The Boeing Company Boeing Black (MDFPP20) Security Target

Version 1.1
2015-03-02

Prepared for:

The Boeing Company

7700 Boston Boulevard, Springfield VA 22153

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 Boeing Black provided by The Boeing Company. The TOE is being evaluated as a mobile device.

The Security Target contains the following additional sections:

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

Acronyms and Terminology

AA	Assurance Activity
CC	Common Criteria
CCEVS	Common Criteria Evaluation and Validation Scheme
EAR	Entropy Analysis Report
GUI	Graphical User Interface
PCL	Product Compliant List
PP	Protection Profile
SAR	Security Assurance Requirement
SEC	Secure Embedded Component
SFR	Security Functional Requirement
SOF	Strength of Function
ST	Security Target
TOE	Target of Evaluation
U.S.	United States
VR	Validation Report

Conventions

The following conventions have been applied in this document:

- Security Functional Requirements – Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.
    - Iteration: allows a component to be used more than once with varying operations. In the ST, iteration is indicated by a letter placed at the end of the component. For example FDP_ACC.1a and FDP_ACC.1b indicate that the ST includes two iterations of the FDP_ACC.1 requirement, a and b.
    - Assignment: allows the specification of an identified parameter. Assignments are indicated using bold and are surrounded by brackets (e.g., [**assignment**]). Note that an assignment within a selection would be identified in italics and with embedded bold brackets (e.g., [***[selected-assignment]***]).
    - Selection: allows the specification of one or more elements from a list. Selections are indicated using bold italics and are surrounded by brackets (e.g., [***selection***]).
    - Refinement: allows the addition of details. Refinements are indicated using bold, for additions, and strike-through, for deletions (e.g., "… **all** objects …" or "… ~~some~~ **big** things …").

- The 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 – The Boeing Company Boeing Black (MDFPP20) Security Target

ST Version – Version 1.1

ST Date – 2015-03-02

1.2 TOE Reference

TOE Identification – The Boeing Company Boeing Black

TOE Developer – The Boeing Company

Evaluation Sponsor – The Boeing Company

1.3 TOE Overview

The Target of Evaluation (TOE) is the Boeing Black, a smartphone featuring a 4.3" qHD (540 x 960 pixels); Dual SIM supporting LTE, WCDMA, GSM; Bluetooth v2.1; Dual 1.2 GHz ARM Cortex-A9 CPUs; microSD; micro USB, PDMI, and a Modular 24-Pin Connector; and an embedded FIPS 140-2 level 3 module.

The Black provides telephony features (make and received phone calls, send and receive SMS messages), networking features (connect to Wi-Fi networks, send and receive MMS messages, connect to mobile data networks)

1.4 TOE Description

The TOE was designed with security and modularity in mind to ensure that users can employ the same smartphone across a range of missions and configurations. Boeing based the Black's PureSecure software (version 1.2.6) upon Android 4.1.2. With improvements by Boeing to provide enhanced software security policy configuration, the TOE provides uses a trusted, more flexible and productive solution.

The Black contains 32 GB of internal Flash, 1 GB of memory and supports microSD external storage.

1.4.1 TOE Architecture

The TOE provides a rich Application Programming Interface to mobile applications and provides users installing an application the ability 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 managed by the TOE. Moreover, the TOE provides users the ability to AES encrypt data and files stored on microSD Cards inserted into the device.

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

1.4.1.1 Physical Boundaries

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

1.4.1.2 Logical Boundaries

This section summarizes the security functions provided by Black:

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

1.4.1.2.1 Cryptographic support

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

1.4.1.2.2 User data protection

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

1.4.1.2.3 Identification and authentication

The TOE supports a number of features related to identification and authentication. From a user perspective, except for making phone calls to an emergency number, a password (i.e., Password Authentication Factor) must be correctly entered to unlock the TOE. Also, even when the TOE is unlocked the password must be re-entered to change the password. Passwords are obscured when entered so they cannot be read from the TOE's display. The TOE limits a user to five incorrect password attempts in thirty seconds and when a configured number of failures occurs, the TOE will be wiped to protect its contents. Passwords can be constructed using upper and lower cases characters, numbers, and special characters and passwords up to 22 characters are supported.

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

1.4.1.2.4 Security management

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

1.4.1.2.5 Protection of the TSF

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

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

1.4.1.2.6 TOE access

The TOE can be locked, obscuring its display, by the user or after a configured interval of inactivity. The TOE also has the capability to display a user specified message (banner) when the user unlocks the TOE.

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

1.4.1.2.7 Trusted path/channels

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

1.4.2 TOE Documentation

Boeing PureSecure Administrator Guidance v1.0.6 for PureSecure 1.3, March 2, 2015

2. Conformance Claims

This TOE is conformant to the following CC specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 3, July 2009.
    - Part 2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1 Revision 3, July 2009.
    - Part 3 Extended
- Protection Profile For Mobile Device Fundamentals, Version 2.0, 17 September 2014 (MDFPP20)
- Package Claims:
    - Assurance Level: EAL 1 augmented with ALC_TSU_EXT.1

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 Boeing Black 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_BBD_EXT.1: Application Processor Mediation
- FPT_KST_EXT.1: Extended: Key Storage
- FPT_KST_EXT.2: Extended: No Key Transmission
- FPT_KST_EXT.3: Extended: No Plaintext Key Export
- FPT_NOT_EXT.1: Extended: Self-Test Notification
- FPT_TST_EXT.1: Extended: TSF Cryptographic Functionality Testing
- FPT_TST_EXT.2: Extended: TSF Integrity Testing
- FPT_TUD_EXT.1: Extended: Trusted Update: TSF version query
- FPT_TUD_EXT.2: Extended: Trusted Update Verification
- FTA_SSL_EXT.1: Extended: TSF- and User-initiated locked state
- FTA_TAB.1: Default TOE Access Banners
- FTA_WSE_EXT.1: Extended: Wireless Network Access
- FTP_ITC_EXT.1: Extended: Trusted channel Communication

Extended SARs:

- ALC_TSU_EXT.1: Timely Security Updates

5. Security Requirements

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

The SFRs have all been drawn from the 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 the Boeing Black 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_BBD_EXT.1: Application Processor Mediation
	FPT_KST_EXT.1: Extended: Key Storage
	FPT_KST_EXT.2: Extended: No Key Transmission
	FPT_KST_EXT.3: Extended: No Plaintext Key Export
	FPT_NOT_EXT.1: Extended: Self-Test Notification
	FPT_STM.1: Reliable time stamps
	FPT_TST_EXT.1: Extended: TSF Cryptographic Functionality Testing
	FPT_TST_EXT.2: Extended: TSF Integrity Testing
	FPT_TUD_EXT.1: Extended: Trusted Update: TSF version query
	FPT_TUD_EXT.2: Extended: Trusted Update Verification
FTA: TOE access	FTA_SSL_EXT.1: Extended: TSF- and User-initiated locked state
	FTA_TAB.1: Default TOE Access Banners
	FTA_WSE_EXT.1: Extended: Wireless Network Access
FTP: Trusted path/channels	FTP_ITC_EXT.1: Extended: Trusted channel Communication

Table 1 TOE Security Functional Components

5.1.1 Cryptographic support (FCS)

5.1.1.1 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 FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Appendix B.3;,***
o ANSI X9.31-1998, Section 4.1;],
- [ECC schemes] using ['NIST curves' P-256, P-384 and [P-521]] that meet the following:

[FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Appendix B.4];,
- [FFC schemes] using cryptographic key sizes of [2048-bit or greater] that meet the following: [FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Appendix B.1]].

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

FCS_CKM.1(2).1
The TSF shall generate symmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm [PRF-384] and specified cryptographic key sizes [128 bits] using a Random Bit Generator as specified in FCS_RBG_EXT.1 that meet the following: [IEEE 802.11-2012].

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

FCS_CKM.2(1).1
The TSF shall perform cryptographic key establishment in accordance with a specified cryptographic key establishment method:
- [RSA-based key establishment schemes] that meets the following: [NIST Special Publication 800-56B, 'Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography'];
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 key of size [***256 bits***].
FCS_CKM_EXT.1.2
System software on the TSF shall be able only to request [***AES encryption/decryption***] 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.

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

FCS_CKM_EXT.2.1
All DEKs shall be randomly generated with entropy corresponding to the security strength of AES key sizes of [***128, 256***] bits.

5.1.1.7 Extended: Cryptographic Key Generation (FCS_CKM_EXT.3)

FCS_CKM_EXT.3.1

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

FCS_CKM_EXT.3.2

The TSF shall generate 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)***].

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:
o For volatile memory, the destruction shall be executed by a single direct overwrite [***consisting of zeroes***] following by a read-verify.
o For non-volatile EEPROM, the destruction shall be executed by a single direct overwrite consisting of a pseudo random pattern using the TSF's RBG (as specified in FCS_RBG_EXT.1), followed a read-verify.
o For non-volatile flash memory, the destruction shall be executed ***by a block erase followed by a read-verify***].
o 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.

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 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***] and message digest sizes 160 and [***256, 384***] that meet the following: [FIPS Pub 180-4].

5.1.1.13 Cryptographic operation (FCS_COP.1(3))

FCS_COP.1(3).1

The TSF shall perform [cryptographic signature services (generation and verification)] in accordance with a specified cryptographic algorithm
- [RSA schemes] using cryptographic key sizes [of 2048-bit or greater] that meet the following: [FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Section 4]
and [***- [ECDSA schemes] using ['NIST curves' P-256, P-384 and [P-521]] that meet the following: [FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Section 5];***].

5.1.1.14 Cryptographic operation (FCS_COP.1(4))

FCS_COP.1(4).1

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

5.1.1.15 Cryptographic operation (FCS_COP.1(5))

FCS_COP.1(5).1

The TSF shall perform [Password-based Key Derivation Functions] in accordance with a specified cryptographic algorithm [HMAC-[***SHA-256***]], with [**8192**] iterations, and output cryptographic key sizes [***256***] that meet the following: [NIST SP 800-132].

5.1.1.16 Extended: HTTPS Protocol (FCS_HTTPS_EXT.1)

FCS_HTTPS_EXT.1.1

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

FCS_HTTPS_EXT.1.2

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

FCS_HTTPS_EXT.1.3

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

5.1.1.17 Extended: Initialization Vector Generation (FCS_IV_EXT.1)

FCS_IV_EXT.1.1

The TSF shall generate IVs in accordance with 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)
- 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, software-based***] secure key storage for asymmetric private keys and [***symmetric keys, persistent secrets***].

FCS_STG_EXT.1.2

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

FCS_STG_EXT.1.3

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

FCS_STG_EXT.1.4

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

FCS_STG_EXT.1.5

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

5.1.1.21 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 AES in the [***CBC mode***].

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 [***- 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,
o TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,
o TLS_ECDHE_ECDSA_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,
o TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,
o TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,
o TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289]].

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

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

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

FCS_TLSC_EXT.2.5
The TSF shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [***secp256r1, secp384r1, secp521r1***] and no other curves.

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, groups of***

application processes] from accessing [***private***] data stored by other [***application processes, groups of application processes***]. Exceptions may only be explicitly authorized for such sharing by [***a common application developer***].

5.1.2.2 Extended: Protected Data Encryption (FDP_DAR_EXT.1)

FDP_DAR_EXT.1.1
Encryption shall cover all protected data.

FDP_DAR_EXT.1.2
Encryption shall be performed using DEKs with AES in the [***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 provides a means for non-TSF applications executing on the TOE to use TLS, HTTPS, Bluetooth BR/EDR, and [***no other protocol***] to provide a protected communication channel between the non-TSF application and another IT product that is logically distinct from other communication channels, provides assured identification of its end points, protects channel data from disclosure, and detects modification of the channel data.

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

5.1.3 Identification and authentication (FIA)

5.1.3.1 Authentication failure handling (FIA_AFL_EXT.1)

FIA_AFL_EXT.1.1
The TSF shall detect when a configurable positive integer within [**1-127**] 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 [**22**] 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 [***preventing authentication via an external port, enforcing a delay between incorrect authentication attempts***]. The minimum delay shall be such that no more than 10 attempts can be attempted per 500 milliseconds.

5.1.3.6 Protected authentication feedback (FIA_UAU.7)

FIA_UAU.7.1

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

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

FIA_UAU_EXT.1.1

The TSF shall require the user to present the Password Authentication Factor prior to decryption of protected data and encrypted DEKs, KEKs and [***long-term trusted channel key material, all software-based key storage***] 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, and configure sound/vibrate/mute]***] 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 [***no other conditions***].

5.1.3.10 Extended: Validation of certificates (FIA_X509_EXT.1)

FIA_X509_EXT.1.1

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

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

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 to the user.
15. enroll the TOE in management
[***4. enable/disable cellular modem, Wi-Fi,***
11. import keys/secrets into the secure key storage,
12. destroy imported keys/secrets and [no other secrets] in the secure key storage,
14. remove imported X.509v3 certificates and [no other X.509v3 certificates] in the Trust Anchor Database,
21. enable/disable display notification in the locked state of: [f. all notifications],
24. enable/disable developer modes,
26. enable removable media's data-at-rest protection,
29. approve [import] by applications of X.509v3 certificates in the Trust Anchor Database,
39. enable/disable [a. USB mass storage mode],
36. configure the unlock banner,
41. enable/disable [a. Hotspot functionality authenticated by [pre-shared key], b. USB tethering authenticated by [no authentication]],
45. enable/disable hotspot, usb tethering].

FMT_MOF_EXT.1.2

The TSF shall restrict the ability to perform the functions in column 5 of Table 1 to the administrator when the device is enrolled and according to the administrator-configured policy.
1. configure password policy: a. minimum password length, b. minimum password complexity, c. maximum password lifetime
2. configure session locking policy: a. screen-lock enabled/disabled, b. screen lock timeout, c. number of authentication failures
5. enable/disable camera, microphones: a. across device, [***c. no other method***]
10. configure application installation policy by [***a. restricting the sources of applications,***

b. specifying a set of allowed applications based on [application package name, application signature key] (an application whitelist),
c. denying installation of applications]
[***3. enable/disable the VPN protection: a. across device, [c. no other method],***
6. specify wireless networks (SSIDs) to which the TSF may connect,
7. configure security policy for each wireless network: a. [specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s)], b. security type, c. authentication protocol, d. client credentials to be used for authentication,
23. enable/disable [WiFi tethering (personal hotspot)]
25. enable data-at rest protection,
33. configure [certificate, public-key] used to validate digital signature on applications,
44. enable/disable location services: a. across device, [c. no other method]].

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:
1. configure password policy: a. minimum password length, b. minimum password complexity, c. maximum password lifetime
2. configure session locking policy: a. screen-lock enabled/disabled, b. screen lock timeout, c. number of authentication failures
3. enable/disable the VPN protection: a. across device, [***c. no other method***]
4. enable/disable cellular modem, GPS, Wi-Fi
5. enable/disable camera, microphones: a. across device, [***c. no other method***]
6. specify wireless networks (SSIDs) to which the TSF may connect
7. configure security policy for each wireless network: a. [***specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s)***], b. security type, c. authentication protocol, d. client credentials to be used for authentication
8. transition to the locked state
9. TSF wipe of protected data
10. configure application installation policy by [***a. restricting the sources of applications,***
b. specifying a set of allowed applications based on [application package name, application signature key] (an application whitelist),
c. denying installation of applications]
11. import keys/secrets into the secure key storage
12. destroy imported keys/secrets and [***no other keys/secrets***] in the secure key storage
13. import X.509v3 certificates into the Trust Anchor Database
14. remove imported X.509v3 certificates and [***no other X.509v3 certificates/***] in the Trust Anchor Database
15. enroll the TOE in management
16. remove applications
17. update system software
18. install applications
19. remove Enterprise applications
20. configure the Bluetooth trusted channel: a. disable/enable the Discoverable mode (for BR/EDR), b. change the Bluetooth device name,on
[***no other Bluetooth configuration***]
21. enable/disable display notification in the locked state of: [***f. all notifications***]
22. enable/disable all data signaling over usb, 1/8th jack microphone
44. enable/disable location services: a. across device, [***c. no other method***]
[***23. enable/disable [WiFi tethering (personal hotspot)],***
24. enable/disable developer modes,
25. enable data-at rest protection,
26. enable removable media's data-at-rest protection,
29. approve [import, removal] by applications of X.509v3 certificates in the Trust Anchor

Database,
33. configure [certificate, public-key] used to validate digital signature on applications,
36. configure the unlock banner,
39. enable/disable [a. USB mass storage mode,
b. USB data transfer without user authentication],
41. enable/disable [a. Hotspot functionality authenticated by [pre-shared key], b. USB tethering authenticated by [no authentication]],
45. enable/disable hotspot, usb tethering].

FMT_SMF_EXT.1.2

The TSF shall be capable of allowing the administrator to perform the following management functions:
1. configure password policy: a. minimum password length, b. minimum password complexity, c. maximum password lifetime
2. configure session locking policy: a. screen-lock enabled/disabled, b. screen lock timeout, c. number of authentication failures
5. enable/disable camera, microphones: a. across device, [***no other method***]
6. specify wireless networks (SSIDs) to which the TSF may connect
7. configure security policy for each wireless network: a. [***specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s)***], b. security type, c. authentication protocol, d. client credentials to be used for authentication
8. transition to the locked state
9. TSF wipe of protected data
10. configure application installation policy by [***a. restricting the sources of applications,***
b. specifying a set of allowed applications based on [application package name, application signature key]] (an application whitelist),
c. denying installation of applications]
13. import X.509v3 certificates into the Trust Anchor Database
16. remove applications
17. update system software
18. install applications
19. remove Enterprise applications
[***3. enable/disable the VPN protection: a. across device, [c. no other method],***
11. import keys/secrets into the secure key storage,
14. remove imported X.509v3 certificates and [no other X.509v3 certificates] in the Trust Anchor Database,
23. enable/disable [WiFi tethering (personal hotspot)],
25. enable data-at rest protection,
33. configure [certificate, public-key] used to validate digital signature on applications,
45. enable/disable hotspot, usb tethering].

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

FMT_SMF_EXT.2.1

The TSF shall offer [***lock the screen***] upon unenrollment and [***no other triggers***].

5.1.5 Protection of the TSF (FPT)

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

FPT_AEX_EXT.1.1

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

FPT_AEX_EXT.1.2

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

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

FPT_AEX_EXT.2.1

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

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

FPT_AEX_EXT.3.1

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

5.1.5.4 Extended: Domain Isolation (FPT_AEX_EXT.4)

FPT_AEX_EXT.4.1

The TSF shall protect itself from modification by untrusted subjects.

FPT_AEX_EXT.4.2

The TSF shall enforce isolation of address space between applications.

5.1.5.5 Application Processor Mediation (FPT_BBD_EXT.1)

FPT_BBD_EXT.1.1

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

5.1.5.6 Extended: Key Storage (FPT_KST_EXT.1)

FPT_KST_EXT.1.1

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

5.1.5.7 Extended: No Key Transmission (FPT_KST_EXT.2)

FPT_KST_EXT.2.1

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

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

FPT_KST_EXT.3.1

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

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

FPT_NOT_EXT.1.1

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

5.1.5.10 Reliable time stamps (FPT_STM.1)

FPT_STM.1.1

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

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

FPT_TST_EXT.1.1

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

5.1.5.12 Extended: TSF Integrity Testing (FPT_TST_EXT.2)

FPT_TST_EXT.2.1

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

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

FPT_TUD_EXT.1.1

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

FPT_TUD_EXT.1.2

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

FPT_TUD_EXT.1.3

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

5.1.5.14 Extended: Trusted Update Verification (FPT_TUD_EXT.2)

FPT_TUD_EXT.2.1

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

FPT_TUD_EXT.2.2

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

FPT_TUD_EXT.2.3

The TSF shall verify that the digital signature verification key used for TSF updates [***is validated to a public key in the Trust Anchor Database***].

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) [**lock the hardware keystore (prevent use of KEKs)**].

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

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

Table 2 EAL 1 augmented with ALC_TSU_EXT.1 Assurance Components

5.2.1 Development (ADV)

5.2.1.1 Basic functional specification (ADV_FSP.1)

ADV_FSP.1.1d

The developer shall provide a functional specification.

ADV_FSP.1.2d

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

ADV_FSP.1.1c

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

ADV_FSP.1.2c

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

ADV_FSP.1.3c

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

ADV_FSP.1.4c

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

ADV_FSP.1.1e

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

ADV_FSP.1.2e

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

5.2.2 Guidance documents (AGD)

5.2.2.1 Operational user guidance (AGD_OPE.1)

AGD_OPE.1.1d

The developer shall provide operational user guidance.

AGD_OPE.1.1c

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

AGD_OPE.1.2c

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

AGD_OPE.1.3c

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

AGD_OPE.1.4c

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

AGD_OPE.1.5c

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

AGD_OPE.1.6c

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

AGD_OPE.1.7c

The operational user guidance shall be clear and reasonable.

AGD_OPE.1.1e

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

5.2.2.2 Preparative procedures (AGD_PRE.1)

AGD_PRE.1.1d

The developer shall provide the TOE including its preparative procedures.

AGD_PRE.1.1c
The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.

AGD_PRE.1.2c
The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

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

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

5.2.3 Life-cycle support (ALC)

5.2.3.1 Labelling of the TOE (ALC_CMC.1)

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

ALC_CMC.1.1c
The TOE shall be labeled 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

FCS_CKM.1(1): The TOE provides asymmetric key generation for DH, RSA, and ECDH. The TOE generates RSA keys in its Secure Embedded Component (in accordance with FIPS 186-4) and can also generate RSA and DH/ECDH/ECDSA (including 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/ECDSA 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 Enterprise) and received a WiFi CERTIFIED Interoperability Certificate from the WiFi Alliance. The WiFi Alliance maintains a website providing further information about the testing program: http://www.wi-fi.org/certification

FCS_CKM.2(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 an internally-generated, hardware-protected, 256-bit REK within its Secure Embedded Component (SEC).

During the manufacturing provisioning process (as well as after a full wipe), the TOE commands the SEC to generate the REK. The SEC generates the REK using its SP 800-90A AES-256 CTR_DRBG (and it seeds that DRBG using an internal, hardware entropy source). The TOE sets permissions on the REK and configures the SEC such that afterwards, no operator can read, write, or delete it. An operator can only re-provision the device, which will completely destroy all keys. Only the Trusted Execution Environment that executes on the TOE may request the SEC to use the REK for encryption and decryption. Because the TOE stores the REK within the separate, secure hardware of the SEC (which does not physically share any memory with the Application or Baseband processor), no

application or process (Trusted Execution Environment [TEE], system-level, or otherwise) can ever access the value of the REK.

FCS_CKM_EXT.2: While the TOE can generate both 128-bit and 256-bit DEKs, the TOE itself only utilizes 256-bit DEKs. Mobile applications may request generation of either 128-bit or 256-bit DEKs. When generating DEKs, the TOE uses one of its approved RBGs depending upon how (which component of the TOE requires the DEK). For example, during mobile application API calls into the TOE's Android Framework the TOE uses its OpenSSL AES-256 CTR_DRBG and when generating the UDEK (User Data Encryption Key), SDEKs (SDcard Data Encryption Keys), and AMK (Android [keystore] Master Key) the TOE uses the SEC's AES-256 CTR_DRBG.

FCS_CKM_EXT.3: The TOE exclusively uses 256-bit AES KEKs to encrypt KEKs and DEKs. The TOE uses its SEC to generate all KEKs that encrypt DEKs. The TOE generates the KEKs by requesting data from the SEC's internal AES-256 CTR_DRBG and then creates KEKs from that data and writes it back into the SEC during the provisioning process. The TOE XORs the value derived from the user's password [derived using SP 800-108 as described in FCS_COP.1(5)] with a device specific value and AES encrypts the result with the REK and uses the resulting value as a SEC authentication credential to utilize the KEKs stored within the SEC. The TOE must do this, as the SEC provided hardware key store does not allow the TOE direct access (read, write, or delete) to any KEK values. Hence, the TOE cannot use the value derived from the user's password to encrypt any of its KEKs).

FCS_CKM_EXT.4: The TOE cryptographically erases all of its DEKs by requesting zeroization of the KEKs stored within the SEC.

FCS_CKM_EXT.5: The TOE cryptographically erases all protected data (which is stored Flash in the user data partition) by zeroizing the KEKs encrypting the DEKs, that in turn encrypt protected data. Upon completing such a zeroization (either at the explicit request of the user [factory Settings->Backup and reset->Factory data reset] or upon the user exceeding the number of incorrect login attempts), the TOE will power cycle itself.

FCS_CKM_EXT.6: The TOE generates all salts and nonces using one of its RBGs described in FCS_RBG_EXT.1.

FCS_COP.1: The TOE provides implementations of 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, GCM	FIPS 197, SP 800-38A/C/D/F	FCS_COP.1(1)	1884
CVL ECC CDH P-224/256/384/521	SP 800-56A	FCS_CKM.1(1)	10
DRBG Hash/HMAC/CTR	SP 800-90A	FCS_RBG_EXT.1(1)	157
ECDSA PKG/PKV/SigGen/SigVer	FIPS 186-4	FCS_CKM.1(1) FCS_CKM.1(2) FCS_COP.1(3)	264
HMAC SHA-1/256/384/512	FIPS 198-1 & 180-4	FCS_COP.1(4)	1126
RSA SIG(gen)/SIG(ver)/Key(gen)	FIPS 186-2, ANSI X9.31, SP 800-56B	FCS_CKM.1(1) FCS_CKM.1(2) FCS_COP.1(3)	960
SHS SHA-1/256/384/512	FIPS 180-4	FCS_COP.1(2)	1655

Table 3 OpenSSL Cryptographic Algorithms

The Linux kernel provides the following cryptographic algorithms

Algorithm	NIST Standard	SFR Reference	Cert#
AES 128/256 CBC	FIPS 197, SP 800-38A	FCS_COP.1(1)	3209
HMAC SHA-1/256	FIPS 198-1 & 180-4	FCS_COP.1(4)	2022
SHS SHA-1/256	FIPS 180-4	FCS_COP.1(2)	2656

Table 4 Kernel Cryptographic Algorithms

The TOE uses HMAC as part of the TLS ciphersuites. For TLS, the TOE uses HMAC using SHA-1 (with a 160-bit key) to generate a 160-bit MAC, using SHA-256 (with a 256-bit key) to generate a 256-bit MAC, and using SHA-

384 (with a 384-bit key) to generate a 384-bit MAC. FIPS 198-1 & 180-4 dictate the block size used, and they specify block sizes of 512, 512, and 1024-bits for HMAC-SHA-1, HMAC-SHA-256, and HMAC_SHA-384 respectively.

In addition to utilizing HMAC as part of TLS, the TOE conditions the user's password exactly as per SP 800-132 (and thus as per SP 800-197-1) with no deviations by using PBKDF2 with 8192 HMAC-SHA-256 iterations to combine a 256-bit salt with the user's password to derive a 256-bit key. The time needed to derive keying material does not matter as the combination of the password derived KEK with REK value entirely prevents offline attacks and the TOE's maximum incorrect login attempts prevents exhaustive online attacks.

FCS_HTTPS_EXT.1: The TOE supports the HTTPS protocol (compliant with RFC 2818) so that (mobile and system) applications executing on the TOE can act as HTTPS clients (the TOE cannot act as an HTTPS server) and securely connect to external servers using HTTPS.

FCS_IV_EXT.1: The TOE typically generates IVs using the AES-256 CTR_DRBG provided by its SEC. The TOE uses AES-CBC mode and generates the IVs in compliance with the requirements of table 14 of MDFPP20.

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

1. An AES-256 CTR_DRBG provided by the SEC.
2. RBGs provided by OpenSSL: Hash_DRBG (using any size SHA), HMAC_DRBG (again using size SHA), and CTR_DRBG (using AES). Note that while the TOE includes implementations of three DRBG variants (and supports all options within each variant), the TOE (and its current system level applications) makes use of only the AES-256 CTR_DRBG.

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

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

FCS_STG_EXT.1: The TOE provides both hardware and software-based key storage depending upon the specific type of key. The TOE stores the REK and all KEKs in its hardware keystore (implemented by the SEC). The TOE stores the user's RSA private keys, ECDSA private keys, and public keys within its software-based secure key storage (which is ultimately encrypted using the Android [keystore] Master Key). The TOE stores loaded CA certificates in its file system using file permissions to prevent any unauthorized modifications.

The user and administration can import (and subsequently use) asymmetric public and private keys into the TOE's Secure Key Storage (again, RSA and ECDSA private keys, public keys, and symmetric keys all reside in the software-based key store).

The user alone can request the TOE to destroy the keys stored in the TOE's Secure Key Storage. While normally mobile applications cannot import, use or destroy keys in the keystore, the user can import keys and certificates on behalf of an application (the application requests the system to prompt the user to import a key/certificate or grant approval for the application to utilize a user key within the keystore). Additionally, if two applications share a common application developer (and are thus signed by the same developer key) declare a shared UID, then user approval to utilize a user keystore key is shared by both applications..

Entity	Can import?	Can destroy?	Allow other app to use?	Allow other app to destroy
User	Yes	Yes	Yes	
Administrator	Yes			

Mobile application	Can Prompt User			
Common application developer			Yes	

FCS_STG_EXT.2: The TOE employs a key hierarchy that encrypts all DEKs with 256-bit KEKs. The TOE maintains three KEKs (in addition to the REK) within its hardware key store: the UDEKKEK, the SDEKKEK, and the AMK. The UDEKKEK encrypts the UDEK (a 256-bit DEK), which encrypts the user data partition. The AMK encrypts all Android keystore keys (including DEKs). The SDEKKEK encrypts SDEKs, which are KEKs stored on each SD Card inserted into the TOE. Each 256-bit SDEK (the TOE generates and writes an SDEK to each newly inserted SD Card) encrypts the FEK and FNEKs associated with external SD Card file encryption (and is stored within the header of each encrypted file residing on the SD Card).

By storing the UDEKKEK, SDEKKEK, and AMK within its hardware key store, the TOE ensures that an entity must have knowledge and authenticate using the password-derived KEK (which one must also wrap with the REK) before being able to request decryption of DEKs and KEKs encrypted with hardware key store KEKs. In this way, the TOE protects is KEKs with both a KEK derived from the user's password and the REK value (in that derivation of the authentication credential employs the REK).

The TOE always AES CBC encrypts DEKs using a 256-bit KEK, thus ensuring a commensurate or greater strength KEK for each DEK.

FCS_STG_EXT.3: The TOE protects the integrity of DEKs stored in Flash by verifying the SHA-256 hash of the decrypted key.

FCS_TLSC_EXT.1: The TSF supports TLS version 1.0 with client (mutual) authentication and supports the eight selected ciphersuites utilizing SHA (see the selections in section 5.1.1.23) for use with EAP-TLS as part of WPA2. The TOE does not support any other ciphersuites with EAP-TLS, as the remainder of selectable cipher suites are based on SHA256, which requires TLS v1.2. The TOE, by design, supports only evaluated elliptic curves (P-256, P-384, & P-521 and no others) and requires/allows no configuration of the supported curves.

FCS_TLSC_EXT.2: The TOE provides mobile applications (through its Android API) the use of TLS versions 1.0, 1.1, and 1.2 including support for all sixteen selectable ciphersuites (see the selections in section 5.1.1.24). The TOE supports Common Name (CN) and Subject Alternative Name (SAN) (DNS and IP address) as reference identifiers. The TOE supports client (mutual) authentication. The TOE, by design, supports only evaluated elliptic curves (P-256, P-384, & P-521 and no others) and requires/allows no configuration of the supported curves. While the TOE supports the use of wildcards in X.509 reference identifiers (CN and SAN), the TOE does not support certificate pinning.

6.2 User data protection

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)
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).

An example of a System or Signature permission is the `android.permission.LOCATION_HARDWARE`, which allows an application to use location features in hardware (such as the geofencing API). The device grants this permission to requesting applications that either have been signed with the same developer key used to sign the android application declaring the permissions or that reside in the "system" directory within Android, which for Android 5.0 and above, are applications residing in the /system/priv-app/ directory on the read-only system partition). Put another way, the device grants System or Signature permissions by Signature or by virtue of the requesting application being part of the "system image."

FDP_ACF_EXT.1: Android provides application separation using Linux/Unix user/group permissions. Every application (at the time of installation) receives a unique user and group ID, and Android, by default, sets the read/write permissions so as to prevent applications from accessing each other's data. Applications can actively choose to share data using one of several methods: Android's binder service, explicitly setting file permissions to be world readable, or through a shared UID method. These methods are all documented methods, and each one requires an active choice by the developer to enable sharing of data.

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

FDP_IFC_EXT.1: The TOE provides an interface to VPN clients (`android.net.VpnService`). The TOE provides an API to mobile applications to allow them to signal the TOE to alter its network routing (by inserting rules) to force all traffic the IPsec VPN client. Once can find more detail here:

http://developer.android.com/reference/android/net/VpnService.html

FDP_STG_EXT.1: The TOE's Trusted Anchor Database consists of the built-in certs (individually stored in /system/etc/security/cacerts) which the user can disable using the TOE's Android user interface [Settings->Security-> Trusted Credentials]) and any additional user or admin/MDM loaded certificates. Disabled default certificates and user added certificates reside in the /data/misc/keychain/cacerts-removed and /data/misc/keychain/cacerts-added directories respectively. The built-in ones are protected as they are part of the TSF's read only system partition, while the TOE protects user-loaded certs by storing them with appropriate permissions to prevent modification by mobile applications. The TOE also stores the user-loaded certificates in the user's keystore.

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. 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

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 login attempts to a value lower than the default of 10. Boeing can set the default value at compilation time (to a value between 1 and 127), but once compiled for a given build, the default cannot be changed.. The TOE maintains the number of failed logins across power-cycles (so for example, assuming a configured maximum retry of ten incorrect attempts, if one were to enter five incorrect passwords and power cycle the phone, the phone would only allow five more incorrect login attempts before wiping,

FIA_BLT_EXT.1: The TOE requires explicit user authorization before it will pair with a remote Bluetooth device. When pairing with another device, the TOE requires that the user either confirm that a displayed numeric passcode matches between the two devices or that the user enter (or choose) a numeric passcode that the peer device generates (or must enter).

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

FIA_PMG_EXT.1: The TOE authenticates user through a password consisting of basic Latin characters (upper and lower case, numbers, and the special characters noted in the selection (see section 5.1.3.4). The TOE defaults to requiring passwords to have a minimum of four characters and can support up to 22 characters. An MDM application can change these defaults and impose password restrictions (like quality, specify another minimum length, the minimum number of letters, numeric characters, lower case letters, upper case letters, symbols, a non-letters).

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 n^{th}] and prior four authentication attempts have failed, and the $n\text{-}4^{th}$ 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. The TOE will 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. The user can configure the TOE's behavior for the normal lockscreen so that it does not briefly display the last typed character; however, the TOE always briefly displays the last entered character for the crypt-lock screen.

FIA_UAU_EXT.1: As described before, the TOE's key hierarchy requires the user's password in order to derive the KEK that when decrypted with the REK serves as an authentication credential to utilize the KEKs stored in the TOEs hardware key store. Those KEKs, in turn, allow the TOE to decrypt its DEKs. Thus, until it has the user's

password, the TOE cannot decrypt the UDEK utilized for user data partition encryption, and thus cannot decrypt the user's protected data.

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, and configure sound/vibrate/mute. Beyond those actions, a user cannot perform any other actions other than observing notifications displayed on the lock screen until after successfully authenticating.

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

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

FIA_X509_EXT.2: The TOE uses X.509v3 certificates during EAP-TLS, TLS. The TOE comes with a built-in set of default of Trusted Credentials (as described in TSS section FDP_STG_EXT.1 above). And while the user cannot remove any of the built-in default CA certificates, the user can disable any of those certificates through the user interface and the TOE will trust certificates issued by disabled CA's as invalid. In addition, a user and an administrator/MDM can import a new trusted CA certificate into the Trust Anchor Database (the TOE stores the new CA certificate in the user's keychain).

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

FIA_X509_EXT.3: The TOE's Android operating system provides applications the `java.security.cert` Java API Class of methods validating certificates and certification paths (certificate chains establishing a trust chain from a certificate to a trust anchor). The available APIs may be found here:

http://developer.android.com/reference/java/security/cert/package-summary.html

6.4 Security management

FMT_MOF.1(1): The TOE grants the user the exclusive functions to enroll the TOE in a management (effectively granting Device Administration permissions to a mobile application), to enable and disable the cellular mode, GP, and Wi-Fi radios, to destroy keys/secrets in the secure key storage, update the system software and approve the import and removed of X509v3 certs.

FMT_MOF.1(2): When the TOE had enrolled in Device Management, the TOE grants the administrator the exclusive ability to perform the management functions specified in the SFR selections (see section 5.1.4.2).

FMT_SMF_EXT.1: The TOE provides the capability to perform all the management functions specified in the SFR selections (see section 5.1.4.3).

FMT_SMF_EXT.2: The TOE offers MDM agents the ability to lock the screen upon unenrollment.

The following table describes the TOE provided management functions, which role(s) can perform each function, and how these functions are (or can be) restricted to the identified roles:

Management Function **Status Markers:** I – Implemented by TOE	Function Implemented	Restricted to User role	Available to Admin role	Restricted to Admin
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.	I		I	I
2. configure session locking policy: a. screen-lock enabled/disabled b. screen lock timeout c. number of authentication failures The administrator can configure the session locking policy using the Android MDM APIs. The user can also adjust each of the session locking policies; however, if set by the administrator, the user can only set a more strict policy (e.g., setting the device to allow fewer authentication failures than configured by the administrator).	I		I	I
3. enable/disable the VPN protection: a. across device [c. *no other method*] The user can configure and then enable a VPN to protect the traffic. The TOE does not provide its own IPsec VPN, but instead relies upon 3rd party IPsec VPN. Any 3rd party IPsec VPN will provide the user the ability to enable and disable the VPN protection. The administrator (through an MDM Agent that calls the TOE's MDM APIs) can restrict the TOE's networking such that it may only connect to a specified IPsec VPN Gateway (and connect to no other combination of IP address, port, and protocol). By doing so, the administrator can require the user to connect via VPN; however the user must specify/configure the IPsec VPN details using the 3rd party IPsec VPN mobile application.	I			I
4. enable/disable [*cellular modem, Wi-Fi*] The user can enable and disable the TOE's cellular modem through the TOE's user interface (Settings->Airplane mode). Additionally the user can turn off the TOE's Wi-Fi radio through the user interface (Settings->Wi-Fi).	I	I		
5. enable/disable [*camera, microphones*]: a. across device [c. *no other method*]	I		I	I

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.				
6. specify wireless networks (SSIDs) to which the TSF may connect An administrator can specify a list of wireless network SSIDs to which the TOE may connect and can restrict the TOE to only allow a connection to the specified SSIDs.	I		I	I
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 In addition to specifying the SSIDs to which the TOE may connect, an administrator can also configure the above details of a Wi-Fi connection to the TOE and restrict the TOE to only connect to such configured Wi-Fi connections. When the administrator configures the CA and client credentials (again, through an MDM Agent calling the TOE's MDM APIs), the user must acknowledge and import the X.509v3 CA certificate and their client certificate (which requires the user to enter the corresponding PFX/PKCS#12 password).	I		I	I
8. transition to the locked state An administrator may (through an MDM API) cause the TOE to transition to the locked state. Once locked, the user can enter their password to unlock the TOE.	I		I	
9. full wipe of protected data An administrator (through an MDM API) cause the TOE to wipe its data, returning the TOE to a pristine state.	I		I	
10. configure application installation policy by [selection: *a. specifying authorized application repository(s),* *b. specifying a set of allowed applications based on [application package name, application signature key] (an application whitelist),* c. *denying installation of applications*] An administrator can create application whitelists and blacklists that specify either application package names or application signature keys so as to restrict the TOE's ability to install applications. Additionally, an administrator can block application repositories by blocking specific IP addresses associated with the repositories.	I		I	I
11. import keys/secrets into the secure key storage The user can import keys and certificates into the secure key storage using the TOE's UI (Settings->Security->Credential storage->Install from phone storage).	I	I		
12. destroy imported keys/secrets and [*no other keys/secrets*] in the secure key storage The user can destroy all keys and certificates present in the secure key storage using the TOE's UI (Settings->Security->Credential storage->Clear credentials). Note that the TOE only allows the user to clear the entire contents of the user's secure key storage, and does not afford the user the ability to selectively destroy specific keys/secrets.	I	I		
13. import X.509v3 certificates into the Trust Anchor Database The administrator can (through the use of an MDM Agent utilizing the TOE's MDM APIs) specify an X.509v3 certificate to import into the TOE's TAD. Note that the TOE will prompt	I	-	I	

the user to accept and import the CA certificate into the TAD. The user can also import keys and certificates into the secure key storage using the TOE's UI (see above).				
14. remove imported X.509v3 certificates and [*no other X.509v3 certificates*] in the Trust Anchor Database The user, using the TOE's UI, can remove imported X.509v3 certificates in two ways. First, the user can selectively remove a specific imported CA certificate (Settings->Security->Credential storage->Trusted credentials->User [tab]->[select CAcert]->[Scroll down and select the "Remove" button]). Second, by clearing the secure key store (Settings->Security->Credential storage->Clear credentials), the user can remove all imported CA certificates; however, this action also removes all user imported keys/secrets as well.	I	I		
15. enroll the TOE in management The user can enroll the TOE in management with either the TOE's built-in MDM Agent or by using a third-party MDM Agent. The user can enroll the TOE using the built-in by using the TOE's UI (Settings->PureSecure Settings). Alternatively, the user can load, install, and enroll the third-party MDM Agent (though the steps beyond installation are specific to the third-party Agent).	I	I		
16. remove applications An administrator can specify (through an MDM Agent utilizing the TOE MDM APIs) an application package name to remove. The TOE will then uninstall the mobile application if present.	I	-	I	
17. update system software The administrator updates the TOE's software by inserting an SD card containing an signed update package into the TOE and then rebooting the TOE to the recovery mode. Once in recovery mode, the administrator can direct the TOE to verify and install the updated from SD Card.	I	-	I	
18. install applications An administrator can specify an application package name to be installed on the TOE.	I	-	I	
19. remove Enterprise applications An administrator can specify (through an MDM Agent utilizing the TOE MDM APIs) an application package name to remove. The TOE will then uninstall the mobile application if present.	I	-	I	
20. configure the Bluetooth trusted channel: a. disable/enable the Discoverable mode (for BR/EDR) b. change the Bluetooth device name [i. *no other Bluetooth configuration*] The user can configure the TOE's Bluetooth configuration through the UI (Settings->Network connections->Bluetooth) and configure whether the TOE is discoverable (and how long it will remain discoverable [2 minutes, 5 minutes, 1 hour, or indefinitely]) as well as the device's Bluetooth name.	I	I		
21. enable/disable display notification in the locked state of: [selection: f. *all notifications*] While the TOE does support notifications while in the locked state, the TOE does not display any content for a given notification. The TOE only displays a notification icon (for example, an letter icon denoting a new email).	I	I		
22. enable/disable all data signaling over [assignment: *list of externally accessible hardware*				

ports]				
23. enable/disable [*WiiFi tethering (personal hotspot)*] The TOE allows an administrator to restrict (through an MDM Agent utilizing the TOE's MDM APIs) the user's ability to share the TOE's mobile data connection with other devices via a Wi-Fi hotspot. If the administrator had not restricted the user's ability to do so, then the user can configure the TOE to act as a Wi-Fi hotspot using the TOE's UI (Settings->Network connections->Tethering and portable hotspot->Mobile Hotspot).	I	-		I
24. enable/disable developer modes A user can configure the TOE to enable or disable USB debugging through the TOE's UI. The user must first enable the TOE's developer mode (Settings->About device-> Build number [tap this item seven times]) and can then enable USB debugging (Settings->Developer options->Debugging->USB debugging). An administrator can disable TOE's USB debugging capability (through an MDM Agent utilizing the TOE's MDM APIs), and thus prevent a user from enabling USB Debugging thereafter.	I	-	I	I
25. enable data-at rest protection The TOE allows an administrator to specify device encryption, which will force the TOE to encrypt its user data partition on its internal eMMC/Flash.	I	-	I	I
26. enable removable media's data-at-rest protection The User can enable TOE encryption of the files on SD card through the TOE's UI (Settings->Security->Encryption->Encryption external SD card).	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 The TOE prompts the user whenever a mobile application wishes to import an X.509v3 certificate into the TOE's trusted credentials. The user must agree to the request before the TOE will import the certificate.	I	I		
30. configure whether to establish a trusted channel or disallow establishment if the TSF cannot establish a connection to determine the validity of a certificate				
31. enable/disable the cellular protocols used to connect to cellular network base stations				
32. read audit logs kept by the TSF				
33. configure [*certificate, public-key*] used to validate digital signatures on applications The administrator (through an MDM Agent utilizing the TOE's MDM APIs) can specify a certificate or certificates that mobile applications must be signed with else the TOE will reject installation.	I		I	I
34. approve exceptions for shared use of keys/secrets by multiple applications				
35. approve exceptions for destruction of keys/secrets by applications that did not import the key/secret				
36. configure the unlock banner The TOE allows the user to configure an unlock banner through the TOE's UI (Settings->Lock screen->Owner information). The TOE will display the configured information while in the locked state.	I	I		
37. configure the auditable items				
38. retrieve TSF-software integrity verification values				
39. enable/disable [a. *USB mass storage mode*]	I	I		

An user can configure the TOE to operate as a mass storage device by connecting the TOE to a PC via USB, then swiping down from the top of the screen to "pull down" the Notifications shade, then selecting the notification showing a USB icon and stating "Connected as a media device/camera, Touch for other USB options", and finally choosing whether the TOE should connect as a Media device (MTP) or Camera (PTP).				
40. enable/disable backup to [*locally connected system, remote system*]				
41. enable/disable [a. *Hotspot functionality authenticated by [pre-shared key]*, b. *USB tethering authenticated by [no authentication]*] The user can enable and disable Wi-Fi Hotspot and USB tethering functionality through the TOE's UI (Settings->Network connections->Tethering and portable hotspot->Mobile Hotspot and also USB tethering).	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*] The TOE allows an administrator (though an MDM Agent utilizing the TOE's MDM APIs) to restrict the TOE's use of location services.	I	-		I
45. [*enable/disable hotspot, usb tethering*] The user can enable and disable Wi-Fi Hotspot and USB tethering functionality through the TOE's UI (Settings->Network connections->Tethering and portable hotspot->Mobile Hotspot and also USB tethering).	I	I		

6.5 Protection of the TSF

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

FPT_AEX_EXT.2: The TOE's Android 4.1 operating system utilizes a 3.0.31 based Linux kernel (with many security backports), 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 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://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0406c/index.html

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

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

The TOE protects its REK by requesting DRBG output from the SEC and then immediately storing the value as the REK in the SEC (its hardware keystore).

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

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

FPT_KST_EXT.2: The TOE itself (i.e., the mobile device) comprises a cryptographic module that utilizes a hardware keystore (with its Secure Embedded Component) in addition to cryptographic libraries including OpenSSL, kernel cryptography, 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 first by not allowing the REK or any KEKs to leave the hardware key store and by ensuring that only authenticated entities can request the SEC to utilize a KEK for a cryptographic operation and only allowing the TEE to use the REK. Furthermore, the TOE only allows the system-level executables access to plaintext DEK values needed for their operation. The TSF software (the system-level executables) protects those plaintext DEK values in memory both by not providing any access to these values and by clearing them when no longer needed (in compliance with FCS_CKM.4).

FPT_KST_EXT.3: The TOE does not provide any way to export plaintext DEKs or KEKs as the TOE stores the REK and all KEKs in the hardware key store of its SEC. The TOE encrypts all DEKs with a KEK stored within the SEC.

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 TLS 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 automatically performs known answer power on self-tests (POST) on its cryptographic algorithms to ensure that they are functioning correctly. Each component providing cryptography (the SEC, kernel, and OpenSSL library) perform known answer tests on their cryptographic algorithms to ensure they are working correctly.

Should any of the tests fail, the TOE reboots and displays a screen indicating a hardware failure, at which point a user can attempt to power cycle the phone to see if that will clear the error.

Algorithm	Implemented in	Description

Algorithm	Implemented in	Description
AES encryption/decryption	Secure Component	Comparison of known answer to calculated valued
DRBG random bit generation	Secure Component	Comparison of known answer to calculated valued
ECDSA sign/verify	Secure Component	Comparison of known answer to calculated valued
HMAC-SHA	Secure Component	Comparison of known answer to calculated valued
RSA sign/verify	Secure Component	Comparison of known answer to calculated valued
SHA hashing	Secure Component	Comparison of known answer to calculated valued
AES encryption/decryption	OpenSSL	Comparison of known answer to calculated valued
ECDH key agreement	OpenSSL	Comparison of known answer to calculated valued
DRBG random bit generation	OpenSSL	Comparison of known answer to calculated valued
ECDSA sign/verify	OpenSSL	Comparison of known answer to calculated valued
HMAC-SHA	OpenSSL	Comparison of known answer to calculated valued
RSA sign/verify	OpenSSL	Comparison of known answer to calculated valued
SHA hashing	OpenSSL	Comparison of known answer to calculated valued
AES encryption/decryption	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

Table 5 Power-up Cryptographic Algorithm Known Answer Tests

FPT_TST_EXT.2: The TOE ensures a secure boot process in which the TOE verifies the digital signature of the bootloader software for the Application Processor before transferring control. The bootloader, in turn, verifies the signature of the Linux kernel (either the primary or the recovery kernel) it loads and an early step in the system image verifies the file hashes of the system image executables.

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

FPT_TUD_EXT.2: 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 Android OS on the TOE requires that all applications bear a valid signature before Android will install the application.

ALC_TSU_EXT.1: Boeing provides its customers with a mechanism to report any potential issues. Boeing analyzes the feedback from its customers, and takes action appropriate depending upon where the bug may lie (whether in code completely maintained by Boeing or working with Google for Android code issues). Boeing creates updates and patches to revolved reported issues as quickly as possible, and makes these updates available to the wireless carriers fielding Boeing phones. Issues reported to Google directly are handled through Google's notification processes.

6.6 TOE access

FTA_SSL_EXT.1: The TOE transitions to its locked state either immediately after a User initiates a lock by pressing the power button (if so configured) or after a (also configurable) period of inactivity, and as part of that transition, the TOE will display a lock screen to obscure the previous contents; however, the TOE's lock screen still displays notifications icons to indicate new emails, calendar appointments, text messages, and calls. The TOE's lock screen also shows the time, date, battery life, signal strength, and carrier network. But without authenticating first, a user cannot perform any related actions based upon these notifications (they cannot view or respond to emails, calendar appointments, calls, or text messages) other than the actions assigned in FIA_UAU_EXT.2.1 (see section 5.1.3.8).

Note that during power up, the TOE presents the user with an initial power-up user data partition encryption lock screen, where the user can only make an emergency call or enter the user password in order to allow the TOE to

decrypt the user data partition key so as to be able to access the data partition. After successfully authenticating at the power-up login screen, the TOE presents the user with the lock screen.

FTA_TAB.1: The TOE can be configured by a user to display a Lock screen message (Settings->Security->Owner info). Additionally, an Administrator can configure the TOE to display a banner on the lock screen.

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

6.7 Trusted path/channels

FTP_ITC_EXT.1: The TOE provides secured (encrypted and mutually authenticated) communication channels between itself and other trusted IT products through the use of 802.11-2012, 802.1X, and EAP-TLS, 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 that are built with the -fstack-protector option set. For each binary/library, the name, path and security function is provided.

Name	Path	Securuty Function
dalvikvm	/system/bin	Virtual Machine
keystore	/system/bin	Key Store
rngd	/system/bin	RNG
time_daemon	/system/bin	Time
vold	/system/bin	User data encryption
wpa_supplicant	/system/bin	WPA2 w/ EAP-TLS
libcrypto.so	/system/lib	Crypto
libfotajni.so	/system/lib	FOTA JNI
libjavacrypto.so	/system/lib	Crypto JNI
libkeystore_binder.so	/system/lib	KeyStore
libkeyutils.so	/system/lib	DAR
libecryptfs.so	/system/lib	SD Card encryption
libsoftkeymaster.so	/system/lib	Key Store
libssl.so	/system/lib	SSL/TLS