Huawei P8 (MDFPP20) Security Target

Version 0.93
May 24, 2016

Prepared for:

Huawei Device (Dongguan) Co., Ltd.

3F, Block 7, Vision Software Park, High Tech S9 ST., Nanshan, Shenzhen 518057, China

Prepared By:

www.gossamersec.com

LIST OF TABLES

1. Security Target Introduction

This section identifies the Security Target (ST) and Target of Evaluation (TOE) identification, ST conventions, ST conformance claims, and the ST organization. The TOE is Huawei P8 (GRA-L09) provided by Huawei Device (Dongguan) Co., Ltd.. The TOE is being evaluated as a mobile device.

The Security Target contains the following additional sections:

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

Conventions

The following conventions have been applied in this document:

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

1.1 Security Target Reference

ST Title – Huawei P8 (MDFPP20) Security Target

ST Version – Version 0.93

ST Date – May 24, 2016

1.2 TOE Reference

TOE Identification – Huawei Device (Dongguan) Co., Ltd. Huawei P8 (GRA-L09)

TOE Developer – Huawei Device (Dongguan) Co., Ltd.

Evaluation Sponsor – Huawei Device (Dongguan) Co., Ltd.

1.3 TOE Overview

The Target of Evaluation (TOE) is Huawei P8 (GRA-L09).

1.4 TOE Description

The Target of Evaluation (TOE) is the Huawei P8. The Huawei P8 is a smartphone based upon a HiSilicon K3V3+ processor. The Huawei P8 includes a 5.2 inch, Full HD 1090p resolution LCD display; a 13MP rear facing camera and 8MP front facing camera; the Huawei HiSilicon Kirin 930 chipset, the Mali-T628 MP4 GPU; 3GB of RAM; 16GB/64GB of built-in storage; and a microSD card slot, doubling as a secondary SIM slot on the Dual SIM model. The Huawei P8 ships with the Android 5.0 (Lollipop) with EmotionUI 3.1. The HiSilicon Kirin 930 chipset couples slower, low-speed processor cords (quad-core 1.5GHz Cortex-A53) with high-power process cores (quad-core 2GHz Cortex-A53) to reduce energy consumption.

The Huawei P8 is a mobile device that supports individual users as well as corporate enterprises. The Huawei P8 is based upon Android 5.0 as customized by Huawei.

The TOE provides wireless connectivity and creates a runtime environment for applications designed for the mobile Android environment. The TOE also provides telephony features (make and receive phone calls, send and receive SMS messages), networking features (connect to Wi-Fi networks, send and receive MMS messages, connect to mobile data networks).

The Huawei P8 contains between 16 and 64 GB of built-in storage and 3 GB of memory.

The evaluated device is the Huawei P8 Mobile Device (Model Number: GRA_L09).

The software identification for the evaluated devices is as follows:

- Kernel Version: 3.10
- Build Number: GRA-L09C900B078

1.4.1 TOE Architecture

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

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

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

1.4.1.1 Physical Boundaries

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

1.4.1.2 Logical Boundaries

This section summarizes the security functions provided by Huawei P8:

- Cryptographic support
- User data protection

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

1.4.1.2.1 Cryptographic support

The TOE includes cryptographic modules with CAVP validated algorithms that are used for 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, right, and key encryption keys) used by the TOE. Many of these cryptographic functions must also be 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 and the frequency of entering passwords is limited and when a configured number of failures occurs, the TOE is wiped of user data. Passwords can be constructed using upper and lower cases characters, numbers, and special characters and password lengths of 16 characters are supported.

The TOE can serve as an 802.1X supplicant. The TOE can use X509v3 certificates and can perform certificate validation EAP-TLS, TLS, and HTTPS exchanges.

1.4.1.2.4 Security management

The TOE provides all the interfaces necessary to manage the security functions identified by this Security Target as well as other functions commonly found in mobile devices. Many of the available functions are available to the users of the TOE while many are restricted to administrators operating through a Mobile Device Management solution once the TOE has been enrolled.

1.4.1.2.5 Protection of the TSF

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

The TOE includes functions to perform self-tests and software/firmware integrity checking so that it can detect when it is failing or may be corrupt. If any self-test fails, the TOE does not go into an operational mode. It 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 also has the capability to display an advisory message (banner) when users unlock the TOE for use.

The TOE can attempt to connect to wireless networks as configured.

1.4.1.2.7 Trusted path/channels

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

1.4.2 TOE Documentation

Huawei offers the following documentation to users for the installation and operation of their product. The following list of documents was examined as part of the evaluation.

[Guide] Administrator Guidance Instructions, version 0.5, February 14, 2016.

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 Huawei P8 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 augmented with ALC_TSU_EXT.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 augmented with ALC_TSU_EXT.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 Huawei P8 TOE.

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_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: Default TOE Access Banners
	FTA_WSE_EXT.1: Extended: Wireless Network Access
FTP: Trusted path/channels	FTP_ITC_EXT.1: Extended: Trusted channel Communication

Table 5-1 TOE Security Functional Components

5.1.1 Cryptographic support (FCS)

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

FCS_CKM.1(1).1

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

- ***[RSA schemes] using cryptographic key sizes of [2048-bit or greater] that meet the following: [o ANSI X9.31-1998, Section 4.1;],***
- ***[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']; and [
- ***[Elliptic curve-based key establishment schemes] that meets the following: [NIST Special Publication 800-56A, 'Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography'];***
- ***[Finite field-based key establishment schemes] that meets the following: [NIST Special Publication 800-56A, 'Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography'];***].

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-protected***] REK with a [***asymmetric***] key of ~~size~~ strength [1][***256 bits***].

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.

FCS_CKM_EXT.1.3

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

FCS_CKM_EXT.1.4

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

[1] Added selection & changed size to strength per TD0038.

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 [***asymmetric KEKs of [256 bits] security strength***] corresponding to at least the security strength of the keys encrypted by the KEK.[2]

FCS_CKM_EXT.3.2

The TSF shall generate all KEKs using one 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), d) Combine the KEK from other KEKs in a way that preserves the effective entropy of each factor by [using an XOR operation]***].

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***] ~~following by a read-verify~~[3].
  - 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[4], the destruction shall be executed [***by a single direct overwrite consisting of zeros followed by a read-verify***].
  - For non-volatile flash memory that is wear-leveled, the destruction shall be executed [***by a single direct overwrite consisting of zeros***].
  - 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.

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.

---

[2] Element rewritten per TD0038.
[3] Phrase deleted per TD0028.
[4] Wear-leveling phrase and next bullet added by TD0047 and selection modified by TD0057.

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 Key Wrap (KW) (as defined in NIST SP 800-38F),***
AES-GCM (as defined in NIST SP 800-38D),]
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 [***- no other algorithms***].

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***] and cryptographic key sizes [**160, 256, 384**] and message digest sizes 160 and [***256, 384***] 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-1, SHA-256, SHA-384, SHA-512***]], with [**2000**] 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 [***no other action***] 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 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)]***].

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 [***selected algorithms***] in FCS_CKM.2(1)[5]
- The following algorithms in FCS_COP.1(1): AES-CBC, [***no other modes***]
- All mandatory and selected algorithms in FCS_COP.1(3)
- All mandatory and selected algorithms in FCS_COP.1(2)
- All mandatory and selected algorithms in FCS_COP.1(4)

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

5.1.1.20 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 [***no other keys***].

FCS_STG_EXT.1.2

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

FCS_STG_EXT.1.3

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

FCS_STG_EXT.1.4

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

FCS_STG_EXT.1.5

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

[5] Changed text of the first bullet per TD0059.

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 [***all software-based key storage***] by KEKs that are [***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 [*all software-based key storage*] shall be encrypted using one of the following methods: [***using AES in the [****GCM****, CBC mode]***].[6]

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 [***all software-based key storage***] by [***- [GCM] cipher mode for encryption according to FCS_STG_EXT.2; a hash (FCS_COP.1(2)) of the stored key that is encrypted 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 [***no other TLS version***] supporting the following ciphersuites: [- Mandatory Ciphersuites: o TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246 - [***o TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,***
o TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246,
o TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,
o TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,
o TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,

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: o TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246 - [***o TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,***
o TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246,
o TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,
o TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,
o TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,

[6] Updated per TD0038.

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.

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 [***application processes***] from accessing [***private***] data stored by other [***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 [***CBC***] mode with key size [***256***] bits.

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

FDP_IFC_EXT.1.1
The TSF shall [***provide an interface to VPN 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 [***Bluetooth LE***] to provide a protected communication channel between the non-TSF application and another IT product that is logically distinct from other communication channels, provides assured identification of its end points, protects channel data from disclosure, and detects modification of the channel data.

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

5.1.3 Identification and authentication (FIA)

5.1.3.1 Authentication failure handling (FIA_AFL_EXT.1)

FIA_AFL_EXT.1.1

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

FIA_AFL_EXT.1.2

When the defined number of unsuccessful authentication attempts has been 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 [***no other keys***] at startup.

5.1.3.8 Extended: Timing of Authentication (FIA_UAU_EXT.2)

FIA_UAU_EXT.2.1

The TSF shall allow [***[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, change display brightness, configure sound/vibrate/mute, choose the keyboard input method, use the flashlight, use a calculator app, use a recorder app and change notification display level.] on behalf of the user to be performed before the user is authenticated.

FIA_UAU_EXT.2.2

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

5.1.3.9 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 [***[when changing user lock screen, when changing one's password, and when initializing the keystore.]***].

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:

o Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
o Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
o (Conditional) Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) in the extendedKeyUsage field.

FIA_X509_EXT.1.2

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

5.1.3.11 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 [***not accept the certificate***].

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.	M		M	M
3. enable/disable the VPN protection: a. across device [**c.** ***no other method***] The user can configure and then enable the TOE's VPN to protect traffic. The administrator (through an MDM Agent that utilizes the TOE's MDM APIs) can	M		I	I

restrict the TOE's ability to connect to a VPN.				
4. enable/disable [**nfc, Bluetooth, Wi-Fi, and cellular radios**] The administrator can disable the radios using the TOE's MDM APIs. Once disabled, a user cannot enable the radio. The TOE's radio's operate at frequencies of 2.4 GHz (NFC/Bluetooth), 2.4/5 GHz (Wi-Fi), and 850 MHz (4G/LTE).	M			
5. enable/disable [**camera, microphone**]: a. across device [**c.** ***no other method***] An administrator may configure the TOE (through an MDM agent utilizing the TOE's MDM APIs) to turn off the camera and or microphones. If the administrator has disabled either the camera or the microphones, then the user cannot use those capture devices.	M		I	I[7]
6. specify wireless networks (SSIDs) to which the TSF may connect [8] An administrator can specify a list of wireless network SSIDs to which the TOE may connect and can restrict the TOE to only allow a connection to the specified SSIDs.				
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 Both users and administrators (using the TOE's MDM APIs) can define wireless connection policies for specific SSIDs.	M		M	
8. transition to the locked state Both users and administrators (using the TOE's MDM APIs) can transition the TOE into a locked state.	M		M	-
9. full wipe of protected data Both users and administrators (using the TOE's MDM APIs) can force the TOE to perform a full wipe (factory reset) of data.	M		M	
10. configure application installation policy by [***c. denying installation of applications***] The administrator using the TOE's MDM APIs can configure the TOE so that applications cannot be installed and can also block the use of the Google Market Place.	M		M	M
11. import keys/secrets into the secure key storage Both users and administrators (using the TOE's MDM APIs) can import secret keys into the secure key storage.	M		I	
12. destroy imported keys/secrets and [***no other keys/secrets***] in the secure key storage Both users and administrators (using the TOE's MDM APIs) can destroy secret keys in the secure key storage.	M		I	
13. import X.509v3 certificates into the Trust Anchor Database	M		M	

[7] Changed from Mandatory to Optional by TD0044.
[8] Changed from Mandatory to Optional per TD0064.

Both users and administrators (using the TOE's MDM APIs) can import X.509v3 certificates into the Trust Anchor Database.				
14. remove imported X.509v3 certificates and [***default X.509v3 certificates***] in the Trust Anchor Database Both users and administrators (using the TOE's MDM APIs) can remove imported X.509v3 certificates from the Trust Anchor Database as well as disable any of the TOE's default Root CA certificates (in the latter case, the CA certificate still resides in the TOE's read-only system partition; however, the TOE will treat that Root CA certificate and any certificate chaining to it as untrusted).	M		I	
15. enroll the TOE in management TOE users can enroll the TOE in management according to the instructions specific to a given MDM. Presumably any enrollment would involve at least some user functions (e.g., install an MDM agent application) on the TOE prior to enrollment.	M		M [9]	
16. remove applications Users can uninstall applications on the TOE.	M		M	
17. update system software Users can check for updates and cause the device to update if an update is available. A wide range of possibilities exist for the administrator since and using the MDM APIs the administrator can query the version of the TOE and installed applications and an MDM agent on the TOE could issues pop-ups, initiate updates, block communication, etc. until any necessary updates are completed.	M		M	
18. install applications Users can install applications on the TOE.	M		M	
19. remove Enterprise applications Users can uninstall applications on the TOE.	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***] TOE users can enable Bluetooth discoverable mode for a short period of time and can also change the device name which is used for the Bluetooth name. Additional wireless technologies include Android Beam which are related to NFC and can be enabled and disabled by the TOE user.	M			
21. enable/disable display notification in the locked state of: [**f.** ***all notifications***] TOE users can configure the TOE to allow or disallow notifications while in a locked state.	M			
22. enable/disable all data signaling over [**USB**]				
23. enable/disable [Hotspot, **Wi-Fi tethering, USB tethering, and Bluetooth tethering**]	I	I		
24. enable/disable developer modes	I	I		
25. enable data-at rest protection	I	I		

[9] Changed Mandatory/Optional values per TD0058.

26. enable removable media's data-at-rest protection	I	I		
27. enable/disable bypass of local user authentication				
28. wipe Enterprise data				
29. approve [import] 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				
37. configure the auditable items				
38. retrieve TSF-software integrity verification values				
39. enable/disable [a. USB mass storage mode]				
40. enable/disable backup to [remote system]				
41. enable/disable [a. Hotspot functionality authenticated by [pre-shared key], b. USB tethering authenticated by [no authentication]]	I	I		
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 [c. no other method]	M			
45. [**enable/disable voice command control of device functions**]				

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

FMT_SMF_EXT.2.1

The TSF shall offer [***wipe of protected data, alert the administrator***] 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 Extended: Key Storage (FPT_KST_EXT.1)

FPT_KST_EXT.1.1

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

5.1.5.6 Extended: No Key Transmission (FPT_KST_EXT.2)

FPT_KST_EXT.2.1

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

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

FPT_KST_EXT.3.1

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

5.1.5.8 Extended: 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***].

5.1.5.9 Reliable time stamps (FPT_STM.1)

FPT_STM.1.1

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

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

FPT_TST_EXT.1.1

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

5.1.5.11 Extended: TSF Integrity Testing (FPT_TST_EXT.2)

FPT_TST_EXT.2.1

The TSF shall verify the integrity of the bootchain up through the Application Processor OS kernel, and [***no other executable code***], stored in mutable media prior to its execution through the use of [***a digital signature using a hardware-protected asymmetric key, a hardware-protected hash***].

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

FPT_TUD_EXT.1.1
The TSF shall provide authorized users the ability to query the current version of the TOE firmware/software.

FPT_TUD_EXT.1.2
The TSF shall provide authorized users the ability to query the current version of the hardware model of the device.

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

5.1.5.13 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 system 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 [***hash***].

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.

Requirement Class	Requirement Component
ADV: Development	ADV_FSP.1: Basic functional specification
AGD: Guidance documents	AGD_OPE.1: Operational user guidance
	AGD_PRE.1: Preparative procedures
ALC: Life-cycle support	ALC_CMC.1: Labelling of the TOE
	ALC_CMS.1: TOE CM coverage
	ALC_TSU_EXT.1: Timely Security Updates
ATE: Tests	ATE_IND.1: Independent testing - conformance
AVA: Vulnerability assessment	AVA_VAN.1: Vulnerability survey

Table 5-3 **Assurance Components**

5.2.1 Development (ADV)

5.2.1.1 Basic functional specification (ADV_FSP.1)

ADV_FSP.1.1d

The developer shall provide a functional specification.

ADV_FSP.1.2d

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

ADV_FSP.1.1c

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

ADV_FSP.1.2c

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

ADV_FSP.1.3c

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

ADV_FSP.1.4c

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

ADV_FSP.1.1e

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

ADV_FSP.1.2e

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

5.2.2 Guidance documents (AGD)

5.2.2.1 Operational user guidance (AGD_OPE.1)

AGD_OPE.1.1d

The developer shall provide operational user guidance.

AGD_OPE.1.1c

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

AGD_OPE.1.2c

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

AGD_OPE.1.3c

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

AGD_OPE.1.4c

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

AGD_OPE.1.5c

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

AGD_OPE.1.6c

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

AGD_OPE.1.7c

The operational user guidance shall be clear and reasonable.

AGD_OPE.1.1e

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

5.2.2.2 Preparative procedures (AGD_PRE.1)

AGD_PRE.1.1d

The developer shall provide the TOE including its preparative procedures.

AGD_PRE.1.1c

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

AGD_PRE.1.2c

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

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

AGD_PRE.1.2e
The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared securely for operation.

5.2.3 Life-cycle support (ALC)

5.2.3.1 Labelling of the TOE (ALC_CMC.1)

ALC_CMC.1.1d
The developer shall provide the TOE and a reference for the TOE.

ALC_CMC.1.1c
The TOE shall be labelled with its unique reference.

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

5.2.3.2 TOE CM coverage (ALC_CMS.1)

ALC_CMS.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 implements cryptographic algorithms in accordance with the following NIST standards and has received the following CAVP algorithm certificates. The OpenSSL FIPS Object Module provides the following algorithms

Algorithm	NIST Standard	SFR Reference	Cert#
AES 128/256 CBC, CCM, GCM	FIPS 197, SP 800-38A/C/D/F	FCS_COP.1(1)	3565
AES KW 128/256	SP800-38F	FCS_COP.1(1)	3565
CVL ECCCDH	SP 800-56A	FCS_CKM.2(1)	703
CVL KAS FFC/ECC	SP 800-56A	FCS_CKM.2(1)	800
DRBG AES-256-CTR	SP 800-90A	FCS_RBG_EXT.1(1)	909 1131
DSA SIG(gen)/SIG(ver)/Key(gen)	FIPS 186-4	FCS_CKM.2(1)	1069
HMAC SHA-1/256/384	FIPS 198-1 & 180-4	FCS_COP.1(4)	2271
RSA SIG(gen)/SIG(ver)/Key(gen)	FIPS 186-4	FCS_CKM.1(1) FCS_CKM.1(2) FCS_COP.1(3)	1833
SHS SHA-1/256/384/512	FIPS 180-4	FCS_COP.1(2)	2933

Table 6-1 OpenSSL Cryptographic Algorithms

The TOE's application processor includes a source of hardware entropy that the TOE distributes throughout TrustZone[10] and Android user space. The TOE's RBGs make use of that entropy when seeding/instantiating themselves.

FCS_CKM_EXT.2: The TOE uses data encryption keys protecting the Full disk encryption (data-at-rest) and SD card encryption functionality respectively. These DEK are each 256-bit AES keys, generated from the hardware-based CTR_DRBG meeting FCS_RBG_EXT.1. Each of these data encryption keys are encrypted by the combination of the KEK (for the appropriate function) and a master key derived from the user's password. Other DEKs include WPA2 keys used in wireless WPA2 EAP-TLS (which are 256 bit AES keys), HTTPS/TLS session keys (128/256 bit AES keys), Bluetooth keys (128 bit AES keys), and mobile application keys (2048 bit RSA keys). All keys are generated using the TOE's SP 800-90A AES-CTR-256 DRBG. As the DRBG is an AES-CTR-256 based SP 800-90A DRBG, the strength of the key requested is equal to the strength of the keys used for encryption/decryption.

[10] TrustZone is the term for Andriod's use of the 'secure mode' provided by an ARM processor. 'Secure mode' provides hardware isolation for security critical operations such as cryptography. (See the ARM TrustZone description at http://www.arm.com/products/processors/technologies/trustzone/index.php).

FCS_CKM_EXT.3: The TOE derives all password based KEKs by combining a user's password with an RBG generated salt value using PBKDFv2 (in accordance with FCS_COP.1(5)). The TOE derives the KEKs from the REK using a NIST SP 800-108 derivation scheme, using a CMAC KDF in counter mode. Finally, all other KEKS are generated using an AES-256-CTR DRBG meeting FCS_RBG_EXT.1. The DRBG generates keys that are the same size as the key being requested. The TOE combines the password-based KEK with a KEK chained to the REK, before using the resulting value to protect keys used to protect Data-at-Rest.

Key Name	Key Type	Usage	Algorithm / Size	Generated From / Derived from
DAR-DEK	DEK	Encrypt / Decrypt user data	AES-CBC 256-bits	Generated from hardware entropy
SD Card DEK	DEK	SD Card data encryption/decryption key	AES-CBC 256-bit	Generated from hardware entropy
WPA2 keys	DEK	Encrypt/decryption Wi-Fi/802.11 communications	32-byte nonce	Generated from hardware entropy
TLS/HTTPS session keys	DEK	Encryption/decryption of TLS/HTTPS sessions	AES – 128/256 CBC	OpenSSL – Derived as specified in TLS/HTTPS protocol
Bluetooth EPR/LE	DEK	Encryption/decryption of Bluetooth communications	AES – 128	Generated within Broadcom BT Chipset
Mobile application keys	DEK	Keys generated / used by various mobile applications	RSA – 2048 bits	OpenSSL (if Android interface is used)

FCS_CKM_EXT.6: The Huawei's Mobile Device utilizes Salt values in the following locations:

1. Salts used as part of derivation process of the Master key (MK1) and SD Card key (MK2)
2. TLS/HTTPS (c_random, pre-master secret, ECDHE_*, DHE_*)
3. WPA2 nonces

All of these values come from an RBG meeting FCS_RBG_EXT.1.

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, RSA, and ECDH. The TOE generates RSA and DH/ECDH (including P-256, P384 and P-521) keys in its OpenSSL software library (which generates the keys in accordance with ANSI X9.31 and FIPS 186-4 respectively). The TOE supports generating keys with a security strength of 112-bits and larger, thus supports 2048-bit RSA and DH keys, and 256-bit ECDH keys.
- 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 WiFi CERTIFIED Interoperability Certificate from the WiFi Alliance

(Certification ID: WFA59359[11]). The WiFi Alliance maintains a website providing further information about the testing program: http://www.wi-fi.org/certification.

- FCS_CKM.2(1): The TOE 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. 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 HUK (Hardware Unit Key) that is provided by fuses on the processor. The HUK is used to derive an REK that are generated per FCS_RBG_EXT.1. The HUK is never entered, read, exported, or updated after manufacturing. Only operation allowed by or performed by the TrustZone kernel on a REK is that it be used for NIST SP 800-108 key derivation of the KEKs.
- FCS_CKM_EXT.2: All data encryption keys (DEKs) used by the TOE are generated using its approved RBGs. In all cases, the TOE generates DEKs using a compliant RBG seeded with entropy from the TOE's hardware entropy source. The RBGs generate DEKs corresponding to the security strength of AES key sizes of 128 and 256-bits. The TOE utilizes the following DEKs:
    a. Full-disk-encryption (Data-at-rest)
    b. SD Card DEK
    c. WPA2 keys
    d. TLS/HTTPS session keys
    e. TLS/HTTPS pre-master secret
    f. Bluetooth EPR/LE keys
    g. Mobile application keys
- FCS_CKM_EXT.3: The TOE derives all password based KEKs from a user's password using PBKDFv2. The TOE also uses an XOR operation to combine KEKs for full disk encryption and SD card encryption, thus preserving the effective entropy of each KEK. All other KEKs are generated from an RBG meeting FCS_RBG_EXT.1.
- 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 DEK that is used to encrypt the user data partition. Because the TOE's keystore resides within the user data partition, TOE effectively cryptographically erases those keys when clearing the Full disk encryption DEK. In turn, the TOE clears the Full disk encryption DEK and SD Card DEK 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.

[11] Note that the WiFi Certification was performed prior to product release when the internal name for the TOE was "Grace", thus the certificate showing the model number of GRA-UL00 is the same product as the TOE identified in section 1.2.

- 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 Full disk encryption DEK protecting the user data partition, clears that key from memory, reformats the partition, and then reboots. The TOE's clearing of the Full disk encryption DEK follows the requirements of FCS_CKM_EXT.4.
- FCS_CKM_EXT.6: The salts used by the TOE are obtained from a DRBG satisfying FCS_RBG_EXT.1. These include the salts used to generate the KEK that is pre-cursor data to the DEK that used to protect the full-disk-encryption and SD card encryption.
- FCS_COP.1(1): The TOE performs encryption and decryption using AES in accordance with the standards, key sizes and modes referenced in the tables above.
- FCS_COP.1(2): The TOE performs cryptographic hashing using SHA algorithms in accordance with the standards and message digest sizes referenced in the tables above.
- FCS_COP.1(3): The TOE performs cryptographic signature generation and verification using either an RSA scheme that is in accordance with standards referenced in the tables above. The TOE is capable of generating keys with key size of 2048-bits.
- FCS_COP.1(4): The TOE performs keyed-hash message authentication using the HMAC-SHA-1, HMAC-SHA-256, and HMAC-SHA-384 algorithms, meeting the standards referenced in the tables above. These include key sizes and message digest sizes of 160, 256 and 384.
- FCS_COP.1(5): The TOE conditions a user's password per SP 800-132 using PBKDF2 using HMAC-SHA-256 with 2000 iterations, to combine a 128-bit salt with the user's password to obtain 256-bit master key which are used to protect the full disk encryption and SD card encryption keys. The TOE utilizes several mechanism to increase the computational complexity of the protections afforded by keys derived from passwords. First, the TOE performs 2000 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) using TLS as described by FCS_TLSC_EXT.2 so that (mobile and system) applications executing on the TOE can act as HTTPS clients and securely connect to external servers using HTTPS. Huawei has also modified TLS APIs to ensure that certificate checking performed by the TOE satisfies FIA_X509_EXT.1, FIA_X509_EXT.2 and FIA_X509_EXT.3.
- FCS_IV_EXT.1: The initialization vectors (IVs) used by the TOE for AES CBC encryption are generated by a DRBG satisfying FCS_RBG_EXT.1. The IVs that are used to support the encryption of the DEKs for full disk encryption and SD card encryption use output from the PBPKF2 function. This output would meet SP800-38D section A.5 Key/IV pair uniqueness by virtue of the fact that for these IV (which are deterministically established from the user's password and the 128-bit salt) the collision probability would not exceed 2^{-32}.
- FCS_RBG_EXT.1: The TOE provides a number of different RBGs including the following:

  a. An SP 800-90A AES-256 CTR_DRBG provided in the hardware.

  b. An SP800-90A AES-256 CTR DRBG provided by OpenSSL.

  The TOE initializes each RBG with sufficient entropy ultimately accumulated from a TOE-hardware-based noise source. The hardware based AES CTR_DRBG is seeded from the hardware noise source. Output

from the hardware based AES CTR_DRBG provides seed material for both the OpenSSL module (384-bits). These sources for RBG all provide a security strength of at least 256-bits.

The hardware based CTR-DRBG is also made available to Android applications through a character device /dev/tee_random. These sources for RBG all provide a security strength of at least 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 user, the administrator and mobile applications the ability to import and use asymmetric public and private keys into the TOE's hardware-isolated Secure Key Storage. Additionally, the user and administrator can request the TOE to destroy the keys stored in the Secure Key Storage. While normally mobile applications cannot use or destroy the keys of another application, applications that share a common application developer (and are thus signed by the same developer key) may do so. In other words applications with a common developer may use and destroy each other's keys located within the Secure Key Storage.

  The TOE utilizes a TrustZone implementation of KeyMaster[12] key storage service to provide hardware-isolated secure key storage, for asymmetric private keys.
- FCS_STG_EXT.2: KEKs derived from the REK are never stored in non-volatile memory. These KEKs are combined (via XOR) with a key derived from the user's password to protect the DEK. Encryption of the DEK is performed using AES-GCM with 256-bit keys, entirely within the TrustZone. All DEKs and KEKs stored in non-volatile memory are encrypted using AES-GCM with 256-bit keys.

  One KEK (which is derived from the REK) is used to protect asymmetric key pairs. These key pairs are used with RSA and are in cleartext only within TrustZone. The private keys are encrypted with the KEK using AES-256 in CBC mode with a saved hash to provide integrity.

  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: See the description of FCS_STG_EXT.2 above.
- FCS_TLSC_EXT.1: The TSF supports TLS versions 1.0 and supports the following pre-configured cipher suites use with EAP-TLS as part of WPA2.
  - TLS_RSA_WITH_AES_128_CBC_SHA
  - TLS_RSA_WITH_AES_256_CBC_SHA
  - TLS_DHE_RSA_WITH_AES_128_SHA
  - TLS_DHE_RSA_WITH_AES_256_SHA
  - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
  - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA

  The TOE does not support any other ciphersuites with EAP-TLS, as the remainder of selectable cipher suites are based on SHA256, which requires TLS v1.2.

  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

[12] See https://source.android.com/security/keystore/ for more information about Keymaster key storage.

during TLS negotiation is deemed invalid, the TOE rejects the TLS channel negotiation (i.e., no connection is established). The TOE offers the curves secp256r1, secp384r1 and secp521r1 in the TLS Client Hello for use when an ECDHE ciphersuite is selected.

- FCS_TLSC_EXT.2: The TOE provides mobile applications (through its Android API) the use of TLS versions 1.0 and 1.2 including support for the following pre-configured cipher suites.
  - TLS_RSA_WITH_AES_128_CBC_SHA
  - TLS_RSA_WITH_AES_256_CBC_SHA
  - TLS_DHE_RSA_WITH_AES_128_SHA
  - TLS_DHE_RSA_WITH_AES_256_SHA
  - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
  - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA

  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. 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 does not support certificate pinning.

6.2 User data protection

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

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

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

  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).

- FDP_DAR_EXT.1: The TOE provides user Data-At-Rest AES-256 GCM encryption for all data stored on the TOE in the user data partition (which includes both user data and TSF data). The TOE also has TSF data relating to key storage for TSF keys not stored in the system's Secure Key Store. The TOE separately encrypts those TSF keys and data. Additionally, the TOE includes a read-only 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-256 bit DEK in GCM mode to encrypt the entire partition. The TOE also provides AES-256 GCM mode encryption of protected data stored on the external SD Card. The TOE encrypts each individual file stored on the SD Card, generating a unique FEK for each file.
- FDP_IFC_EXT.1: The TOE provides mobile applications with an API to ensure IP routes are configured to direct IP traffic through the VPN. The API ties IP routing table rules to processes with specific UIDs (namely the UID of the currently logged in user).
- FDP_STG_EXT.1: The TOE's Trusted Anchor Database consists of the built-in certs and any additional user or admin/MDM loaded certificates. The built-in certs are (individually stored in device's read-only system image in the /system/etc/security/cacerts directory, and the user can individually disable through Android's user interface [Settings->Security-> Trusted Credentials]. Because the built-in CA certificates reside on the read-only system partition, the TOE places a copy of any disabled built-in certificate into the /data/misc/user/X/cacerts-removed/ directory, where "X" represents the user's number (which starts at 0). The TOE stores added CA certificates in the corresponding /data/misc/user/X/cacerts-added/ directory and also stores a copy of the CA certificate in the user's Secure Key Storage (residing in the /data/misc/keystore/user_X/ directory).
- FDP_UPC_EXT.1: The TOE provides APIs allowing non-TSF applications (mobile applications) the ability to establish a secure channel using TLS, HTTPS, and Bluetooth DR/EDR & LE. Mobile applications can use the following Android APIs for TLS, HTTPS, and Bluetooth respectively:

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

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

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

6.3 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 through 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 cryptolock 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 !, @, #, $, %, ^, &, *, (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 through external ports (either a USB keyboard or a Bluetooth keyboard paired in advance of the login attempt). If not using an external keyboard, 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 five attempts within 30 seconds (irrespective of what keyboard the operator uses). Thus if the current [the nth] and prior four authentication attempts have failed, and the n-4th 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 cryptolock 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.
- 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, choose the keyboard input method, 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->Security->LockScreen->Password” menu in the TOE’s user interface. Only after entering their current user password can the user then elect to change their password.
- FIA_X509_EXT.1: The TOE checks the validity of all imported CA certificates by checking for the presence of the basicConstraints extension and that the CA flag is set to TRUE as the TOE imports the certificate (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 [e.g., the CA flag in the basic constraints extension for a CA certificate, or that a server certificate contains the Server Authentication purpose in the ExtendedKeyUsagefield]), then verifies each certificate in the chain (applying the same rules as above, but also ensuring that the Issuer of each certificate matches the Subject in the next rung 'up' in the chain and that the chain ends in a self-signed certificate present in either the TOE's trusted anchor database or matches a specified Root CA), and finally the TOE performs revocation checking for all certificates in the chain 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 will deem the server's certificate as invalid and not establish a TLS connection with the server.

- FIA_X509_EXT.3: The TOE's Android operating system provides applications 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 described in Table 5-2 Security Management Functions in section 5.1.4.2. 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 wipes all protected data and alerts the administrator of the unenrollment.

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 Android operating system provides address space layout randomization utilizing the get_random_int(void) kernel random function to provide eight unpredictable bits to the base address of any user-space memory mapping. The random function, though not cryptographic, ensures that one cannot predict the value of the bits.
- FPT_AEX_EXT.2: The TOE's Android 5.0 operating system utilizes a 3.4 Linux kernel, whose memory management unit (MMU) enforces read, write, and execute permissions on all pages of virtual memory. The Android operating system (as of Android 2.3) sets the ARM No eXecute (XN) bit on memory pages and the TOE's ARMv7 Application Processor's Memory Management Unit (MMU) circuitry enforces the XN bits. From Android's documentation (https://source.android.com/devices/tech/security/index.html), Android 2.3 forward supports "Hardware-based No eXecute (NX) to prevent code execution on the stack and heap". Section B4.2 of the ARM v7 Architecture Reference Manual contains additional details about the MMU of ARM-based processors:

  http://www.club.cc.cmu.edu/~mjrosenb/ARM%20v7%20Architecture%20Reference%20Manual.pdf.

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

- FPT_KST_EXT.1: The TOE does not store any plaintext key material in its internal Flash or on external SD cards. This ensures that irrespective of how the TOE powers down (e.g., a user commands the TOE to power down, the TOE reboots itself, or battery depletes or is removed), all keys in internal Flash or on an external SD Card are wrapped with a KEK. Please refer to section 6.1of the TSS for further information (including the KEK used) regarding the encryption of keys stored in the internal Flash and on external SD Cards. As the TOE encrypts all keys stored in Flash, upon boot-up, the TOE must first decrypt any keys in order to utilize them.
- FPT_KST_EXT.2: The TOE itself (i.e., the mobile device) comprises a cryptographic module that utilizes cryptographic libraries including OpenSSL, Bouncy Castle JCE, and the following system-level executables that utilize KEKs: dm-crypt, eCryptfs, wpa_supplicant, and the Secure Key Store. The TOE ensures that plaintext key material does not leave this cryptographic module by only allowing the system-level executables access to the plaintext KEK values that protect all other keys in the TOE. The TSF software (the system-level executables) protects the plaintext KEKs and any plaintext DEK values in memory both by not providing any access to these values and by clearing them when no longer needed (in compliance with FCS_CKM_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 the TOE encounters a critical failure (either a self-test failure or TOE software integrity verification failure), the TOE transitions to a non-operational mode. The user may attempt to power-cycle the TOE to see if the failure condition persists, and if it does persist, the user may attempt to boot to the recovery mode/kernel to wipe data and perform a factory reset in order to recover the device.
- FPT_STM.1: The TOE requires time for certificate validation, wpa_supplicant, and key store. 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 utilizes a custom built service to do the power-up self-tests. When the services receives a BOOT_COMPLETE message, the service will start up and perform the self-test for the crypto modules (including OpenSSL and hardware crypto module). If the self-test failed, the TOE presents a system dialog to user, and then goes into recovery mode once the user confirms the message. The self-test includes all of the cryptographic algorithm known answer tests shown in table.

Table 6-2 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
RSA sign/verify	OpenSSL	Comparison of known answer to calculated valued
SHA hashing	OpenSSL	Comparison of known answer to calculated valued

AES encryption/decryption	Bouncy Castle JCE	Comparison of known answer to calculated valued
HMAC-SHA	Bouncy Castle JCE	Comparison of known answer to calculated valued
RSA sign/verify	Bouncy Castle JCE	Comparison of known answer to calculated valued
SHA hashing	Bouncy Castle JCE	Comparison of known answer to calculated valued
AES encryption/decryption	Linux Kernel	Comparison of known answer to calculated valued
HMAC-SHA	Linux Kernel	Comparison of known answer to calculated valued
SHRNG random bit generation	Linux Kernel	Comparison of known answer to calculated valued
SHA hashing	Linux Kernel	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 XLoader, fastboot[13], and Android OS (using a public key whose hash resides in the processor's internal fuses). The processor first verifies the Xloader image, which verifies and loads fastboot. The fastboot image includes an image signature and an RSA public key that was used to sign the image. The Xloader verifies the RSA public key, which is in a certificate that must be signed with the same Huawei signing key whose hash is in the processor's internal fuses. Once the RSA public key has been verified, the fastboot image signature is verified, and if the signature is valid fastboot is started. Fastboot verifies the boot.img that contains the OS Linux Kernel and the TZ image, prior to staring the TZ image, which eventually starts the OS Linux Kernel.
- FPT_TUD_EXT.1: The TOE's user interface provides a method to query the current version of the TOE software/firmware (Android version, baseband version, kernel version, and build number) and hardware (model number). Additionally, the TOE provides users the ability to review the currently installed apps (including 3rd party 'built-in' applications) and their version.
- FPT_TUD_EXT.2: When in CC mode, the TOE verifies all updates to the TOE software using a public key chaining ultimately to the Root Public Key, a hardware protected value that resides inside the application processor. The TOE uses the SHA-256 hash of the Hauwei RSA public signing key that is in e-fuses within the application processor when verifying the TSF boot integrity hash and digital signatures on TSF updates. The Android OS on the TOE requires that all applications bear a valid signature before Android will install the application.
- ALC_TSU_EXT: Huawei provides its customers with the "HiCare" application, to report bugs. The 'Feedback' service in this application sends an email to Huawei reporting the bug. Huawei reviews all bug reports making product changes to resolve issues associated with Huawei developed code. Huawei makes updates and code patches to resolve issues as quickly as possible, and makes updates available to customers. The TOE checks for updates daily, performing updates as described above when new updates are available on the Huawei update servers. Issues associated with Android and reported directly to Google are handled through the Google product support process. Android security updates are sent by Google to all of their partners. Huawei receives these updates and incorporates them into their own android image which is then distributed as described above.

6.6 TOE access

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

- FTA_SSL_EXT.1: The TOE can become locked immediately after a User initiates an explicit lock action (i.e., pressing thee power button) or after a configurable period of inactivity. As part of the transition to a locked state, the TOE displays a lock screen that hides prior contents of the display. The TOE's lock screen displays email notifications, calendar appointments, user configured widgets, text message notifications, the time, date, call notifications, battery life, signal strength, and carrier network. In order for the user to perform any actions using these notifications, the user must authenticate. Only actions explicitly identified by FIA_UAU_EXT.2.1 (see section 5.1.3.7) are permitted prior to user login. During power up, the TOE presents the user with an initial power-up login screen (also known as the Cryptolock screen), where the user can only make an emergency call or enter the user's password in order to allow the TOE to derive the

[13] The 'fastboot' is based upon Google's Android fastboot with Huawei enhancements.

key needed to be able to access the data partition. After successfully authenticating at the power-up login screen, the TOE (when subsequently locked) presents the user with the normal user interface.

- 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 WiFi 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. The TOE permits itself and applications to initiate communicate via the trusted channel, and the TOE initiates communicate via the trusted channel for connection to a wireless access point. The TOE provides access to TLS via published APIs which are accessible to any application that needs an encrypted end-to-end trusted channel.

7. TSF Inventory

Below is a list of user-mode TSF binaries and libraries. 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
dalvikvm32	**system/bin**	**VM**
dalvikvm64	**system/bin**	**VM**
keystore	**system/bin**	**Key Store**
vold	**system/bin**	**DAR**
wpa_supplicant	**system/bin**	**DAR**
libcrypto.so	**system/lib**	**Crypto**
libkeystore_binder.so	**system/lib**	**Key Store**
libkeyutils.so	**system/lib**	**DAR**
libsoftkeymaster.so	**system/lib**	**Key Store**
libssl.so	**system/lib**	**SSL/TLS**
teecd	**sbin/teecd**	**trust zone daemon**
libHewdkmgr.so	**system/lib64**	**DAR**
libecryptfs.so	**system/lib64**	**DAR**
libkeyutils.so	**system/lib64**	**DAR**
keystore.hi3635.so	**system/lib64/hw/keystore.hi3635.so**	**Key Store**
libteec.so	**system/lib**	**trust zone client api**
libopenssl_selftest.so	**system/lib**	**selftest**
libbspatchhwouc.so	**system/lib**	**OTA**