - enable or disable airplane mode,
  - configure sound/vibrate/mute,
  - use a flashlight,
  - use a calculator app,
  - use a voice recorder app, and
  - change notification display level.

  Beyond those actions, a user cannot perform any other actions other than observing notifications displayed on the lock screen until after successfully authenticating. The TOE allows a user to configure, on a per application basis, whether notifications will be displayed.
- FIA_UAU_EXT.3: The TOE requires the user to enter their password in order to unlock the TOE. Additionally the TOE requires the user to confirm their current password when accessing the 'Settings->Screen Lock' menu in the TOE's user interface. Only after entering their current user password can the user then elect to change their password.
- FIA_X509_EXT.1: The TOE checks the validity of all imported CA certificates by checking for the presence of the basicConstraints extension and that the CA flag is set to TRUE. These checks occur as the TOE imports the certificate (per RFC 5280). 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 such as 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 using a Certificate Revocation List (per RFC 5759).
- FIA_X509_EXT.2: The TOE uses X.509v3 certificates as part of protocol processing for EAP-TLS, TLS and HTTPS. The TOE comes with a built-in set of trusted certificates (a Trust Anchor Database). Users cannot remove any of these built-in CA certificates, but can make any as 'disabled' (which prevents them from being used to validate another certificate). Users and administrators can also import new trusted CA certificates into the Trust Anchor Database.

If, during the process of certificate verification, the TOE cannot establish a connection with the server acting as the CRL Distribution Point (CDP), the TOE EAP-TLS implementation accepts the certificate. Also, the TOE TLS and HTTPS implementations reject the certificate when the revocation server cannot be reached.FIA_X509_EXT.3: The TOE's Android operating system provides applications with several Java API Class of methods for validating certification paths (certificate chains) that establishing a trust chain from a certificate to a trust anchor.

6.4 Security management

The Security management function is designed to satisfy the following security functional requirements:

- FMT_MOF_EXT.1: The TOE provides the management functions as described in Table 5-2 Security Management Functions. The table includes annotations describing the roles that have access to each service and how to access the service.
- FMT_SMF_EXT.1: The TOE provides all management functions indicated as mandatory ('M') or implemented ('I') by Table 5-2 Security Management Functions.
- FMT_SMF_EXT.2: The TOE when unenrolled from an MDM alerts the administrator of the unenrollment and performs a factory reset.

6.5 Protection of the TSF

The Protection of the TSF function is designed to satisfy the following security functional requirements:

- FPT_AEX_EXT.1: The Linux kernel of the TOE's Linux Cells and Android Cell utilize 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 OKL4 hypervisor performs memory management that enforces read, write, and execute permissions on all pages of physical and virtual memory. The TOE's Android 6.0.1 operating system utilizes a 3.10.x Linux kernel, whose memory management unit (MMU) enforces read, write, and execute permissions on all pages of virtual memory within its control. 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'.
- FPT_AEX_EXT.3: The TOE's OKL4 Hypervisor and 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 -fstack-protector GCC feature. This compile option enables stack overflow protection and Android takes advantage of ARM v8 eXecute-Never to make the stack and heap non-executable. The vendor applies these protections to all TSF executable binaries and libraries (refer to 7, TSF Inventory for a complete list).
- FPT_AEX_EXT.4: The TOE protects itself from modification by untrusted subjects using a variety of methods. The first protection employed by the mobile device 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 Device Key (REK) by generating and securely storing the Device Key within hardware and making it accessible only to Android TrustZone software. The REK is used to protect all other keys in the key hierarchy. The TOE ensures that all TrustZone software is cryptographically signed and that the signature is verified when the TrustZone software is invoked.

  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).

Finally, the TOE's OKL4 hypervisor separates functions for a VPN and Data-at-rest encryption from the Android operating system, to ensure that the Outer DAR functio cannot be bypassed.

- FPT_BBD_EXT.1: The TOE's hardware and software architecture ensures separation of the application processor (AP) from the baseband through internal controls of the TOE's SoC.
- FPT_KST_EXT.1: The TOE does not store any plaintext key material in its internal Flash. 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 are wrapped with a KEK. Refer to Section 6.1 for a description of the keys, the key hierarchy, and the key storage protections afforded to all keys. 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 the OpenSSL cryptographic library running within Linux Cells (i.e., Android Cell, Block Cell) along with system-level executables that utilize KEKs within the Android Cell: wpa_supplicant, Gatekeeper[2] and KeyMaster. 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 SD CAr(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_EXT.4).
- FPT_KST_EXT.3: The TOE does not provide any way to export plaintext DEKs or KEKs (including all keys stored in the Secure Key Store) as the TOE chains or directly encrypts all KEKs to the REK.
- FPT_NOT_EXT.1: When TOE self-tests detect a failure of self-test or TSF verification tests, the TOE enters a non-operational mode. In this mode the TOE presents the page 'Decryption unsuccessful' with a 'Reset' button. Upon press of the Reset button the TOE will perform a factory reset of the device. Alternately, a power cycle of the device will attempt a restart to boot again. Upon boot (either following a power-cycle or a factory reset) the self-test and TSF verifications run again and can repeat the process if failures continue.
- FPT_STM.1: The TOE requires accurate time values for certificate validation, wpa_supplicant, and key store applications. These TOE components obtain time from the TOE using system API calls [e.g., time() or gettimeofday()]. An application cannot modify the system time as mobile applications need the android 'SET_TIME' permission to do so. Likewise, only a process with root privileges can directly modify the system time using system-level APIs. The TOE uses the Cellular Carrier time (obtained through the Carrier's network time server) as a trusted source; however, the user can also manually set the time through the TOE's user interface.
- FPT_TST_EXT.1: The TOE performs known answer power on self-tests (POST) on its cryptographic algorithms to ensure that they are functioning correctly. The kernel itself performs known answer tests on its cryptographic algorithms to ensure they are working correctly and the SecurityManager service invokes the self-tests of OpenSSL at start-up to ensure that those cryptographic algorithms are working correctly. Should any of the tests fail, the TOE will reboot to see if that will clear the error.

Table 6-4 Power-up Cryptographic Algorithm Known Answer Tests

Algorithm	Implemented in	Description
AES encryption/decryption	OpenSSL	Comparison of known answer to calculated valued
ECDH key agreement	OpenSSL	Comparison of known answer to calculated valued
DRBG random bit generation	OpenSSL	Comparison of known answer to calculated valued
HMAC-SHA	OpenSSL	Comparison of known answer to calculated valued

[2] Gatekeeper is an Android subsystem that is part of the TOE. It runs in Trustzone and supports user authentication.

SHA hashing	OpenSSL	Comparison of known answer to calculated valued
RSA	OpenSSL	Comparison of known answer to calculated valued
ECDSA	OpenSSL	Comparison of known answer to calculated valued

- 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, validates the signature of the Cog Secure Boot partition using the same public key by which the bootloader was signed. The Cog Secure Boot partition includes the Cog public key, which is used to sign the Hypervisor system. So the bootloader validates and (if validation is successful) starts the Cog Secure Boot image, which validates and (if validation is successful) starts the Hypervisor system. The Hypervisor System image contains the OKL4 Hypervisor along with the all other cells including the Android cell, Block Cell, Inner DAR Cell, Outer DIT cell, HSM Proxy cell, and C2-Agent cell. The Hypervisor System starts these cells, however, some cells can only initialize fully after other cells are operating (e.g., the Android cell and other cells must wait until their filesystems are made available by Outer DAR functionality of the Block cell). The Android User Data partition is available only after the partition has been successfully unlocked(decrypted).
- FPT_TUD_EXT.1: The TOE's user interface provides a method to query the current version of the TOE software/firmware (including the hypervisor version, and kernel version) and hardware (model and version). This information can be viewed through the command "Settings" -> "About" -> "Software information" -> "More" -> "Kernel 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: The TOE supports "side-loading" an update from SD card. A firmware update occurs if a new image is detected when the provisioning service detects a new image during the device boot process. When a new image is detected and before the new image is installed, the HTC bootloader validates the new Cog Secure Boot image in the same manner in which it validates an already installed boot image. This ensures that the new image has been signed using the same public key by which the bootloader was signed (i.e., a public key whose hash is burned into the device's fuses). Once installed, an update entirely replaces the Cog Secure Boot image, and the Hypervisor system image (which includes cell images), then restarts the boot process. The normal boot process then re-validates every part of the boot chain. The code that runs on the baseband processor is loaded as part of the Android image.

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

6.6 TOE access

The TOE access function is designed to satisfy the following security functional requirements:

- FTA_SSL_EXT.1: The TOE can transition into a locked state either as a result of exceeding the configured inactivity period, the user pressing the power button, or as a result of receiving a 'lock' command from an MDM server through an authenticated and protected channel. Upon entering a locked state the TOE obscures the previous content by displaying a lock screen. After being locked, the lock screen displays email notifications, text message notifications, call notifications, date, time, battery life, signal strength and carrier network. In order to act upon the notifications, the user must authenticate to the TOE.

  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 (i.e., FDE_DEK) 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.
- FTA_TAB.1: The TOE can be configured to display a message on the login screen and on the lock screen prior to user login.

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

6.7 Trusted path/channels

The Trusted path/channels function is designed to satisfy the following security functional requirements:

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

7. TSF Inventory

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

Name	Path	Securuty Function
dalvikvm	system/bin	Virtual Machine
dalvikvm32	system/bin	Virtual Machine
dalvikvm64	system/bin	Virtual Machine
keystore	system/bin	Keystore
time_daemon	system/bin	Time
qseecomd	system/bin	Trustone Daemon
qfipsverify	system/bin	Self tests
vold	system/bin	DAR
wpa_supplicant	system/bin	WPA2
libcrypto.so	system/lib	Crypto
libjavacrypto.so	system/lib	Crypto JNI
libkeystore_binder.so	system/lib	KeyStore
libssl.so	system/lib	SSL/TLS
libcrypto.so	system/lib64	Crypto
libjavacrypto.so	system/lib64	Crypto JNI
libkeystore_binder.so	system/lib 64	KeyStore
libssl.so	system/lib64	SSL/TLS