Samsung Electronics Co., Ltd.
Samsung Galaxy Devices on Android 8 (MDFPP31/WLANCEP10/VPNC21)
Security Target

Version 0.4
2018/05/15

Prepared for:

Samsung Electronics Co., Ltd.

416 Maetan-3dong, Yeongtong-gu, Suwon-si, Gyeonggi-do, 443-742 Korea

Prepared By:

www.gossamersec.com

LIST OF TABLES

1. Security Target Introduction

This section identifies the Security Target (ST) and Target of Evaluation (TOE) identification, ST conventions, ST conformance claims, and the ST organization. The TOE consists of the Samsung Galaxy Devices on Android 8 provided by Samsung Electronics Co., Ltd.. The TOE is being evaluated as a Mobile Device.

The Security Target contains the following additional sections:

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

Acronyms and Terminology

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

Conventions

The following conventions have been applied in this document:

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

- Refinement: allows the addition of details. Refinements are indicated using bold, for additions, and strike-through, for deletions (e.g., "… **all** objects …" or "… ~~some~~ **big** things …").

- Other sections of the ST – Other sections of the ST use bolding to highlight text of special interest, such as captions.

1.1 Security Target Reference

ST Title – Samsung Electronics Co., Ltd. Samsung Galaxy Devices on Android 8 (MDFPP31/WLANCEP10/VPNC21) Security Target

ST Version – Version 0.4

ST Date – 2018/05/15

1.2 TOE Reference

TOE Identification – Samsung Electronics Co., Ltd. Samsung Galaxy Devices on Android 8.

TOE Developer – Samsung Electronics Co., Ltd.

Evaluation Sponsor – Samsung Electronics Co., Ltd.

1.3 TOE Overview

The Target of Evaluation (TOE) are the Samsung Galaxy Devices on Android 8.

1.4 TOE Description

The TOE is a mobile device based on Android 8 with a built-in IPsec VPN client and modifications made to increase the level of security provided to end users and enterprises. The TOE is intended for use as part of an enterprise mobility solution providing mobile staff with enterprise connectivity.

The TOE includes a Common Criteria mode (or "CC mode") that an administrator can invoke using an MDM. The TOE must meet the following prerequisites in order for an administrator to transition the TOE to and remain in the CC configuration.

- Require a screen lock password (swipe, PIN, pattern, accessibility (direction), or facial recognition screen locks are not allowed).
- The maximum password failure retry policy should be less than or equal to 50.
- A screen lock password required to decrypt data on boot.
- Revocation checking must be enabled.
- External storage must be encrypted.
- Password (non-container) recovery policy and password history must not be enabled.

When CC mode has been enabled, the TOE behaves as follows:

- The TOE sets the system wide Android CC mode property to enabled.
- The TOE prevents loading of custom firmware/kernels and requires all updates occur through FOTA (Samsung's Firmware Over The Air firmware update method).
- The TOE utilizes CAVP approved cryptographic ciphers for TLS.
- The TOE ensures FOTA updates utilize 2048-bit PKCS #1 RSA-PSS formatted signatures (with SHA-512 hashing).

The TOE includes a containerization capability, Knox Workspace Container, which is part of the Knox Platform. This container provides a way to segment applications and data into two separate areas on the device, such as a personal area and a work area, each with its own separate apps, data and security policies. For this effort the TOE was evaluated both without and with a Knox Workspace container created (and to create a Knox Workspace container, one must purchase an additional license). Thus, the evaluation includes several Knox-specific claims that apply to a Knox Workspace container when created.

There are different models of the TOE, the Samsung Galaxy Devices on Android 8, and these models differ in their internal components (as described in the table below).

The model numbers of the mobile devices used during evaluation testing are as follows:

Device Name	Model Number	Chipset Vendor	CPU	Build Arch/ISA	Android Version	Kernel Version	Build Number
Galaxy S9+	SM-G965F	Samsung	Exynos 9810	A64	8.0	4.9.65	R16NW
Galaxy S9+	SM-G965U	Qualcomm	SDM845	A64	8.0	4.9.59	R16NW
Galaxy S8	SM-G950F	Samsung	Exynos 8895	A64	8.0	4.4.13	R16NW
Galaxy S8+	SM-G955U	Qualcomm	MSM8998	A64	8.0	4.4.78	R16NW

Table 1 Evaluated Devices

In addition to the evaluated devices, the following device models are claimed as equivalent with a note about the differences between the evaluated device and the equivalent models.

Evaluated Device	CPU	Equivalent Devices	Differences
Galaxy S9+ (Qualcomm)	SDM845	Galaxy S9 (Qualcomm)	S9+ is larger
Galaxy S9+ (Samsung)	Exynos 9810	Galaxy S9 (Samsung)	S9+ is larger
Galaxy S8+ (Qualcomm)	MSM8998	Galaxy S8 (Qualcomm)	S8+ is larger
		Galaxy Note8 (Qualcomm)	Note8 includes S Pen & functionality to take advantage of it for input (not security related)
		Galaxy S8 Active	S8+ is larger S8 Active has a IP68 & MIL-STD-810G certified body
Galaxy S8 (Samsung)	Exynos 8895	Galaxy S8+ (Samsung)	S8+ is larger
		Galaxy Note8 (Samsung)	Note8 is larger Note8 includes S-Pen

Table 2 Equivalent Devices

In general, the devices include a final letter or number at the end of the name that denotes that the device is for a specific carrier or region (for example, U = US Carrier build and F = International, which were used during the evaluation). The following list of letters/numbers denotes the specific models that may be validated:

J – KDDI,
D – NTT Docomo,
U – All US Carriers (unified US model),
N – All Korean Carriers (unified Korean model),
F/C/I – International

For each device there are specific models which are validated. This table lists the specific carrier models that have the validated configuration (covering both evaluated and equivalent devices).

Device Name	Base Model Number	Carrier Models
Galaxy S9 (Qualcomm)	SM-G960	U, SC-02K*, SCV38*
Galaxy S9 (Samsung)	SM-G960	N, F
Galaxy S9+ (Qualcomm)	SM-G965	U
Galaxy S9+ (Samsung)	SM-G965	N, F
Galaxy Note8 (Qualcomm)	SM-N950	U, SC-01K*, SCV37*
Galaxy Note8 (Samsung)	SM-N950	N, F
Galaxy S8 (Qualcomm)	SM-G950	U
Galaxy S8 (Samsung)	SM-G950	N, F
Galaxy S8+ (Qualcomm)	SM-G955	U
Galaxy S8+ (Samsung)	SM-G955	N, F
Galaxy S8 Active	SM-G892	A, U, None

Table 3 Carrier Models

The carrier models marked by * are explicit model numbers for those carriers and do not follow the standard specified for other models. Where Carrier Models specifies "None" that means a device without a suffix is also a device that can be placed into a validated configuration.

1.4.1 TOE Architecture

The TOE combines with a Mobile Device Management solution (note that this evaluation does not include an MDM agent nor server) that enables the Enterprise to watch, control and administer all deployed mobile devices, across multiple mobile service providers as well as facilitate secure communications through a VPN. This partnership provides a secure mobile environment that can be managed and controlled by the environment and reduces the risks that can be introduced through a Bring-Your-Own-Device (BYOD) model which can be extended to Corporate-Owned-Personally-Enabled (COPE) or other corporate-owned deployments.

Data on the TOE is protected through the implementation of Samsung On-Device Encryption (ODE) that utilizes a CAVP certified cryptographic algorithms to encrypt device storage. This functionality is combined with a number of on-device policies including local wipe, remote wipe, password complexity, automatic lock and privileged access to security configurations to prevent unauthorized access to the device and stored data.

The Samsung Knox Software Development Kit (SDK) builds on top of the existing Android security model by expanding the current set of security configuration options to more than 600 configurable policies and including additional security functionality such as application whitelisting and blacklisting.

The Knox Platform for Enterprise provides a set of flexible deployment options for Work environments, including the ability to enhance the BYOD or COPE models by creating a separate container for the Enterprise (the Workspace). Within the Knox Workspace, the Enterprise can provision separate applications and ensure they are kept separate from anything the user may do outside the Knox Workspace. The Enterprise can use policy controls to manage a Work environment on the device as a whole or within the Knox Workspace container specifically, as needed by the organization.

1.4.1.1 Physical Boundaries

The TOE is a multi-user mobile device based on Android 8 that incorporates the Samsung Knox SDK. The TOE does not include the user applications that run on top of the operating system, but does include controls that limit application behavior. The TOE includes an IPsec VPN client integrated into the firmware (as opposed to a downloadable application). Within an Enterprise environment, the Enterprise can manage the configuration of the mobile device, including the VPN client, through a compliant device management solution.

The TOE communicates and interacts with 802.11-2012 Access Points and mobile data networks to establish network connectivity, and the through that connectivity interacts with MDM servers that allow administrative control of the TOE.

1.4.1.2 Logical Boundaries

This section summarizes the security functions provided by the Samsung Galaxy Devices on Android 8:

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

1.4.1.2.1 Security audit

The TOE generates logs for a range of security relevant events. The TOE stores the logs locally so they can be accessed by an administrator or they can be exported to an MDM.

1.4.1.2.2 Cryptographic support

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

1.4.1.2.3 User data protection

The TOE controls access to system services by hosted applications, including protection of the Trust Anchor Database. Additionally, the TOE protects user and other sensitive data using encryption so that even if a device is physically lost, the data remains protected. The functionality provided by a Knox Workspace container enhances the security of user data by providing an additional layer of separation between different categories of apps and data while the device is in use. The TOE ensures that residual information is protected from potential reuse in accessible objects such as network packets.

1.4.1.2.4 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 or Biometric Authentication Factor (BAF) must be correctly entered to unlock the TOE. In addition, even when the TOE is unlocked the password must be re-entered to change the password or re-enroll the biometric template. Passwords are obscured when entered so they cannot be read from the TOE's display, the frequency of entering passwords is limited and when a configured number of failures occurs, the TOE will be wiped to protect its contents. Passwords can be constructed using upper and lower casecharacters, numbers, and special characters and passwords between 4 and 16 characters are supported.

The TOE can also serve as an 802.1X supplicant and can use X.509v3 and validate certificates for EAP-TLS, TLS and IPsec exchanges. The TOE can also act as a client or server in an authenticated Bluetooth pairing. In addition to storing X.509 certificates used for IPsec connections, the TOE can also securely store pre-shared keys for VPN connections.

1.4.1.2.5 Security management

The TOE provides all the interfaces necessary to manage the security functions (including the VPN client) 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 removes all MDM policies and disables CC mode.

1.4.1.2.6 Protection of the TSF

The TOE implements a number of features to protect itself to ensure the reliability and integrity of its security features. It protects particularly sensitive data such as cryptographic keys so that they are not accessible or exportable. It 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

also protects itself from modification by applications as well as isolates 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.

1.4.1.2.7 TOE access

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

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

1.4.1.2.8 Trusted path/channels

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

1.4.2 TOE Documentation

- Samsung Android 8 on Galaxy Devices Administrator Guide, version 4.0, May 15, 2018
- Samsung VPN Client on Galaxy Devices Administrator Guide, version 4.0, March 28, 2018

2. Conformance Claims

This TOE is conformant to the following CC specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 4, September 2012.
    - Part 2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1 Revision 4, September 2012.
    - Part 3 Conformant
- Package Claims:
    - Protection Profile for Mobile Device Fundamentals, Version 3.1, 16 June 2017, General Purpose Operating Systems Protection Profile/Mobile Device Fundamentals Protection Profile Extended Package (EP) Wireless Local Area Network (WLAN) Clients, Version 1.0, 08 February 2016 and PP-Module for Virtual Private Network (VPN) Clients, Version 2.1, 05 October 2017 (MDFPP31/WLANCEP10/VPNC21)
- Technical Decisions:
    - Applicable NIAP Technical decisions: TD0194, TD0244, TD0301, TD0303, TD0304, TD0305.

2.1 Conformance Rationale

The ST conforms to the MDFPP31/WLANCEP10/VPNC21. As explained previously, the security problem definition, security objectives, and security requirements are defined in the PP.

3. Security Objectives

The Security Problem Definition may be found in the MDFPP31/WLANCEP10/VPNC21 and this section reproduces only the corresponding Security Objectives for operational environment for reader convenience. The MDFPP31/WLANCEP10/VPNC21 offers additional information about the identified security objectives, but that has not been reproduced here and the MDFPP31/WLANCEP10/VPNC21 should be consulted if there is interest in that material.

In general, the MDFPP31/WLANCEP10/VPNC21 has defined Security Objectives appropriate for Mobile Devices and as such are applicable to the Samsung Galaxy Devices on Android 8 TOE.

3.1 Security Objectives for the Operational Environment

- **OE.CONFIG** TOE administrators will configure the Mobile Device security functions correctly to create the intended security policy.
- **OE.NO_TOE_BYPASS** (WLANCEP10) Information cannot flow between external and internal networks located in different enclaves without passing through the TOE.
- **OE.NOTIFY** The Mobile User will immediately notify the administrator if the Mobile Device is lost or stolen.
- **OE.PRECAUTION** The Mobile User exercises precautions to reduce the risk of loss or theft of the Mobile Device.
- **OE.TRUSTED_ADMIN** TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.
- **OE.NO_TOE_BYPASS** (VPNC21) Information cannot flow onto the network to which the VPN client's host is connected without passing through the TOE.
- **OE.PHYSICAL** Physical security, commensurate with the value of the TOE and the data it contains, is assumed to be provided by the environment.
- **OE.TRUSTED_CONFIG** Personnel configuring the TOE and its operational environment will follow the applicable security configuration guidance.

4. Extended Components Definition

All of the extended requirements in this ST have been drawn from the MDFPP31/WLANCEP10/VPNC21. The MDFPP31/WLANCEP10/VPNC21 defines the following extended SFRs and SARs and since they are not redefined in this ST the MDFPP31/WLANCEP10/VPNC21 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_IPSEC_EXT.1: Extended: IPsec
- FCS_IV_EXT.1: Extended: Initialization Vector Generation
- FCS_RBG_EXT.1: Extended: Cryptographic Operation (Random Bit Generation)
- FCS_RBG_EXT.2: Extended: Cryptographic Operation (Random Bit Generation)
- FCS_SRV_EXT.1: Extended: Cryptographic Algorithm Services
- FCS_STG_EXT.1: Extended: Cryptographic Key Storage
- FCS_STG_EXT.2: Extended: Encrypted Cryptographic Key Storage
- FCS_STG_EXT.3: Extended: Integrity of encrypted key storage
- FCS_TLSC_EXT.1: Extended: TLS Protocol
- FCS_TLSC_EXT.1/WLAN: Extended: Extensible Authentication Protocol-Transport Layer Security - WLAN
- FCS_TLSC_EXT.2/WLAN: Extended: TLS Client Protocol - WLAN
- FDP_ACF_EXT.1: Extended: Security access control
- FDP_DAR_EXT.1: Extended: Protected Data Encryption
- FDP_DAR_EXT.2: Extended: Sensitive Data Encryption
- FDP_IFC_EXT.1: Extended: Subset information flow control
- FDP_STG_EXT.1: Extended: User Data Storage
- FDP_UPC_EXT.1: Extended: Inter-TSF user data transfer protection
- FIA_AFL_EXT.1: Extended: Authentication failure handling
- FIA_BLT_EXT.1: Extended: Bluetooth User Authorization
- FIA_BLT_EXT.2: Extended: Bluetooth Mutual Authentication
- FIA_BLT_EXT.3: Extended: Rejection of Duplicate Bluetooth Connections
- FIA_BLT_EXT.4: Extended: Secure Simple Pairing
- FIA_PAE_EXT.1: Extended: Port Access Entity Authentication
- FIA_PMG_EXT.1: Extended: Password Management
- FIA_PSK_EXT.1: Extended: Pre-Shared Key Composition - VPN

- 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.4: Extended: Secondary User Authentication
- FIA_X509_EXT.1: Extended: Validation of certificates
- FIA_X509_EXT.2: Extended: X509 certificate authentication
- FIA_X509_EXT.2/WLAN: Extended: X.509 Certificate Authentication (EAP-TLS) - WLAN
- 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.1/WLAN: Extended: Specification of Management Functions -WLAN
- FMT_SMF_EXT.2: Extended: Specification of Remediation Actions
- FMT_SMF_EXT.3 Extended: Current Administrator
- FPT_AEX_EXT.1: Extended: Anti-Exploitation Services (ASLR)
- FPT_AEX_EXT.2: Extended: Anti-Exploitation Services (Memory Page Permissions)
- FPT_AEX_EXT.3: Extended: Anti-Exploitation Services (Overflow Protection)
- FPT_AEX_EXT.4: Extended: Domain Isolation
- FPT_BBD_EXT.1: Extended: Application Processor Mediation
- FPT_JTA_EXT.1: Extended: JTAG Disablement
- FPT_KST_EXT.1: Extended: Key Storage
- FPT_KST_EXT.2: Extended: No Key Transmission
- FPT_KST_EXT.3: Extended: No Plaintext Key Export
- FPT_NOT_EXT.1: Extended: Self-Test Notification
- FPT_TST_EXT.1: Extended: TSF Cryptographic Functionality Testing
- FPT_TST_EXT.1/WLAN: Extended: TSF Cryptographic Functionality Testing - WLAN
- FPT_TST_EXT.1/VPN: Extended: TSF Self-Test - VPN
- FPT_TST_EXT.2(1): Extended: TSF Integrity Testing
- FPT_TST_EXT.2(2): Extended: TSF Integrity Testing
- FPT_TUD_EXT.1: Extended: Trusted Update: TSF version query
- FPT_TUD_EXT.2: Extended: Trusted Update Verification
- FTA_SSL_EXT.1: Extended: TSF- and User-initiated locked state
- FTA_WSE_EXT.1: Extended: Wireless Network Access - WLAN
- FTP_BLT_EXT.2: Extended: Bluetooth Encryption
- FTP_ITC_EXT.1: Extended: Trusted channel Communication
- FTP_ITC_EXT.1/WLAN: Extended: Trusted Channel Communication (Wireless LAN) - WLAN

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 MDFPP31/WLANCEP10/VPNC21. The refinements and operations already performed in the MDFPP31/WLANCEP10/VPNC21 are not identified (e.g., highlighted) here, rather the requirements have been copied from the MDFPP31/WLANCEP10/VPNC21 and any residual operations have been completed herein. Of particular note, the MDFPP31/WLANCEP10/VPNC21 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 MDFPP31/WLANCEP10/VPNC21 which include 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 MDFPP31/WLANCEP10/VPNC21 that serve to ensure corresponding evaluations will yield more practical and consistent assurance than the assurance requirements alone. The MDFPP31/WLANCEP10/VPNC21 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 Samsung Galaxy Devices on Android 8 TOE.

Requirement Class	Requirement Component
FAU: Security Audit	FAU_GEN.1: Audit Data Generation
	FAU_SAR.1: Audit Review
	FAU_SEL.1: Selective Audit
	FAU_STG.1: Audit Storage Protection
	FAU_STG.4: Prevention of Audit Data Loss
FCS: Cryptographic support	FCS_CKM.1: Cryptographic key generation
	FCS_CKM.1/WLAN: Cryptographic Key Generation (Symmetric Keys for WPA2 Connections) - WLAN
	FCS_CKM.1/VPN: Cryptographic Key Generation (IKE) - VPN
	FCS_CKM.2(1): Cryptographic key establishment
	FCS_CKM.2(2): Cryptographic key establishment (While device is locked)
	FCS_CKM.2/WLAN: Cryptographic Key Distribution (GTK) - WLAN
	FCS_CKM_EXT.1: Extended: Cryptographic Key Support
	FCS_CKM_EXT.2: Extended: Cryptographic Key Random Generation
	FCS_CKM_EXT.3: Extended: Cryptographic Key Generation
	FCS_CKM_EXT.4: Extended: Key Destruction
	FCS_CKM_EXT.5: Extended: TSF Wipe
	FCS_CKM_EXT.6: Extended: Salt Generation
	FCS_COP.1(1): Cryptographic operation
	FCS_COP.1(2): Cryptographic operation
	FCS_COP.1(3): Cryptographic operation
	FCS_COP.1(4): Cryptographic operation
	FCS_COP.1(5): Cryptographic operation
	FCS_HTTPS_EXT.1: Extended: HTTPS Protocol
	FCS_IPSEC_EXT.1: Extended: IPsec
	FCS_IV_EXT.1: Extended: Initialization Vector Generation
	FCS_RBG_EXT.1: Extended: Cryptographic Operation (Random Bit Generation)
	FCS_RBG_EXT.2: Extended: Cryptographic Operation (Random Bit Generation)
	FCS_SRV_EXT.1: Extended: Cryptographic Algorithm Services

Requirement Class	Requirement Component
	FCS_SRV_EXT.2: Extended: Cryptographic Algorithm Services
	FCS_STG_EXT.1: Extended: Cryptographic Key Storage
	FCS_STG_EXT.2: Extended: Encrypted Cryptographic Key Storage
	FCS_STG_EXT.3: Extended: Integrity of encrypted key storage
	FCS_TLSC_EXT.1: Extended: TLS Protocol
	FCS_TLSC_EXT.1/WLAN: Extended: Extensible Authentication Protocol-Transport Layer Security - WLAN
	FCS_TLSC_EXT.2: Extended: TLS Protocol
	FCS_TLSC_EXT.2/WLAN: Extended: TLS Client Protocol - WLAN
FDP: User data protection	FDP_ACF_EXT.1: Extended: Security access control
	FDP_ACF_EXT.2: Extended: Security access control
	FDP_ACF_EXT.3: Extended: Security attribute based access control
	FDP_DAR_EXT.1: Extended: Protected Data Encryption
	FDP_DAR_EXT.2: Extended: Sensitive Data Encryption
	FDP_IFC_EXT.1: Extended: Subset information flow control
	FDP_PBA_EXT.1: Extended: Storage of Critical Biometric Parameters
	FDP_RIP.2: Full Residual Information Protection
	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: Extended: Authentication failure handling
	FIA_BLT_EXT.1: Extended: Bluetooth User Authorization
	FIA_BLT_EXT.2: Extended: Bluetooth Mutual Authentication
	FIA_BLT_EXT.3: Extended: Rejection of Duplicate Bluetooth Connections
	FIA_BLT_EXT.4: Extended: Secure Simple Pairing
	FIA_BLT_EXT.6: Extended: Bluetooth User Authorization
	FIA_BMG_EXT.1(1): Extended: Accuracy of Biometric Authentication
	FIA_BMG_EXT.1(2): Extended: Accuracy of Biometric Authentication
	FIA_PAE_EXT.1: Extended: Port Access Entity Authentication
	FIA_PMG_EXT.1: Extended: Password Management
	FIA_PSK_EXT.1: Extended: Pre-Shared Key Composition - VPN
	FIA_TRT_EXT.1: Extended: Authentication Throttling
	FIA_UAU.5: Multiple Authentication Mechanisms
	FIA_UAU.6(1): Re-Authentication
	FIA_UAU.6(2): Re-Authentication
	FIA_UAU.7: Protected authentication feedback
	FIA_UAU_EXT.1: Extended: Authentication for Cryptographic Operation
	FIA_UAU_EXT.2: Extended: Timing of Authentication
	FIA_UAU_EXT.4: Extended: Secondary User Authentication
	FIA_X509_EXT.1: Extended: Validation of certificates
	FIA_X509_EXT.2: Extended: X509 certificate authentication
	FIA_X509_EXT.2/WLAN: Extended: X.509 Certificate Authentication (EAP-TLS) - WLAN
	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.1/WLAN: Extended: Specification of Management Functions - WLAN
	FMT_SMF.1/VPN: Specification of Management Functions - VPN
	FMT_SMF_EXT.2: Extended: Specification of Remediation Actions
	FMT_SMF_EXT.3 Extended: Current Administrator
FPT: Protection of	FPT_AEX_EXT.1: Extended: Anti-Exploitation Services (ASLR)

Requirement Class	Requirement Component
the TSF	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_AEX_EXT.5: Extended: Anti-Exploitation Services (ASLR)
	FPT_AEX_EXT.6: Extended: Anti-Exploitation Services (Memory Page Permissions)
	FPT_BBD_EXT.1: Extended: Application Processor Mediation
	FPT_JTA_EXT.1: Extended: JTAG Disablement
	FPT_KST_EXT.1: Extended: Key Storage
	FPT_KST_EXT.2: Extended: No Key Transmission
	FPT_KST_EXT.3: Extended: No Plaintext Key Export
	FPT_NOT_EXT.1: Extended: Self-Test Notification
	FPT_STM.1: Reliable time stamps
	FPT_TST_EXT.1: Extended: TSF Cryptographic Functionality Testing
	FPT_TST_EXT.1/WLAN: Extended: TSF Cryptographic Functionality Testing - WLAN
	FPT_TST_EXT.1/VPN: Extended: TSF Cryptographic Functionality Testing - VPN
	FPT_TST_EXT.2(1): Extended: TSF Integrity Checking
	FPT_TST_EXT.2(2): Extended: TSF Integrity Checking
	FPT_TUD_EXT.1: Extended: Trusted Update: TSF version query
	FPT_TUD_EXT.2: Extended: TSF Update Verification
FTA: TOE access	FTA_SSL_EXT.1: Extended: TSF- and User-initiated Locked State
	FTA_TAB.1: Default TOE Access Banners
	FTA_WSE_EXT.1: Extended: Wireless Network Access - WLAN
	FTP_ITC_EXT.1: Extended: Trusted channel Communication
	FTP_ITC_EXT.1/WLAN: Extended: Trusted Channel Communication (Wireless LAN) - WLAN

Table 4 TOE Security Functional Components

5.1.1 Security audit (FAU)

5.1.1.1 Audit Data Generation (FAU_GEN.1)

FAU_GEN.1.1

The TSF shall be able to generate an audit record of the following auditable events:
1. Start-up and shutdown of the audit functions
2. All auditable events for the not selected level of audit
3. All administrative actions
4. Start-up and shutdown of the Rich OS
5. Insertion or removal of removable media
6. Specifically defined auditable events in Table 1(of the MDFPP31);
7. [***Audit records reaching [95%] percentage of audit capacity***].
8. [***Objective audit events from the MDFPP31 as listed in*** *Error! Reference source not found.****, auditable events in Table 2 of WLANCEP10***]

]

FAU_GEN.1.2

The TSF shall record within each audit record at least the following information:
1. Date and time of the event
2. type of event
3. subject identity

4. the outcome (success or failure) of the event
5. additional information in Table 1
6. [***no additional information***]

5.1.1.2 Audit Review (FAU_SAR.1)

FAU_SAR.1.1

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

FAU_SAR.1.2

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

5.1.1.3 Selective Audit (FAU_SEL.1)

FAU_SEL.1.1

The TSF shall be able to select the set of events to be audited from the set of all auditable events based on the following attributes [

- ***success of auditable security events;***
- ***failure of auditable security events; and***
- ***[*event group, event severity and UserID*]***.

5.1.1.4 Audit Storage Protection (FAU_STG.1)

FAU_STG.1.1

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

FAU_STG.1.2

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

5.1.1.5 Prevention of Audit Data Loss (FAU_STG.4)

FAU_STG.4.1

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

5.1.2 Cryptographic support (FCS)

5.1.2.1 Cryptographic key generation (FCS_CKM.1)

FCS_CKM.1.1

The TSF shall generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm:
- 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,
[***- RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3]***].

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

FCS_CKM.1.1/WLAN

Refinement: The TSF shall generate symmetric cryptographic keys in accordance with a specified

cryptographic key generation algorithm PRF-384 and [***no other***] and specified cryptographic key sizes 128 bits and [***no other key sizes***] using a Random Bit Generator as specified in FCS_RBG_EXT.1 that meet the following: IEEE 802.11-2012 and [***IEEE 802.11ac-2014***].

5.1.2.3 Cryptographic Key Generation (IKE) - VPN (FCS_CKM.1/VPN)

FCS_CKM.1.1/VPN

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

5.1.2.4 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" and implementing "NIST curves" P-256, P-384, and [*P-521*] as defined in FIPS PUB 186-4, "Digital Signature Standard",**
- ***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'***].

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

FCS_CKM.2(2).1

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

5.1.2.6 Cryptographic Key Distribution (GTK) - WLAN (FCS_CKM.2/WLAN)

FCS_CKM.2.1/WLAN

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

5.1.2.7 Extended: Cryptographic Key Support (FCS_CKM_EXT.1)

FCS_CKM_EXT.1.1

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

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

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

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

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

5.1.2.9 Extended: Cryptographic Key Generation (FCS_CKM_EXT.3)

FCS_CKM_EXT.3.1
The TSF shall use [***asymmetric KEKs of [security strength greater than or equal to 128 bits] security strength, symmetric KEKs of [256-bit] security strength 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:
- derive the KEK from a Password Authentication Factor using PBKDF

and [
- ***generate the KEK using an RBG that meets this profile (as specified in FCS_RBG_EXT.1),***
- ***generate the KEK using a key generation scheme that meets this profile (as specified in FCS_CKM.1),***
- ***Combine the KEK from other KEKs in a way that preserves the effective entropy of each factor by [encrypting one key with another]***].

5.1.2.10 Extended: Key Destruction (FCS_CKM_EXT.4)

FCS_CKM_EXT.4.1
The TSF shall destroy cryptographic keys in accordance with the specified cryptographic key destruction methods:
- by clearing the KEK encrypting the target key
- in accordance with the following rules
  - For volatile memory, the destruction shall be executed by a single direct overwrite [***consisting of zeroes***].
  - 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 by a read-verify.
  - For non-volatile flash memory, that is not wear-leveled, the destruction shall be executed [***by a single direct overwrite consisting of zeros followed by a read-verify***].
  - For non-volatile flash memory, that is wear-leveled, the destruction shall be executed [***by a block erase***].
  - For non-volatile memory other than EEPROM and flash, the destruction shall be executed by a single direct overwrite with a random pattern that is changed before each write.

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

5.1.2.11 Extended: TSF Wipe (FCS_CKM_EXT.5)

FCS_CKM_EXT.5.1
The TSF shall wipe all protected data by [***Cryptographically erasing the encrypted DEKs and/or the KEKs in non-volatile memory by following the requirements in FCS_CKM_EXT.4.1***].

FCS_CKM_EXT.5.2

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

5.1.2.12 Extended: Salt Generation (FCS_CKM_EXT.6)

FCS_CKM_EXT.6.1

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

5.1.2.13 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),
- AES-GCM (as defined in NIST SP 800-38D), and [
- ***AES Key Wrap (KW) (as defined in NIST SP 800-38F),***
- ***AES-XTS (as defined in NIST SP 800-38E)***]

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

5.1.2.14 Cryptographic operation (FCS_COP.1(2))

FCS_COP.1(2).1

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

5.1.2.15 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-384 and [P-256, P-521] that meet the following: FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Section 5***].

5.1.2.16 Cryptographic operation (FCS_COP.1(4))

FCS_COP.1(4).1

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

5.1.2.17 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, SHA-512***]], with [**4096 for Keystore**] iterations, and output cryptographic key sizes [***256***] that meet the following: NIST SP 800-132.

5.1.2.18 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.1).

FCS_HTTPS_EXT.1.3
The TSF shall notify the application and [***no other action***] if the peer certificate is deemed invalid.

5.1.2.19 Extended: IPsec (FCS_IPSEC_EXT.1)

FCS_IPSEC_EXT.1.1
The [***TOE***] shall implement the IPsec architecture as specified in RFC 4301.

FCS_IPSEC_EXT.1.2
The [***TOE***] shall implement [***tunnel mode***].

FCS_IPSEC_EXT.1.3
The [***TOE***] shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched, and discards it.

FCS_IPSEC_EXT.1.4
The [***TOE***] shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms AES-GCM-128, AES-GCM-256 as specified in RFC 4106, [***AES-CBC-128, AES-CBC-256 (both specified by RFC 3602) together with a Secure Hash Algorithm (SHA)-based HMAC, no other algorithms***].

FCS_IPSEC_EXT.1.5
The [***TOE***] shall implement the protocol: [
- IKEv1, using Main Mode for Phase I exchanges, as defined in RFCs 2407, 2408, 2409, RFC 4109, [no other RFCs for extended sequence numbers], and [no other RFCs for hash functions] and [support for XAUTH];
- IKEv2 as defined in RFCs 7296 (with mandatory support for NAT traversal as specified in section 2.23), 4307, and [no other RFCs for hash functions]]. (TD0303 applied)

FCS_IPSEC_EXT.1.6
The [***TOE***] shall ensure the encrypted payload in the [***IKEv1, IKEv2***] protocol uses the cryptographic algorithms AES-CBC-128, AES-CBC-256 as specified in RFC 6379 and [***AES-GCM-128, AES-GCM-256 as specified in RFC 5282***][1].

FCS_IPSEC_EXT.1.7
The [***TOE***] shall ensure that [
- ***IKEv2 SA lifetimes can be configured by [VPN Gateway] based on [length of time];***
- ***IKEv1 SA lifetimes can be configured by [VPN Gateway] based on [length of time]***].

If length of time is used, it must include at least one option that is 24 hours or less for Phase 1 SAs and 8 hours or less for Phase 2 SAs.

FCS_IPSEC_EXT.1.8
The [***TOE***] shall ensure that all IKE protocols implement DH Groups 14 (2048-bit MODP), 19 (256-bit Random ECP), 20 (384-bit Random ECP), and [***5 (1536-bit MODP), 24 (2048-bit MODP with 256-bit POS)***].

FCS_IPSEC_EXT.1.9
The [***TOE***] shall generate the secret value x used in the IKE Diffie-Hellman key exchange ("x" in g^x mod p) using the random bit generator specified in FCS_RBG_EXT.1, and having a length of at least [**(224, 256, or 384)**] bits.

FCS_IPSEC_EXT.1.10
The [***TOE***] shall generate nonces used in IKE exchanges in a manner such that the probability that a specific nonce value will be repeated during the life a specific IPsec SA is less than 1 in 2^[**(112, 128, or 192)**].

[1] Note that AES-GCM-128 and AES-GCM-256 are supported only for IKEv2.

FCS_IPSEC_EXT.1.11

The [***TOE***] shall ensure that all IKE protocols perform peer authentication using a [***RSA, ECDSA***] that use X.509v3 certificates that conform to RFC 4945 and [***Pre-Shared Keys***].

FCS_IPSEC_EXT.1.12

The TSF shall not establish an SA if the [***IP address, Fully Qualified Domain Name (FQDN))***] and [***no other reference identifier type***] contained in a certificate does not match the expected value(s) for the entity attempting to establish a connection.

FCS_IPSEC_EXT.1.13

The TSF shall not establish an SA if the presented identifier does not match the configured reference identifier of the peer.

FCS_IPSEC_EXT.1.14

The [***VPN Gateway***] shall be able to ensure by default that the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [***IKEv1 Phase 1, IKEv2 IKE_SA***] connection is greater than or equal to the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [***IKEv1 Phase 2, IKEv2 CHILD_SA***] connection.

5.1.2.20 Extended: Initialization Vector Generation (FCS_IV_EXT.1)

FCS_IV_EXT.1.1

The TSF shall generate IVs in accordance with MDFPP31 Table 11: References and IV Requirements for NIST-approved Cipher Modes.

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

FCS_RBG_EXT.1.1

The TSF shall perform all deterministic random bit generation services in accordance with NIST Special Publication 800-90A using ***[Hash_DRBG (any), HMAC_DRBG (any), 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 [***128,256 bits***] of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

FCS_RBG_EXT.1.3

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

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

FCS_RBG_EXT.2.1

The TSF shall save the state of the deterministic RBG at power-off, and shall use this state as input to the deterministic RBG at startup.

5.1.2.23 Extended: Cryptographic Algorithm Services (FCS_SRV_EXT.1)

FCS_SRV_EXT.1.1

The TSF shall provide a mechanism for applications to request the TSF to perform the following cryptographic operations:

- All mandatory and [***selected algorithms***] in FCS_CKM.2(2)
- The following algorithms in FCS_COP.1(1): AES-CBC, [***AES-GCM***]
- All mandatory and selected algorithms in FCS_COP.1(3)
- All mandatory and selected algorithms in FCS_COP.1(2)
- All mandatory and selected algorithms in FCS_COP.1(4)
- [***No other cryptographic operations***].

5.1.2.24 Extended: Cryptographic Algorithm Services (FCS_SRV_EXT.2)

FCS_SRV_EXT.2.1

The TSF shall provide a mechanism for applications to request the TSF to perform the following cryptographic operations:
- Algorithms in FCS_COP.1(1)
- Algorithms in FCS_COP.1(3)

by keys stored in the secure key storage.

5.1.2.25 Extended: Cryptographic Key Storage (FCS_STG_EXT.1)

FCS_STG_EXT.1.1

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

FCS_STG_EXT.1.2

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

FCS_STG_EXT.1.3

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

FCS_STG_EXT.1.4

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

FCS_STG_EXT.1.5

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

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

FCS_STG_EXT.2.1

The TSF shall encrypt all DEKs and KEKs and [***Wi-Fi, Bluetooth, and SecurityLogAgent (related to SE Android)***] and [***all software-based key storage***] by KEKs that are [

1) Protected by the REK with [
- ***a. encryption by a REK,***
- ***b. encryption by a KEK that is derived from a REK],***

2) Protected by the REK and the password with [
- ***a. encryption by a REK and the password-derived KEK,***
- ***b. encryption by a KEK that is derived from a REK and the password-derived or biometric-unlocked KEK]***].

FCS_STG_EXT.2.2

DEKs, KEKs, [***Bluetooth and WPA2 PSK long-term trusted channel key material***] and [***all software-based key storage***] shall be encrypted using one of the following methods: [***using a SP800-56B key establishment scheme, using AES in the [GCM, CBC mode]***].

5.1.2.27 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 [***long-term trusted channel key material, all software-based key storage***] by [

- [GCM] cipher mode for encryption according to FCS_STG_EXT.2
- a hash (FCS_COP.1(2)) of the stored key that is encrypted by a key protected by FCS_STG_EXT.2
- a keyed hash (FCS_COP.1(4)) using a key protected by a key protected by FCS_STG_EXT.2].

FCS_STG_EXT.3.2

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

5.1.2.28 Extended: TLS Protocol (FCS_TLSC_EXT.1)

FCS_TLSC_EXT.1.1

The TSF shall implement TLS 1.2 (RFC 5246) supporting the following ciphersuites:
Mandatory Ciphersuites: [
- ***TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,***
- ***TLS_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246***]

Optional Ciphersuites: [
- ***TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,***
- ***TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***].

FCS_TLSC_EXT.1.2

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

FCS_TLSC_EXT.1.3

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

FCS_TLSC_EXT.1.4

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

5.1.2.29 Extended: Extensible Authentication Protocol-Transport Layer Security - WLAN (FCS_TLSC_EXT.1/WLAN)

FCS_TLSC_EXT.1.1/WLAN

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

Optional Ciphersuites: [
- ***TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,***
- ***TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,***
- ***TLS_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246,***
- ***TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246,***
- ***TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,***
- ***TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,***
- ***TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246,***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5430,***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5430,***
- ***TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,***
- ***TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,***
- ***TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***].

FCS_TLSC_EXT.1.2/WLAN

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

FCS_TLSC_EXT.1.3/WLAN

The TSF shall use X509 v3 certificates as specified in FIA_X509_EXT.1.

FCS_TLSC_EXT.1.4/WLAN
The TSF shall verify that the server certificate presented includes the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.

FCS_TLSC_EXT.1.5/WLAN
The TSF shall allow an authorized administrator to configure the list of CAs that are allowed to sign authentication server certificates that are accepted by the TOE.

FCS_TLSC_EXT.1.6/WLAN
The TSF shall allow an authorized administrator to configure the list of algorithm suites that may be proposed and accepted during the EAP-TLS exchanges.

5.1.2.30 Extended: TLS Protocol (FCS_TLSC_EXT.2)

FCS_TLSC_EXT.2.1
The TSF shall present the Supported Elliptic Curves Extension in the Client Hello handshake message with the following NIST curves: [***secp256r1, secp384r1***]. (TD0244 applied)

5.1.2.31 TLS Client Protocol - WLAN (FCS_TLSC_EXT.2/WLAN)

FCS_TLSC_EXT.2.1/WLAN
The TSF shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [***secp256r1, secp384r1***]. (TD0244 applied).

5.1.3 User data protection (FDP)

5.1.3.1 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, groups of applications***] from accessing [***all***] data stored by other [***application, groups of applications***]. Exceptions may only be explicitly authorized for such sharing by [***the administrator, a common application developer***].

5.1.3.2 Extended: Security access control (FDP_ACF_EXT.2)

FDP_ACF_EXT.2.1
The TSF shall provide a separate [***address book, calendar***] for each application group and only allow applications within that process group to access the resource. Exceptions may only be explicitly authorized for such sharing by [***the user***].

5.1.3.3 Extended: Security attribute based access control (FDP_ACF_EXT.3)

FDP_ACF_EXT.3.1
The TSF shall enforce an access control policy that prohibits an application from granting both write and execute permission to a file on the device except for [***files stored in the application's private data folder***].

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

5.1.3.5 Extended: Sensitive Data Encryption (FDP_DAR_EXT.2)

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

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

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

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

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

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

5.1.3.7 Extended: Storage of Critical Biometric Parameters (FDP_PBA_EXT.1)

FDP_PBA_EXT.1.1
The TSF shall protect the authentication template [***using a password as an additional factor***].

5.1.3.8 Full Residual Information Protection (FDP_RIP.2)

FDP_RIP.2.1
The [***TOE***] shall enforce that any previous information content of a resource is made unavailable upon the [***allocation of the resource to***] all objects.

5.1.3.9 Extended: User Data Storage (FDP_STG_EXT.1)

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

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

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

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

5.1.4 Identification and authentication (FIA)

5.1.4.1 Extended: Authentication failure handling (FIA_AFL_EXT.1)

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

FIA_AFL_EXT.1.2

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

FIA_AFL_EXT.1.3

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

FIA_AFL_EXT.1.4

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

FIA_AFL_EXT.1.5

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

FIA_AFL_EXT.1.6

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

5.1.4.2 Extended: Bluetooth User Authorization (FIA_BLT_EXT.1)

FIA_BLT_EXT.1.1

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

5.1.4.3 Extended: Bluetooth Mutual Authentication (FIA_BLT_EXT.2)

FIA_BLT_EXT.2.1

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

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

FIA_BLT_EXT.3.1

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

5.1.4.5 Extended: Secure Simple Pairing (FIA_BLT_EXT.4)

FIA_BLT_EXT.4.1

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

5.1.4.6 Extended: Bluetooth User Authorization (FIA_BLT_EXT.6)

FIA_BLT_EXT.6.1

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

5.1.4.7 Extended: Accuracy of Biometric Authentication (FIA_BMG_EXT.1(1))

FIA_BMG_EXT.1(1).1
The one-attempt BAF False Accept Rate (FAR) for [**fingerprint, iris**] shall not exceed [***1:10000***] with a one-attempt BAF False Reject Rate (FRR) not to exceed 1 in [***20***].

FIA_BMG_EXT.1(1).2
The overall System Authentication False Accept Rate (SAFAR) shall be no greater than 1 in [**1,000**] within a 1% margin.

5.1.4.8 Extended: Accuracy of Biometric Authentication (FIA_BMG_EXT.1(2))

FIA_BMG_EXT.1(2).1
The one-attempt BAF False Accept Rate (FAR) for [**hybrid**] shall not exceed [***1:10000***] with a one-attempt BAF False Reject Rate (FRR) not to exceed 1 in [***20***].

FIA_BMG_EXT.1(2).2
The overall System Authentication False Accept Rate (SAFAR) shall be no greater than 1 in [**1,000,000**] within a 1% margin.

5.1.4.9 Port Access Entity Authentication (FIA_PAE_EXT.1)

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

5.1.4.10 Extended: Password Management (FIA_PMG_EXT.1)

FIA_PMG_EXT.1.1
The TSF shall support the following for the Password Authentication Factor:
1. Passwords shall be able to be composed of any combination of [***upper and lower case letters***], numbers, and special characters: [***! @ # $ % ^ & * ( ) + = _ / - ' " : ; , ? ` ~ \ | < > { } [ ]***];
2. Password length up to [**16**] characters shall be supported.

5.1.4.11 Extended: Pre-Shared Key Composition - VPN (FIA_PSK_EXT.1)

FIA_PSK_EXT.1.1
The [***TOE***] shall be able to use pre-shared keys for IPsec.

FIA_PSK_EXT.1.2
The [***TOE***] shall be able to accept text-based pre-shared keys that:
- are 22 characters and [***/between 1 and 64 characters/***];
- composed of any combination of upper and lower case letters, numbers, and special characters (that include: '!', '@', '#', '$', '%', '^', '&', '*', '(', and ')').

FIA_PSK_EXT.1.3
The [***TOE***] shall condition the text-based pre-shared keys by using [***/concatenation of ASCII bit values/***], [***be able to [accept] bit-based pre-shared keys***].

5.1.4.12 Extended: Authentication Throttling (FIA_TRT_EXT.1)

FIA_TRT_EXT.1.1
The TSF shall limit automated user authentication attempts by [***enforcing a delay between incorrect authentication attempts***] for all authentication mechanisms selected in FIA_UAU.5.1. The minimum delay shall be such that no more than 10 attempts can be attempted per 500 milliseconds.

5.1.4.13 Multiple Authentication Mechanisms (FIA_UAU.5)

FIA_UAU.5.1

The TSF shall provide password and [***fingerprint, iris, hybrid***] to support user authentication.

FIA_UAU.5.2

The TSF shall authenticate any user's claimed identity according to the [**following rules:**

- **Passwords**
  - **Can be used at any time**
- **Biometric**
  - **Can only be used**
    - **When Work environment is not enabled for the device unlock screen,**
    - **when there is an enrolled biometric,**
    - **when the user enables the allow biometrics for unlock feature,**
    - **the non-critical biometric failed limit has not been reached, and**
    - **at OS unlock screen (not at reboot/power-on screen)**
- **Hybrid**
  - **For Work environments unlock and hybrid authentication factor configured by the user**

].

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

FIA_UAU.6(1).1

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

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

FIA_UAU.6(2).1

The TSF shall re-authenticate the user via an authentication factor defined in FIA_UAU.5.1 under the conditions TSF-initiated lock, user-initiated lock, [**no other conditions**].

5.1.4.16 Protected authentication feedback (FIA_UAU.7)

FIA_UAU.7.1

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

5.1.4.17 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.

5.1.4.18 Extended: Timing of Authentication (FIA_UAU_EXT.2)

FIA_UAU_EXT.2.1

The TSF shall allow [***[enter password or supply biometric authentication factor to unlock, make emergency calls, receive calls, take pictures and screen shots (automatically named and stored internally by the TOE), turn the TOE off, restart the TOE, see notifications, configure sound/vibrate/mute, set the volume (up and down) for various sound categories, see the configured banner, access Notification Panel functions (including toggles Always on Display, Flashlight, Do not disturb toggle, Airplane mode, Power saving, Auto rotate, and Sound (on, mute, vibrate) and access user configured Edge applications]***] on behalf of the user to be performed before the user is authenticated.

FIA_UAU_EXT.2.2

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

5.1.4.19 Extended: Secondary User Authentication (FIA_UAU_EXT.4)

FIA_UAU_EXT.4.1

The TSF shall provide a secondary authentication mechanism for accessing Enterprise applications and resources. The secondary authentication mechanism shall control access to the Enterprise application and shared resources and shall be incorporated into the encryption of protected and sensitive data belonging to Enterprise applications and shared resources.

FIA_UAU_EXT.4.2

The TSF shall require the user to present the secondary authentication factor prior to decryption of Enterprise application data and Enterprise shared resource data.

5.1.4.20 Extended: Validation of certificates (FIA_X509_EXT.1)

FIA_X509_EXT.1.1

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

- RFC 5280 certificate validation and certificate path validation
- The certificate path must terminate with a certificate in the Trust Anchor Database
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates
- The TSF shall validate the revocation status of the certificate using [***the Online Certificate Status Protocol (OCSP) as specified in RFC 2560, a Certificate Revocation List (CRL) as specified in RFC 5759***]
- The TSF shall validate the extendedKeyUsage field according to the following rules:
  - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field
  - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field
  - (Conditional) Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) in the extendedKeyUsage field

FIA_X509_EXT.1.2

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

5.1.4.21 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 IPsec, 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***].

5.1.4.22 Extended: X.509 Certificate Authentication (EAP-TLS) - WLAN (FIA_X509_EXT.2/WLAN)

FIA_X509_EXT.2.1/WLAN

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

FIA_X509_EXT.2.2/WLAN

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

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

FIA_X509_EXT.3.1

The TSF shall provide a certificate validation service to applications.

FIA_X509_EXT.3.2

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

5.1.5 Security management (FMT)

5.1.5.1 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 5 Security Management Functions** to the user.

FMT_MOF_EXT.1.2

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

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

FMT_SMF_EXT.1.1

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

FMT_SMF_EXT.1.2

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

5.1.5.3 Extended: Specification of Management Functions - WLAN (FMT_SMF_EXT.1/WLAN)

FMT_SMF_EXT.1.1/WLAN

The TSF shall be capable of performing the following management functions **in rows 48-55 of Table 5 Security Management Functions**

5.1.5.4 Specification of Management Functions - VPN (FMT_SMF.1/VPN)

FMT_SMF.1.1/VPN

The TSF shall be capable of performing the following management functions **in rows 56-61 of Table 5 Security Management Functions**

Management Function **Status Markers:** M – Mandatory I – Implemented optional function	FMT_SMF_EXT.1.1	FMT_MOF_EXT.1.	FMT_SMF_EXT.1.2	FMT_MOF_EXT.1.
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. There are distinct settings for the passwords used to unlock the base device and Knox Workspace container. Length: an integer value of characters (0 = no minimum) Complexity: Unspecified, Something, Numeric, Alphabetic, Alphanumeric, Complex. Lifetime: an integer value of days (0 = no maximum)	M		M	M
2. configure session locking policy: a. screen-lock enabled/disabled b. screen lock timeout c. number of authentication failures The administrator can configure the session locking policy using the Android MDM APIs. There are distinct settings for base device and Knox Workspace container inactivity. The user can also adjust each of the session locking policies for the base device and Knox Workspace container; 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). Screen lock timeout: an integer number of minutes before the TOE locks (0 = no lock timeout) Authentication failures: an integer number (0 = no limit)	M		M	M
3. enable/disable the VPN protection: a. across device [**b. on a per-app basis,** **c. on a per-group of applications processes basis**] The user can configure and then enable the TOE's VPN to protect traffic across the entire device. The administrator (through an MDM Agent that utilizes the TOE's MDM APIs) can restrict the TOE's ability to connect to a VPN. The administrator can configure per-app and per-container VPN connections with the Knox Workspace MDM APIs.	M		I	I
4. enable/disable [**NFC, Bluetooth, Wi-Fi, and cellular radios**] The administrator can disable the radios using the TOE's MDM APIs. Once disabled, a user cannot enable the radio. The administrator cannot fully disable/restrict cellular voice capabilities	M		I	I

Management Function **Status Markers:** M – Mandatory I – Implemented optional function	FMT_SMF_EXT.1.1	FMT_MOF_EXT.1.	FMT_SMF_EXT.1.2	FMT_MOF_EXT.1.
5. enable/disable [**camera, microphone**]: a. across device [**d.** ***no other method***] An administrator may configure the TOE (through an MDM agent utilizing the TOE's MDM APIs) to turn off the camera and or microphones. If the administrator has disabled either the camera or the microphones, then the user cannot use those capture devices. The administrator can also disable the use of the camera or microphone inside a Knox Workspace container without affecting access to those devices when outside the container.	M		I	I
6. transition to the locked state Both users and administrators (using the TOE's MDM APIs) can transition the TOE into a locked state.	M		M	-
7. TSF wipe of protected data Both users and administrators (using the TOE's MDM APIs) can force the TOE to perform a full wipe (factory reset) of data.	M		M	
8. configure application installation policy by: ***a. restricting the sources of applications*** [***b. specifying a set of allowed applications based on [application name, developer signature] (an application whitelist)*** ***c. denying installation of applications***] The administrator using the TOE's MDM APIs can configure the TOE so that applications cannot be installed and can also block the use of the Google Market Place. There are distinct settings for disabling the installation of applications for the base device and Knox Workspace container.	M		M	M
9. import keys/secrets into the secure key storage Both users and administrators (using the TOE's MDM APIs) can import secret keys into the secure key storage.	M		I	
10. destroy imported keys/secrets and [***no other keys/secrets***] in the secure key storage Both users and administrators (using the TOE's MDM APIs) can destroy secret keys in the secure key storage.	M		I	
11. import X.509v3 certificates into the Trust Anchor Database Both users and administrators (using the TOE's MDM APIs) can import X.509v3 certificates into the Trust Anchor Database (note that the container does *not* have a separate TAB, but instead shares the TAD of the mobile device).	M		M	
12. remove imported X.509v3 certificates and [***default X.509v3 certificates***] in the Trust Anchor Database Both users and administrators (using the TOE's MDM APIs) can remove imported	M		I	

Management Function **Status Markers:** M – Mandatory I – Implemented optional function	FMT_SMF_EXT.1.1	FMT_MOF_EXT.1.	FMT_SMF_EXT.1.2	FMT_MOF_EXT.1.
X.509v3 certificates from the Trust Anchor Database as well as disable any of the TOE's default Root CA certificates (in the latter case, the CA certificate still resides in the TOE's read-only system partition; however, the TOE will treat that Root CA certificate and any certificate chaining to it as untrusted).				
13. enroll the TOE in management TOE users can enroll the TOE in management according to the instructions specific to a given MDM. Presumably any enrollment would involve at least some user functions (e.g., install an MDM agent application) on the TOE prior to enrollment.	M	M		
14. remove applications Both users and administrators (using the TOE's MDM APIs) can uninstall user and administrator installed applications on the TOE (non-container) and applications inside a Knox Workspace container.	M		M	
15. update system software Users can check for updates and cause the device to update if an update is available. An administrator can use MDM APIs to query the version of the TOE and query the installed applications and an MDM agent on the TOE could issue pop-ups, initiate updates, block communication, etc. until any necessary updates are completed. Note that the system software covers the entire mobile device (including all container software).	M		M	
16. install applications Both users and administrators (using the TOE's MDM APIs) can install applications on the TOE (non-container) and applications inside a Knox Workspace container.	M		M	
17. remove Enterprise applications Both users and administrators (using the TOE's MDM APIs) can uninstall user and administrator installed applications on the TOE (non-container) and applications inside a Knox Workspace container. Applications installed within the Knox Workspace are marked as Enterprise (Work) applications	M		M	
18. configure the Bluetooth trusted channel: a. disable/enable the Discoverable mode (for BR/EDR) b. change the Bluetooth device name [**e.** ***allow/disallow Android Beam and S-Beam to be used with Bluetooth*** **h.** ***disable/enable the Bluetooth services and/or profiles available on the device (for BR/EDR and LE),]*** TOE users can enable Bluetooth discoverable mode for a short period of time and can also change the device name which is used for the Bluetooth name. Additional wireless technologies include Android Beam and S-Beam which are related to NFC and can be enabled and disabled by the TOE user. The administrator can restrict only items a and e.	M			
19. enable/disable display notification in the locked state of: [**f.** ***all notifications***]	M			

Management Function **Status Markers:** M – Mandatory I – Implemented optional function	FMT_SMF_EXT.1.1	FMT_MOF_EXT.1.	FMT_SMF_EXT.1.2	FMT_MOF_EXT.1.
TOE users can configure the TOE to allow or disallow notifications while in a locked state.				
20. enable data-at rest protection The administrator (using the TOE's MDM APIs) can configure a device encryption policy. Once enabled the TOE internal flash will be encrypted. Users can also encrypt the TOE internal flash even if not enforced by administrator policy. However, if the policy is set the user cannot decrypt the TOE.	M		I	I
21. enable removable media's data-at-rest protection The administrator (using the TOE's MDM APIs) can configure a device removable media encryption policy. Once enabled, files on external SD cards will be encrypted. Users can also encrypt the TOE internal flash even if not enforced by administrator policy. However, if the policy is set the user cannot decrypt the TOE.	M		I	I
22. enable/disable location services: a. across device [**d.** ***no other method***] The administrator (using the TOE's MDM APIs) can disable location services. Unless disabled by the administrator, TOE users can enable and disable location services.	M		I	I
23. enable/disable the use of [***Fingerprint, Iris, Hybrid Authentication***] The TOE supports Biometric Fingerprints or Iris (alone), or Hybrid depending on the configuration. The TOE container only supports Hybrid while the device can support either depending on the configuration, though they are mutually exclusive.	M		I	I
24. enable/disable all data signaling over [assignment: ***list of externally accessible hardware ports***]				
25. enable/disable [**Wi-Fi tethering, USB tethering, and Bluetooth tethering**] The administrator (using the TOE's MDM APIs) can individually disable each tethering method. Unless disabled by the administrator, TOE users can individually enable and disable tethering via a Wi-Fi hotspot, USB connection, and Bluetooth pairing. The TOE acts as a server (acting as an access point, a USB Ethernet adapter, and as a Bluetooth Ethernet adapter respectively) in order to share its network connection with another device.	I		I	I
26. enable/disable developer modes The administrator (using the TOE's MDM APIs) can disable Developer Mode. Unless disabled by the administrator, TOE users can enable and disable Developer Mode.	I		I	I
27. enable/disable bypass of local user authentication	I		I	I
28. wipe Enterprise data The TOE container provides the ability to remove only Enterprise data versus user	I			

Management Function **Status Markers:** M – Mandatory I – Implemented optional function	FMT_SMF_EXT.1.1	FMT_MOF_EXT.1.	FMT_SMF_EXT.1.2	FMT_MOF_EXT.1.
data.				
29. approve [***import***] by applications of X.509v3 certificates in the Trust Anchor Database The TOE will prompt the User to approve requests by an application to import an X.509v3 CA Certificate into the TOE's Trust Anchor Database. The User, when prompted, can allow or deny the import.				
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 The administrator (using MDM APIs) can view the TOE's audit records.	I		I	
33. configure [selection: *certificate*, *public-key*] used to validate digital signature on applications				
34. approve exceptions for shared use of keys/secrets by multiple applications				
35. approve exceptions for destruction of keys/secrets by applications that did not import the key/secret				
36. configure the unlock banner The administrator (using the TOE's MDM APIs) can defined a banner of a maximum of 256 characters to be displayed while the TOE is locked. There is no method for the user to change the banner.	I		I	I
37. configure the auditable items	I		I	I
38. retrieve TSF-software integrity verification values				
39. enable/disable [a. ***USB mass storage mode***] The administrator (using the TOE's MDM APIs) can disable USB mass storage mode.	I		I	I
40. enable/disable backup of [**selection:** ***all applications, selected applications, selected groups of applications, configuration data***] to [**selection:** ***local system, remote system***]				
41. enable/disable [**a. *Hotspot functionality authenticated by [pre-shared key],*** b. ***USB tethering authenticated by [no authentication]***] The administrator (using the TOE's MDM APIs) can disable the Wi-Fi hotspot and USB tethering. Unless disabled by the administrator, TOE users can configure the Wi-Fi hotspot with a pre-shared key and can configure USB tethering (with no authentication).	I		I	I
42. approve exceptions for sharing data between [***groups of application processes***] The TOE container provides separation between groups of application processes along with the ability to control the ability to share data between these groups	I			I

Management Function **Status Markers:** M – Mandatory I – Implemented optional function	FMT_SMF_EXT.1.1	FMT_MOF_EXT.1.	FMT_SMF_EXT.1.2	FMT_MOF_EXT.1.
43. place applications into application process groups based on [assignment: *enterprise configuration settings*]				
44. unenroll the TOE from management	I		I	I
45. Enable/disable the Always On VPN protection	I		I	I
46. Revoke Biometric template				
47. [**enable/disable USB host storage** **disable CC Mode** **enable/disable Admin removal** **enable/disable manual Date/Time changes** **enable/disable applications (including pre-installed)** **enable/disable auto-completion of browser text input** **configure whitelist/blacklist of allowed email accounts**] The user can always disable CC Mode by entirely wiping the device (factory reset), as this will return the phone to its factory state (in which CC Mode has not been enabled)	I I I I I I I		I I I I I I I	I I I I I I I
FMT_SMF_EXT.1(1)/WLAN				
48. configure security policy for each wireless network: a. [***specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s)***] b. security type c. authentication protocol d. client credentials to be used for authentication;	M		M	I
49. specify wireless networks (SSIDs) to which the TSF may connect;	I		I	I
50. enable/disable certificate revocation list checking;				
51. disable ad hoc wireless client-to-client connection capability;				
52. disable wireless network bridging capability (for example, bridging a connection between the WLAN and cellular radios on a smartphone so it can function as a hotspot);				
53. disable roaming capability				
54. enable/disable IEEE 802.1X pre-authentication				
55. enable/disable and configure PMK caching a. set the amount of time (in minutes) for which PMK entries are cached b. set the maximum number of PMK entries that can be cached				
FMT_SMF.1/VPN				
56. Specify VPN gateways to use for connections	I			
57. Specify IPsec VPN Clients to use for connections				
58. Specify IPsec-capable network devices to use for connections				
59. Specify client credentials to be used for connections	M			
60. Configure the reference identifier of the peer	M			
61. [any additional VPN management functions]				

Table 5 Security Management Functions

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

FMT_SMF_EXT.2.1

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

5.1.5.6 Extended: Current Administrator (FMT_SMF_EXT.3)

FMT_SMF_EXT.3.1

The TSF shall provide a mechanism that allows users to view a list of currently authorized administrators and the management functions that each administrator is authorized to perform.

5.1.6 Protection of the TSF (FPT)

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

FPT_AEX_EXT.1.1

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

FPT_AEX_EXT.1.2

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

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

FPT_AEX_EXT.2.1

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

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

FPT_AEX_EXT.3.1

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

5.1.6.4 Extended: Domain Isolation (FPT_AEX_EXT.4)

FPT_AEX_EXT.4.1

The TSF shall protect itself from modification by untrusted subjects.

FPT_AEX_EXT.4.2

The TSF shall enforce isolation of address space between applications.

5.1.6.5 Extended: Anti-Exploitation Services (ASLR) (FPT_AEX_EXT.5)

FPT_AEX_EXT.5.1

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

FPT_AEX_EXT.5.2

The base address of any kernel-space memory mapping will consist of at least 4 unpredictable bits.

5.1.6.6 Extended: Anti-Exploitation Services (Memory Page Permissions) (FPT_AEX_EXT.6)

FPT_AEX_EXT.6.1

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

5.1.6.7 Extended: Application Processor Mediation (FPT_BBD_EXT.1)

FPT_BBD_EXT.1.1

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

5.1.6.8 Extended: JTAG Disablement (FPT_JTA_EXT.1)

FPT_JTA_EXT.1.1

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

5.1.6.9 Extended: Key Storage (FPT_KST_EXT.1)

FPT_KST_EXT.1.1

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

5.1.6.10 Extended: No Key Transmission (FPT_KST_EXT.2)

FPT_KST_EXT.2.1

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

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

FPT_KST_EXT.3.1

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

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

FPT_NOT_EXT.1.1

The TSF shall transition to non-operational mode and [***force User authentication failure***] when the following types of failures occur:

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

5.1.6.13 Reliable time stamps (FPT_STM.1)

FPT_STM.1.1

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

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

FPT_TST_EXT.1.1

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

5.1.6.15 Extended: TSF Cryptographic Functionality Testing - WLAN (FPT_TST_EXT.1/WLAN)

FPT_TST_EXT.1.1/WLAN

The [***TOE platform***] shall run a suite of self-tests during initial start-up (on power on) to demonstrate the correct operation of the TSF.

FPT_TST_EXT.1.2/WLAN

The [***TOE platform***] shall provide the capability to verify the integrity of stored TSF executable code when it is loaded for execution through the use of the TSF-provided cryptographic services.

5.1.6.16 Extended: TSF Self Test - VPN (FPT_TST_EXT.1/VPN)

FPT_TST_EXT.1.1/VPN
The [***TOE Platform***] shall run a suite of self -tests during initial start-up (on power on) to demonstrate the correct operation of the TSF.

FPT_TST_EXT.1.2/VPN
The [***TOE, TOE Platform***] shall provide the capability to verify the integrity of stored TSF executable code when it is loaded for execution through the use of the [**cryptographic signature and hash for integrity**].

5.1.6.17 Extended: TSF Integrity Checking (FPT_TST_EXT.2(1))

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

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

FPT_TST_EXT.2(2).1
The TSF shall verify the integrity of [***[the /system partition]***], stored in mutable media prior to its execution through the use of [***hardware-protected hash***].

5.1.6.19 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.

5.1.6.20 Extended: TSF Update Verification (FPT_TUD_EXT.2)

FPT_TUD_EXT.2.1
The TSF shall verify software updates to the Application Processor system software and [***[communications processor software, bootloader software, carrier specific configuration]***] using a digital signature verified by the manufacturer trusted key prior to installing those updates.

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

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

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

5.1.7 TOE access (FTA)

5.1.7.1 Extended: TSF- and User-initiated Locked State (FTA_SSL_EXT.1)

FTA_SSL_EXT.1.1

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

FTA_SSL_EXT.1.2

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

FTA_SSL_EXT.1.3

The TSF shall, upon transitioning to the locked state, perform the following operations:

a. clearing or overwriting display devices, obscuring the previous contents;

b. [**no other actions**].

5.1.7.2 Default TOE Access Banners (FTA_TAB.1)

FTA_TAB.1.1

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

5.1.7.3 Wireless Network Access (FTA_WSE_EXT.1)

FTA_WSE_EXT.1.1

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

5.1.8 Trusted path/channels (FTP)

5.1.8.1 Extended: Trusted channel Communication (FTP_ITC_EXT.1)

FTP_ITC_EXT.1.1

The TSF shall use 802.11-2012, 802.1X, and EAP-TLS, IPsec 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***].

5.1.8.2 Trusted Channel Communication - WLAN (FTP_ITC_EXT.1/WLAN)

FTP_ITC_EXT.1.1/WLAN

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

FTP_ITC_EXT.1.2/WLAN

The TSF shall initiate communication via the trusted channel for wireless access point connections.

5.2 TOE Security Assurance Requirements

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

Requirement Class	Requirement Component
ADV: Development	ADV_FSP.1: Basic Functional Specification
AGD: Guidance documents	AGD_OPE.1: Operational User Guidance
	AGD_PRE.1: Preparative Procedures
ALC: Life-cycle support	ALC_CMC.1: Labelling of the TOE
	ALC_CMS.1: TOE CM Coverage
	ALC_TSU_EXT.1: Timely Security Updates
ATE: Tests	ATE_IND.1: Independent Testing - sample
AVA: Vulnerability assessment	AVA_VAN.1: Vulnerability Survey

Table 6 EAL 1 augmented with ALC_TSU_EXT.1 Assurance Components

5.2.1 Development (ADV)

5.2.1.1 Basic Functional Specification (ADV_FSP.1)

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

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

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

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

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

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

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

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

5.2.2 Guidance documents (AGD)

5.2.2.1 Operational User Guidance (AGD_OPE.1)

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

AGD_OPE.1.1c
The operational user guidance shall describe, for each user role, the user-accessible functions and

privileges that should be controlled in a secure processing environment, including appropriate warnings.

AGD_OPE.1.2c

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

AGD_OPE.1.3c

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

AGD_OPE.1.4c

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

AGD_OPE.1.5c

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

AGD_OPE.1.6c

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

AGD_OPE.1.7c

The operational user guidance shall be clear and reasonable.

AGD_OPE.1.1e

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

5.2.2.2 Preparative Procedures (AGD_PRE.1)

AGD_PRE.1.1d

The developer shall provide the TOE, including its preparative procedures.

AGD_PRE.1.1c

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

AGD_PRE.1.2c

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

AGD_PRE.1.1e

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

AGD_PRE.1.2e

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

5.2.3 Life-cycle support (ALC)

5.2.3.1 Labelling of the TOE (ALC_CMC.1)

ALC_CMC.1.1d

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

ALC_CMC.1.1c

The TOE shall be labelled with its unique reference.

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

5.2.3.2 TOE CM Coverage (ALC_CMS.1)

ALC_CMS.1.1d
The developer shall provide a configuration list for the TOE.

ALC_CMS.1.1c
The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.

ALC_CMS.1.2c
The configuration list shall uniquely identify the configuration items.

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

5.2.3.3 Timely Security Updates (ALC_TSU_EXT.1)

ALC_TSU_EXT.1.1d
The developer shall provide a description in the TSS of how timely security updates are made to the TOE.

ALC_TSU_EXT.1.1c
The description shall include the process for creating and deploying security updates for the TOE software.

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.1.4c
The description shall include where users can seek information about the availability of new updates including details (e.g. CVE identifiers) of the specific public vulnerabilities corrected by each update.

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 - sample (ATE_IND.1)

ATE_IND.1.1d
The developer shall provide the TOE for testing.

ATE_IND.1.1c
The TOE shall be suitable for testing.

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

ATE_IND.1.2e
The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

5.2.5 Vulnerability assessment (AVA)

5.2.5.1 Vulnerability Survey (AVA_VAN.1)

AVA_VAN.1.1d

The developer shall provide the TOE for testing.

AVA_VAN.1.1c

The TOE shall be suitable for testing.

AVA_VAN.1.1e

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

AVA_VAN.1.2e

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

AVA_VAN.1.3e

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

6. TOE Summary Specification

This chapter describes the security functions:

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

6.1 Security audit

FAU_GEN.1: The following table enumerates the events that the TOE audits. Requirements marked with "(O)" are from the Table 2: Additional Auditable Events of the MDFPP31.

Requirement	Audit Event	Content
FAU_GEN.1	Start-up and shutdown of the audit functions	
FAU_GEN.1	All administrative actions	
FAU_GEN.1	Start-up and shutdown of the OS and kernel	
FAU_GEN.1	Insertion or removal of removable media	
FAU_GEN.1	Establishment of a synchronizing connection	
FAU_GEN.1	Reaching the configured audit log critical size limit	
FCS_CKM_EXT.5	Failure of the wipe.	
FCS_CKM.1	Failure of key generation activity for authentication keys.	
FCS_HTTPS_EXT.1 (O)	Failure of the certificate validity check.	Issuer Name and Subject Name of certificate.
FCS_STG_EXT.1	Import or destruction of key.	Identity of key. Role and identity of requestor.
FCS_STG_EXT.3	Failure to verify integrity of stored key.	Identity of key being verified.
FCS_TLSC_EXT.1 (O)	Failure to establish a TLS session.	Reason for failure.
FCS_TLSC_EXT.1 (O)	Failure to verify presented identifier.	Presented identifier and reference identifier.
FCS_TLSC_EXT.1 (O)	Establishment/termination of a TLS session.	Non-TOE endpoint of connection.
FCS_TLSC_EXT.1/WLAN	Failure to establish an EAP-TLS session.	Reason for failure.
FCS_TLSC_EXT.1/WLAN	Establishment/termination of an EAP-TLS session.	Non-TOE endpoint of connection.
FDP_DAR_EXT.2	Failure to encrypt/decrypt data.	
FDP_STG_EXT.1	Addition or removal of certificate from Trust Anchor Database.	Subject name of certificate.
FIA_UAU.6 (O)	User changes Password Authentication Factor.	

FIA_X509_EXT.1	Failure to validate X.509v3 certificate.	Reason for failure of validation.
FIA_X509_EXT.2 (O)	Failure to establish connection to determine revocation status.	
FMT_SMF_EXT.1	Change of settings.	Role of user that changed setting. Value of new setting.
FMT_SMF_EXT.1 (O)	Success or failure of function.	Role of user that performed function. Function performed. Reason for failure.
FMT_SMF_EXT.1 (O)	Initiation of software update.	Version of update.
FMT_SMF_EXT.1 (O)	Initiation of application installation or update.	Name and version of application.
FPT_TST_EXT.1 & FPT_TST_EXT.1/WLAN	Initiation of self-test. Failure of self-test.	[***none***]
FPT_TST_EXT.2(1)	Start-up of TOE. [***none***]	No additional Information. [***no additional information***]
FPT_TUD_EXT.2 (O)	Success or failure of signature verification for software updates.	
FPT_TUD_EXT.2 (O)	Success or failure of signature verification for applications.	
FTA_TAB.1 (O)	Change in banner setting.	
FCS_TLSC_EXT.1/WLAN	Failure to establish an EAP-TLS session.	Reason for failure.
FCS_TLSC_EXT.1/WLAN	Establishment/termination of an EAP-TLS session	Non-TOE endpoint of connection
FPT_TST_EXT.1/WLAN	Execution of this set of TSF self-tests. • [***detected integrity violations***]	[***The TSF binary file that caused the integrity violation***].
FTA_WSE_EXT.1	All attempts to connect to access points.	Identity of access point being connected to as well as success and failures (including reason for failure).
FTP_ITC_EXT.1/WLAN	All attempts to establish a trusted channel. (TD0194 applied)	Identification of the non-TOE endpoint of the channel.

Table 7 Audit Events

FAU_SAR.1: The TOE provides the ability for the administrator to export and read the audit log.

FAU_SEL.1: The TOE provides the ability to select the audit events that are logged. There are three classes of audit events that can be logged, system and apps, kernel and IP tables. Each can be controlled individually, so the administrator can log just select classes of events. Within the classes, events can be selected based upon success, failure, severity, event group, and UID.

FAU_STG.1: The TOE stores audit records in a file within the file system accessible only to Linux processes with system permissions (effectively the TSF itself and MDM agents possessing a valid Knox license using the defined APIs). These restrictions prevent the unauthorized modification or deletion of the audit records stored in the audit files.

FAU_STG.4: The TOE pre-allocates a file system area (between 10MB and 50MB in size, depending upon available storage on the device) by creating a /data/system/[admin_uid]_bubble/bubbleFile and directory (/data/system/[admin_uid]) in which to archive compressed audit logs. If the TOE lacks sufficient space (at least 10MB), then the TOE returns a failure code in response to the administrator's attempt to enable the AuditLog. Once enabled, the TOE writes audit events into nodes until they read a given size, and then compresses and archives the records. The TOE utilizes a circular buffer approach to handle when the accumulated, compressed audit events exceed the allocated file system size. When the limit is reached, the TOE removes the oldest audit logs, freeing space for new records.

6.2 Cryptographic support

FCS_CKM.1: The TOE supports asymmetric key generation for all types in accordance with FIPS 186-4. The TOE generates RSA keys in its SCrypto library and generates DH/ECDH/ECDSA (including P-256, P384 and P-521) keys in both its SCrypto and BoringSSL libraries. 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. The TOE's RSA and ECDSA implementations have the CAVP certificates described in the FCS_COP.1 section below.

Cryptographic Library	RSA Generation	DH (FCC)	ECDH (ECC)	EDCSA (ECC)
BoringSSL (user space)	No	Yes	Yes	Yes
Kernel Crypto (Kernel)	No	No	No	No
SCrypto (TrustZone)	Yes	No	No	Yes
Application Processor	No	No	No	No

Table 8 Asymmetric Key Generation per module

FCS_CKM.1/WLAN: The TOE adheres to IEEE 802.11-2012 and IEEE 802.11ac-2014 for key generation. The TOE's wpa_supplicant provides the PRF384 for WPA2 derivation of 128-bit AES Temporal Key (using the HMAC implementation provided by BoringSSL) and also employs its BoringSSL AES-256 DRBG when generating random values used in the EAP-TLS and 802.11 4-way handshake. The TOE supports the AES-128 CCMP encryption mode. The TOE has successfully completed certification (including WPA2 Enterprise) and received Wi-Fi CERTIFIED Interoperability Certificates from the Wi-Fi Alliance. The Wi-Fi Alliance maintains a website providing further information about the testing program: http://www.wi-fi.org/certification.

Device Name	Model Number	Wi-Fi Alliance Certificate Numbers
Galaxy S9+	SM-G965x	73947, 73948, 74090, 75146, 75153, 75267
Galaxy S9	SM-G960x	73958, 73959, 73962, 75099, 75141, 75250
Galaxy S8+	SM-G955x	69849, 69616, 70173
Galaxy S8	SM-G950x	69612, 69529, 70171

Table 9 Wi-Fi Alliance Certificates

FCS_CKM.1/VPN: The VPN uses the TOE cryptographic libraries to generate asymmetric keys (RSA or ECDSA) for authentication during the IKE key exchange. Note that ECDSA is only supported by IKEv2 connections.

FCS_CKM.2(1): The TOE supports RSA (800-56B, as an initiator only), DHE (FFC 800-56A), and ECDHE (ECC 800-56A) methods in TLS key establishment/exchange. The TOE has CVL KAS and ECDSA CAVP algorithm certificates for Elliptic Curve key establishment and key generation respectively as described in the FCS_COP.1 section below. Samsung vendor-affirms that the TOE's RSA key establishment follows 800-56B. The user and administrator need take no special configuration of the TOE as the TOE automatically generates the keys needed for negotiated TLS ciphersuites. 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 uses ECDH with a P-256 curve key establishment for protection of application sensitive data received while the device is locked.

FCS_CKM.2/WLAN: 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 supports a Root Encryption Key (REK) within the main (application) processor. Requests for encryption or decryption chaining to the REK are only accessible through the Trusted Execution Environment, or TEE (TrustZone). The REK lies in a series of 256-bit fuses, programmed during manufacturing.

The TEE does not allow direct access to the REK but provides services to derive a HEK (Hardware Encryption Key, which is derived from the REK through a KDF function) for encryption and decryption.

The REK value is generated during manufacturing either by the TOE (if it detects that the REK fuses have not been set) using it hardware DRBG or is generated during fabrication using an external RBG that meets the requirements of this PP in that the process utilizes a SHA-256 Hash_DRBG seeded by a hardware entropy source identical in architecture to that within the TOE. This fabrication process includes strict controls (including physical and logical access control to the manufacturing room where programming takes place as well as video surveillance and access only to specific, authorized, trusted individuals) to ensure that the fabricator cannot access any REK values between generation and programming.

FCS_CKM_EXT.2: The TOE supports Data Encryption Key (DEK) generation using its approved RBGs. The TOE RBGs are capable of generating both AES 128-bit and AES 256-bit DEKs in response to applications and services on the device. These can be accessed through both Android native APIs and C APIs depending on the library being called.

FCS_CKM_EXT.3: The TOE generates KEKs (which are always AES 256-bit keys generated by one of the TOE's DRBGs) through a combination of methods. First, the TOE generates a KEK (the Keystore masterkey) for each user of the TOE. The TOE also generates encryption KEKs for Onboard Device Encryption (ODE), the SD Card encryption, and container (Knox) encryption (both normal and sensitive).

The TOE generates a number of different KEKs. In addition to the TSF KEKs, applications may request key generation (through the Android/Java APIs), and the TOE utilizes its BoringSSL CTR_DRBG to satisfy those requests. The requesting application ultimately chooses whether to use that key as a DEK or a KEK (and can choose whether it wishes that key to be 128 or 256-bits), but it is worth mentioning here, as an application can utilize such a key as a KEK, should it choose.

FCS_CKM_EXT.4: The TOE destroys cryptographic keys when they are no longer in use by the system. The exceptions to this are public keys (that protect the boot chain and software updates) and the REK, which are never cleared. Keys stored in RAM during use are destroyed by a zero overwrite. Keys stored in Flash (i.e. eMMC) are destroyed by cryptographic erasure through a block erase call to the flash controller for the location where the ODE and SD Card keys are stored. Once these are erased, all keys (and data) stored within the encrypted data partition of the TOE are considered cryptographically erased.

FCS_CKM_EXT.5: The TOE provides a TOE Wipe function that first erases the encrypted ODE DEK used to encrypt the entire data partition using a block erase and read verify command to ensure that the TOE writes zeros to the eMMC blocks containing the encrypted ODE DEK (the crypto footer contains the encrypted DKEKs and DEKs (ODE and SD Card)). After overwriting that, the TOE will reformat the partition. Upon completion of reformatting the Flash partition holding user data, the TOE will perform a power-cycle.

FCS_CKM_EXT.6: The TOE creates salt and nonces (which are just salt values used in WPA2) using its AES-256 CTR_DRBG.

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

Table 10 Salt Creation

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

The BoringSSL v1.2 library provides the following algorithms.

Algorithm	NIST Standard	SFR Reference	Cert#
AES 128/256 CBC, GCM, KW	FIPS 197, SP 800-38A/D/F	FCS_COP.1(1)	5255

Algorithm	NIST Standard	SFR Reference	Cert#
CVL ECC/FFC	SP 800-56A	FCS_CKM.2(1) FCS_CKM_EXT.3	1725
DSA FFC SigGen/SigVer	FIPS 186-4	FCS_CKM.1	1361
DRBG CTR	SP 800-90A	FCS_RBG_EXT.1	2010
ECDSA PKG/PKV/SigGen/SigVer	FIPS 186-4	FCS_CKM.1 FCS_CKM.2(1) FCS_COP.1(3)	1369
HMAC SHA-1/256/384/512	FIPS 198-1 & 180-4	FCS_COP.1(4)	3479
RSA SigGen/SigVer	FIPS 186-4	FCS_COP.1(3)	2810
SHS SHA-1/256/384/512	FIPS 180-4	FCS_COP.1(2)	4230

Table 11 BoringSSL Cryptographic Algorithms

The Samsung SDPCrypto Module v1.0 provides the following algorithm

Algorithm	NIST Standard	SFR Reference	Cert#
KBKDF	SP 800-108	FCS_CKM_EXT.3	192

Table 12 SDPCrypto Cryptographic Algorithms

The Samsung Kernel Cryptographic ("Kernel Crypto") Module provides the following algorithms. The S9/S9+ uses version 1.9 of the Samsung Kernel Cryptographic Module while the S8/S8+ uses version 1.8.

Algorithm	NIST Standard	SFR Reference	Cert#
AES 128/256 CBC	FIPS 197, SP 800-38A	FCS_COP.1(1)	5184, 5183, 4427, 4426, 4425, 4424
HMAC SHA-1/256	FIPS 198-1 & 180-4	FCS_COP.1(4)	3440, 3439, 2939, 2938, 2937, 2936
DRBG SHA-256 HMAC_DRBG	SP 800-90A	FCS_RBG_EXT.1	1958, 1453, 1452
SHS SHA-1/256	FIPS 180-4	FCS_COP.1(2)	4188, 4187, 3644, 3643, 3642, 3641

Table 13 Samsung Kernel Cryptographic Algorithms

The Samsung SCrypto TEE library provides the following algorithms. The devices with Exynos processors use the Kinibi 400A Trusted Execution Environment (TEE), the S9/S9+ devices with Qualcomm processors use the QSEE 5.0 TEE and the S8/S8+ devices with Qualcomm processors use the QSEE 4.0 TEE. The S9/S9+ devices use SCrypto v2.2 while the S8/S8+ devices use SCrypto v2.0.

Algorithm	NIST Standard	SFR Reference	Cert#'s
AES CBC/GCM	FIPS 197, SP 800-38A/D	FCS_COP.1(1)	5180, 4389
DRBG AES-256 CTR_DRBG	SP 800-90A	FCS_RBG_EXT.1	1955, 1412
ECDSA PKG/PKV/SigGen/SigVer	FIPS 186-4	FCS_CKM.1 FCS_COP.1(3)	1343, 1049
HMAC SHA-1/256/384/512	FIPS 198-1 & 180-4	FCS_COP.1(4)	3436, 2916
RSA SigGen/SigVer	FIPS 186-4	FCS_CKM.1FCS_CKM.2(1) FCS_COP.1(3)	2781, 2372
SHS SHA-1/256/384/512	FIPS 180-4	FCS_COP.1(2)	4184, 3618

Table 14 SCrypto TEE Cryptographic Algorithms

The Chipset hardware provides the following algorithms. All devices use the same Broadcom BCM4361 Wi-Fi chipset that possesses CAVP certificate 4152. The S8/S8+ devices with Exynos processors incorporate the Samsung

FMP [v1.3 (driver) & v3.0.2 (hardware)] and Security Processing Unit (SPU) while the S9/S9+ devices with Exynos processors incorporate the Samsung FMP [v1.4 (driver) & v4.0 (hardware)] and Security Processing Unit (SPU). The devices with Qualcomm processors incorporate the QTI Inline Crypto Engine.

Algorithm	NIST Standard	SFR Reference	Cert#'s
AES 128 CCM (BCM Wi-Fi)	FIPS 197, SP 800-38C	FCS_COP.1(1)	4152
AES 128/256 XTS (Qualcomm)	FIPS 197, SP 800-38E	FCS_COP.1(1)	4958, 4957, 4473, 4472
DRBG SHA-256 Hash_DRBG (Qualcomm)	SP 800-90A	FCS_RBG_EXT.1	885, 1788
SHS SHA-256 (Qualcomm)	FIPS 180-4	FCS_COP.1(2)	2908, 2930, 4047
KBKDF (Qualcomm)	SP 800-108	FCS_CKM_EXT.3	171, 142
AES 128/256 XTS (Exynos FMP)	FIPS 197, SP 800-38E	FCS_COP.1(1)	5169, 4423
SHS SHA-256 (Exynos FMP)	FIPS 180-4	FCS_COP.1(2)	4176, 3642
HMAC SHA-256 (Exynos FMP)	FIPS 198-1 & 180-4	FCS_COP.1(4)	3430, 2940
KBKDF (Exynos SPU)	SP 800-108	FCS_CKM_EXT.3	197, 196
AES 128/256 CBC/GCM (Exynos SPU)	FIPS 197, SP 800-38A/D	FCS_COP.1(1)	5368, 5367
SHS SHA-256 (Exynos SPU)	FIPS 180-4	FCS_COP.1(2)	4310, 4309
HMAC SHA-256 (Exynos SPU)	FIPS 198-1 & 180-4	FCS_COP.1(4)	3556, 3555

Table 15 Chipset Hardware Cryptographic Algorithms

The TOE's application processors include hardware entropy implementations that supply random data within the TEE and to the Linux kernel RNG (/dev/random).

Note that kernel-space system applications utilize the cryptographic algorithm implementations in the Samsung Kernel Cryptographic Module (Kernel Crypto) or in the Chipset hardware, while user-space system applications and mobile applications utilize the BoringSSL library (through the Java API). In the case of each cryptographic library, the library itself includes any algorithms required (for example, BoringSSL provides hash functions for use by HMAC and digital signature algorithms).

Trusted Applications executing with the Trusted Execution Environment (TEE) utilize the SCrypto library. For example, the trusted application implementing the Android Keymaster that supports the Android Keystore utilizes the SCrypto library for all of its cryptographic functionality.

For its HMAC implementations, the TOE accepts all key sizes of 160, 256, 384, & 512; supports all SHA sizes save 224 (e.g., SHA-1, 256, 384, & 512), utilizes the specified block size (512 for SHA-1 and 256, and 1024 for SHA-384 & 512); and outputs MAC lengths of 160, 256, 384, and 512.

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 4096 HMAC-SHA-512 or HMAC-SHA-256 (depending on the use) iterations to combine a 128-bit salt with the user's password. The time needed to derive keying material does not impact or lessen the difficulty faced by an attacker's exhaustive guessing as the combination of the password derived KEK with REK value entirely prevents offline attacks and the TOE's maximum incorrect password login attempts (between 1 and 50 incorrect attempts with 4 character, minimum, passwords) prevents exhaustive online attacks.

The TOE's algorithm certificates represent the BoringSSL library, Kernel Cryptographic module, SCrypto library, and Chipset hardware implementations. These implementations have been tested upon Android 8.0 (Oreo) running atop both Exynos and Qualcomm ARMv8 chipsets. The TOE's Exynos processor devices include the Samsung Flash Memory Protector (FMP v1.2 on the S8/S8+, FMP v1.4 on the S9/S9+), while the TOE's Qualcomm processor devices include hardware AES XTS (Inline Crypto Engine, ICE) implementations for encryption and decryption.

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

FCS_IPSEC_EXT.1: The TOE's VPN Client implements the IPsec protocol as specified in RFC 4301; however, the VPN Client presents as few configuration options as possible to the User in order to minimize the possibility of misconfiguration and relies upon the Gateway to enforce organizational policies (for things like the specific cipher suites and selection of traffic to protect). For this reason, the VPN Client does not support editing of its SPD entries. The VPN Client will insert a PROTECT rule to IPsec encrypt and send all TOE traffic to the VPN GW (as the VPN Client ignores the IKEv1/IKEv2 Traffic Selector negotiated between the client and gateway and always sends all traffic).

The VPN Client routes all packets through the kernel's IPsec interface (ipsec0) when the VPN is active. The kernel compares packets routed through this interface to the SPDs configured for the VPN to determine whether to PROTECT, BYPASS, or DISCARD each packet. The vendor designed the TOE's VPN Client, when operating in CC Mode, to allow no configuration and to always force all traffic through the VPN. The VPN Client ignores any IKEv1/IKEv2 traffic selector negotiations with the VPN GW and will always create an SPD PROTECT rule that matches all traffic. Thus, the kernel will match all packets, subsequently encrypt those packets, and finally forward them to the VPN Gateway. The VPN Client supports tunnel mode for its IPsec connections. The VPN Client provides IKEv1/IKEv2 key establishment as part of its IPsec implementation. The IKEv1/IKEv2 implementation is conformant with RFCs 5996 and 4307 and supports NAT traversal. IKEv1 only supports main mode and requires no configuration for this enforcement.

The TOE provides RFC 4106 conformant AES-GCM-128 and AES-GCM-256, and RFC 3602 conformant AES-CBC-128 and AES-CBC-256 as encryption algorithms. The TOE Platform also provides SHA-1 (SHA-1 can only be used for IKEv2 connections), SHA-256, SHA-384, and SHA-512 in addition to HMAC-SHA1, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 as integrity/authentication algorithms (producing message digests of 160, 256, 384, and 512-bits in length) as well as Diffie-Hellman Groups 5, 14, 19, 20 and 24. The VPN utilizes the algorithms from the BoringSSL and Kernel Cryptographic modules as part of the IKEv1/IKEv2 and IPsec protocols (however, note that IKEv1 does not support SHA-1). The encrypted payload for IKEv1/IKEv2 uses AES-CBC-128, AES-CBC-256 as specified in RFC 6379 and (for IKEv2) AES-GCM-128 and AES-GCM-256 as specified in RFC 5282. The TOE relies upon the VPN Gateway to ensure that by default the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the IKEv1 Phase 1/IKEv2 /IKE_SA connection is greater than or equal to the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the IKEv1 Phase 2/IKEv2 CHILD_SA connection. Note that only RSA is supported for IKEv1. The IKEv1 implementation includes XAUTH authentication.

An administrator can configure the VPN Gateway to limit SA lifetimes based on length of time to values that include 24 hours for IKE SAs and 8 hours for IPsec SAs. The TOE includes hardcoded limits of 10 hours for an IKE SA and 3 hours for an IPsec SA. The TOE and VPN Gateway will rekey their IKE and IPsec SAs after the shorter of either 10 hours and 3 hours respectively (the TOE's fixed lifetimes) or the administrator specified lifetime configured on the VPN Gateway.

The VPN Client generates the secret value x used in the IKEv1/IKEv2 Diffie-Hellman key exchange ('x' in g^x mod p) using the FIPS validated RBG specified in FCS_RBG_EXT.1 and having possible lengths of 224, 256, or 384 bits. When a random number is needed for a nonce, the probability that a specific nonce value will be repeated during the life of a specific IPsec SA is less than 1 in 2^{112}, 2^{128}, or 2^{192}.

The VPN Client implements peer authentication using RSA certificates or ECDSA certificates (only in conjunction with IKEv2) that conform to RFC 4945 and FIPS 186-4, or pre-shared keys. If certificates are used, the VPN Client ensures that the IP address or Fully Qualified Distinguished Name (FQDN) contained in a certificate matches the expected IP Address or FQDN for the entity attempting to establish a connection and ensures that the certificate has not been revoked (using the Online Certificate Status Protocol [OCSP] in accordance with RFC 2560).

Pre-shared keys can include any letter from a-z, A-Z, the numbers 0 – 9, and the special character located above the numbers on a US keyboard ("!@#$%^&*()"). The specific length of 22 characters required by the VPNC21 is supported by the VPN Client. The VPN Client processes the pre-shared keys by concatenating the ASCII bit values.

The TOE supports a number of different Diffie-Hellman (DH) groups for use in SA negotiation including DH Groups 5 (1536-bit MODP), 14 (2048-bit MODP), 19 (256-bit Random ECP), 20 (384-bit Random ECP), and 24 (2048-bit MODP with 256-bit POS). The TOE selects the DH group by selecting the largest group configured by an administrator that is offered by the VPN gateway.

During the Peer Authentication stage of IPsec, the TOE Platform will verify the authenticity of the VPN gateway's X.509v3 certificate by validating the certificate, validating the certificate path, validating the certificate's revocation status using OCSP, validating that the certificate path terminates in a trusted CA certificate, and validating that the CA certificate has the basicConstraints extension present and the CA flag set to true. The TOE will also ensure that the Subject Alternative Name IP address or DNS name in the VPN gateway's certificate matches the IP address or DNS name configured in the VPN profile. If the configured IP address or DNS name does not match a Subject Alternative Name in the VPN gateway's certificate, the TOE will refuse to establish an IPsec connection with the VPN gateway.

The VPN Client relies upon the VPN Gateway to ensure that the cryptographic algorithms and key sizes negotiated during the IKEv1/IKEv2 negotiation ensure that the security strength of the Phase 1/IKE_SA are greater than or equal to that of the Phase 2/CHILD_SA.

FCS_IV_EXT.1: The TOE generates IVs for data storage encryption and for key storage encryption. The TOE uses AES-XTS and AES-CBC mode for data encryption and AES-GCM for key storage.

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

1. An AES-256 CTR_DRBG provided by BoringSSL. The TOE provides mobile applications access (through an Android Java API) to random data drawn from its AES-256 CTR_DRBG
2. An AES-256 CTR_DRBG provided by SCrypto
3. A SHA-256 HMAC_DRBG provided by Kernel Crypto
4. A hardware SHA-256 Hash_DRBG provided by the Qualcomm AP

The TOE ensures that it initializes each RBG with sufficient entropy ultimately accumulated from a TOE-hardware-based noise source. The TOE uses its hardware-based noise source to continuously fill /dev/random with random data that has full entropy, and in turn, the TOE draws from /dev/random to seed both its AES-256 CTR_DRBG and its SHA-256 HMAC_DRBG. The TOE seeds each of its AES-256 CTR_DRBGs using 384-bits of data from /dev/random, thus ensuring at least 256-bits of entropy. Finally, the TOE seeds the SHA-256 HMAC_DRBG with 440-bits of data from /dev/random, also ensuring that it contains at least 256-bits of entropy. These RBGs are all capable of providing other amounts of entropy (such as 128-bits) to any requesting application. The TOE itself always uses 256-bits of entropy, but other applications or services are not subject to this limitation, and can request 128-bits or 256-bits of entropy subject its own requirements.

FCS_RBG_EXT.2: The devices save a block of 4096-bits of random data, and upon the next boot, a service called entropy mixer adds the block of saved random data into the Linux Kernel Random Number Generator's input pool.

FCS_SRV_EXT.1/2: 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-512), generating asymmetric keys for key establishment (RSA and ECDH), and generating asymmetric keys for signature generation and verification (RSA, ECDSA). The TOE provides access through the Android operating system's Java API methods and through the kernel. The vendor also developed testing applications to enable execution of the NIST algorithm testing suite in order to verify the correctness of the algorithm implementations.

FCS_STG_EXT.1: The TOE provides users and applications running on the TOE the ability to generate, import, and securely store symmetric and asymmetric keys through the TOE's Android Keystore. The TOE allows a user to import a certificate (in PKCS#12 [PFX] format) and provides applications running on the TOE an API to import a certificate or secret key. In either case, the TOE will place the key into the user's keystore (and the TOE will remove the PKCS#12 password-based protection if the imported key is a certificate) and doubly encrypt the imported key with DEKs, which in turn are encrypted by a KEK derived from the user's DKEK and a KEK derived from the REK. All user and application keys placed into the user's keystore are secured in this fashion. The TOE also encrypts (using ODE) the Flash file system in which all keystore blobs reside (as encrypted files), providing yet another layer of encryption protection to keystore objects.

The user of the TOE can elect to delete keys from the keystore, as well as to securely wipe the entire device.

The TOE affords applications control (control over use and destruction) of keys that they create or import, and only the user or a common application developer can explicitly authorize access, use, or destruction of one application's key by any other application.

Entity	Can import?	Can destroy?	Allow other app to use?	Allow other app to destroy
User	Yes	Yes	Yes	
Administrator	Yes	Yes		
Mobile application	Yes	Yes		
Common application developer			Yes	Yes

Table 16 Key Management Matrix

FCS_STG_EXT.2: The TOE provides protection for all stored keys (i.e. those written to storage media for persistent storage) chained to both the user's password and the REK. All keys are encrypted with AES-GCM or AES-CBC (in the case of SD card File Encryption Keys). All KEKs are 256-bit, ensuring that the TOE encrypts every key with another key of equal or greater strength/size.

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

FCS_STG_EXT.3: The key hierarchy shows AES-256 GCM is used to encrypt all KEKs other than SD Card keys (which uses HMAC for integrity) and the GCM encryption mode itself ensures integrity as authenticated decryption operations fail if the encrypted KEK becomes corrupted.

FCS_TLSC_EXT.1/2: The TOE provides mobile applications (through its Android API) the use of TLS version 1.2 including support for the selected ciphersuites in the selections in section 5.1.2.28. 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 the evaluated elliptic curves (P-256 and P-384) and requires/allows no configuration of the supported curves. The TOE supports the use of wildcards in X.509 reference identifiers (CN and SAN), and the TOE supports certificate pinning.

FCS_TLSC_EXT.1/WLAN & FCS_TLSC_EXT.2/WLAN: The TSF supports TLS versions 1.0, 1.1, and 1.2 with client (mutual) authentication and supports the ciphersuites in the selections in section 5.1.2.29 for use with EAP-TLS as part of WPA2. The TOE, by design, supports the evaluated elliptic curves (P-256 and P-384) and requires/allows no configuration of the supported curves.

6.3 User data protection

FDP_ACF_EXT.1/2/3:

The TOE provides protection for high-level services like location, email, calendar in addition to providing individual permissions to which mobile applications can request access.

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

1. normal – A lower-risk permission that gives an application access to isolated application-level features, with minimal risk to other applications, the system, or the user. The system automatically grants this type of permission to a requesting application at installation, without asking for the user's explicit approval (though the user always has the option to review these permissions before installing).
2. dangerous – A higher-risk permission that would give a requesting application access to private user data or control over the device that can negatively impact the user. Because this type of permission introduces potential risk, the system may not automatically grant it to the requesting application. For example, any dangerous permissions requested by an application may be displayed to the user and require confirmation before proceeding, or some other approach may be taken to avoid the user automatically allowing the use of such facilities.

3. signature – A permission that the system is to grant only if the requesting application is signed with the same certificate as the application that declared the permission. If the certificates match, the system automatically grants the permission without notifying the user or asking for the user's explicit approval.
4. signatureOrSystem – A permission that the system is to grant only to packages in the Android system image *or* that are signed with the same certificates. Please avoid using this option, as the signature protection level should be sufficient for most needs and works regardless of exactly where applications are installed. This permission is used for certain special situations where multiple vendors have applications built in to a system image which need to share specific features explicitly because they are being built together.

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

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

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

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

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

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

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

requested services. Thereafter (each time the application is run), the TOE will grant the application access to the services specified during install.

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

While Android provides a large number of individual permissions, they are generally grouped into categories or features that provide similar functionality. **Table 17** shows a series of functional categories centered around common functionality.

Service Features	Description
Sensitive I/O Devices & Sensors	Location services, Audio & Video capture, Body sensors
User Personal Information & Credentials	Contacts, Calendar, Call logs, SMS
Metadata & Device ID Information	IMEI, Phone Number
Data Storage Protection	SD Card, App data, App cache
System Settings & Application Management	Date time, Reboot/Shutdown, Sleep, Force-close application, Administrator Enrollment
Wi-Fi, Bluetooth, USB Access	Wi-Fi, Bluetooth, USB tethering, debugging and file transfer
Mobile Device Management & Administration	MDM APIs
Peripheral Hardware	NFC, Camera, Headphones
Security & Encryption	Certificate/Key Management, Password, Revocation rules

Table 17 Access Control Categories

FDP_PBA_EXT.1: The TOE requires the user to enter their password to enroll, re-enroll or un-enroll any biometric templates. When the user attempts biometric authentication to the TOE, the biometric sensor takes an image of the presented biometric for comparison to the enrolled templates. The captured image is compared to all the stored templates on the device to determine if there is a match. The complete biometric authentication process is handled inside the TEE (including image capture, all processing and match determination). The image is provided to the biometric service to check the enrolled templates for a match to the captured image.

FDP_DAR_EXT.1: The TOE provides AES-256 XTS encryption of all data (which includes both user data and TSF data) stored on the TOE through On Device Encryption (ODE) of the data partition. The TOE also has TSF data for its ODE keys, which the TOE stores outside the ODE encrypted data partition, and the TOE also includes a read-only file system in which the TOE's system executables, libraries, and their configuration data reside. For its ODE encryption of the data partition (where the TOE stores all user data and all application data), the TOE uses an AES-256 bit DEK with XTS mode to encrypt the entire partition. The TOE also provides AES-256 CBC encryption of protected data stored on the external SD Card using FEKs. The TOE encrypts each individual file stored on the SD Card, generating a unique FEK for each file.

FDP_DAR_EXT.2: The TOE, as part of the Knox Platform for Enterprise, provides mobile applications the ability to store sensitive data and have the TOE encrypt it accordingly. Based on the Work environment lock-state, sensitive data protected by this mechanism will be encrypted when locked (either directly by the user via timeout). An application can determine whether sensitive data should remain encrypted in this manner or if it should be re-encrypted (such as by a symmetric key for better performance). Applications can use this to securely receive data while the Work environment is locked (such as an email application).

FDP_IFC_EXT.1: The TOE supports the installation of VPN Client applications, which can make use of the provided VPN APIs in order to configure the TOE's routing functionality to direct all traffic through the VPN. The TOE also includes an IPsec VPN Client that ensures all traffic other than traffic necessary to establish the VPN connection (for example, ARP, 802.11-2012 traffic, IKEv1, and IKEv2) flows through the VPN. The TOE routes all packets through the kernel's IPsec interface (ipsec0) when the VPN is active. The kernel compares packets routed through this interface to the SPDs configured for the VPN to determine whether to PROTECT, BYPASS, or DISCARD each packet. The vendor developed the TOE's VPN, when operating in CC Mode, to allow no configuration and to always force all traffic through the VPN. The TOE ignores any IKEv2 traffic selector

negotiations with the VPN GW and will always create an SPD PROTECT rule that matches all traffic. Thus, the kernel will match all packets, subsequently encrypt those packets, and finally forward them to the VPN Gateway.

FDP_RIP.1: The TOE has been designed to ensure that no residual information exists in network packets when the VPN is turned on. When the TOE allocates a new buffer for either an incoming or outgoing a network packet, the new packet data will be used to overwrite any previous data in the buffer. If an allocated buffer exceeds the size of the packet, any additional space will be overwritten (padded) with zeros before the packet is forwarded (to the external network or delivered to the appropriate internal application).

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 consists of any additional user or admin/MDM loaded certificates. Disabled default certificates and user added certificates reside in the `/data/misc/user/0/cacerts-removed` and `/data/misc/user/0/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 certificates 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 IPsec, TLS, HTTPS, Bluetooth BR/EDR and Bluetooth LE. Mobile applications can use the following Android APIs for IPsec, TLS, HTTPS, and Bluetooth respectively:

`android.net.VpnService`

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

`com.samsung.android.knox.net.vpn`

https://seap.samsung.com/api-references/android/reference/com/samsung/android/knox/net/vpn/package-summary.html

`javax.net.ssl.SSLContext`:

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

`javax.net.ssl.HttpsURLConnection`:

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

`android.bluetooth`:

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

6.4 Identification and authentication

FIA_AFL_EXT.1: The TOE maintains, 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 users' data). An administrator can adjust the number of failed logins from the (CC Mode) limit of ten failed password logins to a value between one and 50 through an MDM. 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) by storing the number of logins remaining within its Flash file system. The TOE also separately stores (in a different Flash location) the number of master user (user_0) logins attempts remaining for the crypt-lock screen (before the TOE will wipe itself). This value defaults to a vendor specified number (typically set to 10), and prior to the first unlock after power-up, the user data partition is encrypted and the TOE does not have access to data stored within it.

For users with biometrics enabled, biometric authentication attempts are maintained along with the password attempts.

The maximum number of incorrect password authentication attempts can be configured to a value between 1 and 50. The maximum number of biometric attempts is 10 (and cannot be changed). The user can attempt 10 biometric attempts followed by the max value password attempts before the TOE will wipe itself. For example, if the counter were set to 15, 10 biometric attempts would be followed by 15 password attempts before the device was wiped. The last attempt must always be a password attempt as the biometric is a non-critical mechanism.

The hybrid authentication method requires the user to authenticate with a biometric and a password in sequential order. The combined entries are counted as a single attempt, and an incorrect match of either part will be counted as a failed attempt. To login, the user must first enter his or her biometric (a timeout is enforced upon successive failed password attempts) and upon successfully verifying the user's biometric, the user must enter his or her password. Even though the TOE hybrid authentication mechanism does not accept a combination of password and biometric sample (with both to pass, without the user being made aware of which factor failed, should either fail), the TOE's design ensures that no more than the configured maximum number of attempts is possible. This prevents disclosing whether an authentication factor is correct or incorrect without also incrementing the failed attempt counter. The hybrid mechanism is not a critical authentication mechanism.

The TOE validates passwords by providing them to Android's Gatekeeper (which runs in the Trusted Execution Environment), and if the presented password fails to validate, the TOE increments the failed attempt counter before displaying a visual error to the user. The TOE validates biometric attempts through the biometric service (which runs in the Trusted Execution Environment), and if the presented biometric does not match the registered templates, the TOE increments the failed attempt counter before displaying a visual error to the user.

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

FIA_BLT_EXT.2: The TOE requires explicit user authorization or user authorization before data transfers over the link. When transferring data with another device, the TOE requires that the user must confirm an authorization popup displayed allowing data transfer (Obex Object Push) or confirm the user passkey displayed numeric passcode matches between the two devices (RFCOMM).

FIA_BLT_EXT.3: The TOE tracks active connections and actively ignores connection attempts from Bluetooth device addresses for which the TOE already has an active connection.

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

FIA_BLT_EXT.6: The TOE supports OPP and MAP profile on a per service basis (as opposed to a per app basis).

FIA_BMG_EXT.1:

The TOE provides options for fingerprint and iris biometric authentication. The user must always have a password, even when a biometric authentication method is enabled. Biometric authentication attempts are limited though there are differences based on the devices. The user can switch between mechanisms at any time though the counters will not reset between modality switches. All biometric attempts are enumerated in one counter with a maximum of 10 total attempts before no more attempts are allowed.

In the evaluated configuration, the maximum number of authentication attempts is 60 before a wipe event is triggered. This means that on all devices up to 10 biometric attempts and 50 password attempts could be made before a wipe occurs. Using this as the worst-case scenario leads to a maximum of 10 biometric attempts that can be made for the SAFAR calculations. All devices or configurations provide for fewer attempts (both password and biometric), and so any resulting SAFAR would be lower than this scenario. The last attempt before a wipe must be a password attempt.

For a password-only configuration, the SAFAR claim would be 1:1,000,000 when set for 10 attempts. The password minimum length is 4 characters and there are 93 possible characters that can be used in the password. This is not claimed since this configuration (i.e. no biometric authentication allowed) would not require FIA_BMG_EXT.1, but is shown here for the overall SAFAR calculations.

$$SAFAR_{password} = 1 - \left(1 - {}^{1}/_{93}{}^{4}\right)^{50} = 6.684 * 10^{-7}$$

The FAR for fingerprint and iris is 1:10,000 and the FRR is 1:20. The SAFAR when a fingerprint or iris is in use is 1:1,000 based on a maximum of 10 attempts of the biometric as noted in the worst-case scenario.

$$SAFAR_{biometric} = 1 - (1 - 10^{-4})^{10} = 9.996 * 10^{-4}$$

The SAFAR calculations for a password being the critical factor are the same as the $SAFAR_{any}$ calculations detailed in MDFPP31 section H.4. This value is accepted because it is within the 1% margin allowed by the PP.

$$SAFAR_{any} = 1 - (1 - SAFAR_{biometric}) * (1 - SAFAR_{password})$$

$$SAFAR_{any} = 1.00021785 * 10^{-3}$$

The Knox Platform for Enterprise provides hybrid (multi-factor: password and fingerprint or password and iris) authentication. When using the hybrid authentication mechanism, the user must enter either a correct biometric factor (or 10 failed biometric factors) and then a correct password in order to successfully unlock the container.

The SAFAR when a hybrid mechanism is in use is equal to $SAFAR_{password}$ as whether or not one enters a correct biometric, one must always enter a correct password. The password minimum length is 4 characters and there are 93 possible characters that can be used in the password.

$$SAFAR_{hybrid} = SAFAR_{password}$$

The biometric FAR/FRR values are tested internally by two independent groups using different methodologies. The first set of tests are "offline" in that a specially configured device is connected to a test harness and used to enroll biometric samples for storage. The test harness then uses the samples to run through numerous combinations of the samples to determine FAR/FRR results that are used to tune the algorithms controlling the biometric system. Once testing is complete a second set of tests are performed in an "online" manner where the testing is done with users directly testing on a live device.

All devices with different hardware combinations that could impact the biometric subsystem are tested. For example, both the Exynos and Qualcomm versions of the mobile devices are tested individually since the TrustZone components in each device are different. These tests are integrated into the production process and so are repeated continually during the development process.

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

FIA_PMG_EXT.1: The TOE authenticates the user through a password consisting of basic Latin characters (upper and lower case, numbers, and the special characters noted in the selection (see section 5.1.4.10)). The TOE can support a minimum password length of as few as four characters and a maximum of no more than sixteen characters. The TOE defaults to requiring passwords to have a minimum of four characters that contain at least one letter and one number. 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, and non-letters).

FIA_PSK_EXT.1: The TOE supports the use of pre-shared keys (the TOE allows 1 to 64 character PSKs) for IPsec VPNs. The usage of pre-shared keys within the IPsec implementation is described in Section 6.2.

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-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.5: The TOE, in CC mode, allows the user to authenticate to the device using a password, iris, fingerprint, or hybrid mechanism depending on the device configuration. The TOE prohibits other authentication mechanisms such as pattern or PIN. Use of Smart Lock mechanisms (on-body detection, trusted places, trusted devices, trusted face, trusted voice) can be blocked through management controls. Upon restart or power-on the user can only use a password for authentication. Once past this authentication screen the user is able to use one of the configured methods at the Android lock screen to login.

FIA_UAU.6(1)/FIA_UAU.6(2): The TOE requires the user to enter their password or supply their biometric 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. The TOE can disable Smart Lock through management controls. Only after entering their current user password can the user then elect to change their password.

FIA_UAU.7: The TOE allows the user to enter the user's password from the lock screen. The TOE will, by default, display the most recently entered character of the password briefly or until the user enters the next character in the password, at which point the TOE obscures the character by replacing the character with a dot symbol. 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: The TOE's Key Hierarchy requires the user's password in order to derive the PKEK1 & PKEK2 keys in order to decrypt other KEKs and DEKs. Thus, until it has the user's password, the TOE cannot decrypt the DEK utilized by ODE to decrypt protected data.

FIA_UAU_EXT.2: The TOE, when configured to require user authentication (as is the case in CC mode), allows only those actions described in section 5.1.4.18. 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.4: The TOE requires a separate password or hybrid authentication for its container, thus protecting all Enterprise application data and shared resource data. The user must enter either their password or both their password and biometric (if the user has configured hybrid container authentication and enrolled a biometric) in order to access any of the Enterprise application data.

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 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 is 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 extendedKeyUsage field]), then verifies each certificate in the chain (applying the same rules as above, but also ensuring that the Issuer of each certificate matches the Subject in the next rung "up" in the chain and that the chain ends in a self-signed certificate present in either the TOE's trusted anchor database or matches a specified Root CA), and finally the TOE performs revocation checking for all certificates in the chain.

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

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 will not establish the trusted connection if the peer certificate cannot be verified (including validity, certification path, and revocation [through CRL and OCSP]).

The VPN requires that for each VPN profile, the user specify the client certificate the TOE will use (the certificate must have been previously imported into the keystore) and specify the CA certificate to which the server's certificate must chain. The VPN thus uses the specified certificate when attempting to establish that VPN connection. The VPN processes a connection to a server by first comparing the Identification (ID) Payload received from the server against the certificate sent by the server, and if the DN of the certificate does not match the ID, then the TOE does not establish the connection.

During revocation checking (for any type of connection), the TOE first attempts to determine a certificate's revocation status through OCSP (if the Authority Information Access, AIA, extension is present). If the certificate lacks AIA or if the OCSP server does not respond, or if the OCSP responder returns an unknown status; the TOE attempts to determine revocation status using CRLs, if the certificate includes a CRL Distribution Point (CDP). If the TOE cannot establish a connection with the server acting as the CDP, the TOE will deem the server's certificate as invalid and not establish a TLS connection with the server. Note that the VPN only checks OCSP for revocation.

FIA_X509_EXT.3: The TOE's Android operating system provides applications the `java.security.cert.CertPathValidator` Java API Class of methods for validating certificates and certification paths (certificate chains establishing a trust chain from a certificate to a trust anchor). This class is also recommended to be used by third-party Android developers for certificate validation. However, `TrustedCertificateStore` must be used to chain certificates to the Android System Trust Anchor Database (anchors should be retrieved and provided to `PKIXParameters` used by `CertPathValidator`). The available APIs may be found here:

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

6.5 Security management

FMT_MOF.1/FMT_SMF_EXT.1: The TOE provides the management functions described in **Table 5**. The table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted.

FMT_SMF_EXT.1/WLAN: The TOE provides the management functions described in **Table 5** Rows 48-55. The table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted. It is worth noting that the TOE's ability to specify authorized application repositories takes the form of allowing enterprise applications (i.e., restricting applications to only those applications installed by an MDM Agent).

FMT_SMF_EXT.2: The TOE disables CC mode when unenrolled (note that removing a container will not disable CC mode). The TOE also provides the capability to the MDM Agent to wipe all protected data and to remove any installed Enterprise applications. The TOE can distinguish between apps installed inside a Work environment and can provide the capability of removing apps and data within the Work environment when unenrolled from the MDM.

FMT_SMF_EXT.3: The TOE provides the user with the ability to see all apps installed on the device that have administrative capabilities. Each app listing also shows the status of the app privileges for administration (enabled or disabled) and the permissions the app has on the device.

FMT_SMF.1/VPN: The TOE provides the management functions described in **Table 5** Rows 56-61. In addition, the VPN Gateway, acting as administrator, can specify the IKE algorithms, protocols and authentication techniques, and the crypto period for session keys.

The TOE provides users the ability to specify an X.509v3 certificate (previously loaded into the TOE Platform's keystore) for the TOE to use to authenticate to the VPN gateway during IPsec peer authentication as well as an X.509v3 certificate to use as the CA certificate. The TOE alternatively provides users the ability to enter a Pre-Shared Key to be used in lieu of an X.509v3 certificate during IPsec peer authentication.

6.6 Protection of the TSF

FPT_AEX_EXT.1: The Linux kernel of the TOE's Android operating system provides address space layout randomization utilizing a non-cryptographic kernel random function to provide 8 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/FPT_AEX_EXT.6: The TOE's Android 8.0 operating system utilizes 4.4/4.9 Linux kernels, whose memory management unit (MMU) enforces read, write, and execute permissions on all pages of virtual memory and ensures that write and execute permissions are not simultaneously granted on all memory (exceptions are only made for Dalvik JIT compilation). The Android operating system (as of Android 2.3) sets the ARM eXecute Never (XN) bit on memory pages and the TOE's ARMv8 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 D.5 of the ARM v8 Architecture Reference Manual contains additional details about the MMU of ARM-based processors: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0487a.f/index.html.

FPT_AEX_EXT.3: The TOE's Android operating system provides explicit mechanisms to prevent stack buffer overruns (enabling -fstack-protector) in addition to taking advantage of hardware-based No eXecute to prevent code execution on the stack and heap. Samsung requires and 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's Secure Boot process employs a series of public keys to form a chain of trust that operates as follows. The Application Processor (AP) contains the SHA-256 hash of the Secure Boot Public Key (an RSA 2048-bit key embedded in the end of the signed bootloader image), and upon verifying the SBPK attached to the bootloader produces the expected hash, the AP uses this public key to verify the PKCS 1.5 RSA 2048 w/ SHA-256 signature of the bootloader image, to ensure its integrity and authenticity before transitioning execution to the bootloader. The bootloader, in turn, contains the Image Signing Public Key (ISPK), which the bootloader will use to verify the PKCS 1.5 RSA 2048 w/ SHA-1 signature on either kernel image (primary kernel image or recovery kernel image). Note that when configured for Common Criteria mode, the TOE only accepts updates to the TOE Firmware Over The Air; however, when not configured for CC mode, the TOE allows updates through the bootloader's ODIN mode. The primary kernel includes an embedded FOTA Public Key, which the TOE uses to verify the authenticity and integrity of Firmware Over The Air update signatures (which contain a PKCS 2.1 PSS RSA 2048 w/ SHA-512 signature).

The TOE protects access to the REK and derived HEK to only trusted applications[2] within the TEE (TrustZone). The TOE key manager includes a TEE module which utilizes the HEK to protect all other keys in the key hierarchy. All TEE applications are cryptographically signed, and when invoked at runtime (at the behest of an untrusted application), the TEE will only load the trusted application after successfully verifying its cryptographic signature. Furthermore, the device encryption library checks the integrity of the system by checking the result from both Secure Boot/SecurityManager and from the Integrity Check Daemon before servicing any requests. Without this TEE application, no keys within the TOE (including keys for ScreenLock, the keystore, and user data) can be successfully decrypted, and thus are useless.

The third protection is the TOE's internal SecurityManager watchdog service. The SecurityManager manages the CC mode of the TOE by looking for unsigned kernels or failures from other, non-cryptographic checks on system

[2] A TrustZone application is a trusted application that executes in a hardware-isolated domain.

integrity, and upon detection of a failure in either, disables the CC mode and notifies the TEE application. The TEE application then locks itself, again rendering all TOE keys useless.

Finally, the TOE's Android OS provides "sandboxing" that ensures each non-system 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 (it is possible for two processes to utilize shared memory, but not directly access the memory of another application) or files and cannot access the memory space or files of system-level applications.

FPT_AEX_EXT.5: The TOE provides Kernel Address Space Layout Randomization to ensure that the base address of kernel-space memory mappings consist of 4 unpredictable bits.

FPT_BBD_EXT.1: The TOE's hardware and software architecture ensures separation of the application processor (AP) from the baseband or communications processor (CP). From a software perspective, the AP and CP communicate logically through the Android Radio Interface Layer (RIL) daemon. This daemon, which executes on the AP, coordinates all communication between the AP and CP. It makes requests of the CP and accepts the response from the CP; however, the RIL daemon does not provide any reciprocal mechanism for the CP to make requests of the AP. Because the mobile architecture provides only the RIL daemon interface, the CP has no method to access the resources of the software executing on the AP.

FPT_JTA_EXT.1: The TOE prevents access to its processor's JTAG interface by only enabling JTAG when the TOE has a special image written to its bootloader/TEE partitions. That special image must be signed by the appropriate key (corresponding to the public key that has its SHA-256 hash programmed into the processor's fuses).

FPT_KST_EXT.1: The TOE does not store any plaintext key material in its internal Flash; 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 is removed), all keys in internal Flash are wrapped with a KEK. Please refer to section 6.2 of the TSS for further information (including the KEK used) regarding the encryption of keys stored in the internal Flash. As the TOE encrypts all keys stored in Flash, upon boot-up, the TOE must first decrypt and utilize keys. Note as well that the TOE does not use a user's biometric fingerprint to encrypt/protect key material. Rather the TOE always requires the user enter his or her password after a reboot (in order to derive the necessary keys to decrypt the user data partition and other keys in the key hierarchy).

FPT_KST_EXT.2: The TOE utilizes a cryptographic library consisting of an implementation of BoringSSL, the kernel crypto module, the SCrypto module, and the following system-level executables that utilize KEKs: dm-crypt, eCryptfs, wpa_supplicant, and the keystore.

The TOE ensures that plaintext key material is not exported by not allowing the REK to be exported and by ensuring that only authenticated entities can request utilization of 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). Note again that the TOE does not use the user's biometric fingerprint to encrypt/protect key material (and instead only relies upon the user's password).

FPT_KST_EXT.3: The TOE does not provide any way to export plaintext DEKs or KEKs (including all keys stored in the keystore) as the TOE chains all KEKs to the HEK/REK.

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

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

FPT_TST_EXT.1: The TOE performs known answer power on self-tests (POST) on its cryptographic algorithms to ensure that they are functioning correctly. The kernel itself performs known answer tests on its cryptographic algorithms to ensure they are working correctly and the SecurityManager service invokes the self-tests of BoringSSL at start-up to ensure that those cryptographic algorithms are working correctly. The Chipset hardware performs a power-up self-test to ensure that its AES implementation is working as does the TEE SCrypto cryptographic library. Should any of the tests fail, the TOE will reboot to see if that will clear the error.

Algorithm	Implemented in	Description
AES encryption/ decryption	BoringSSL, SCrypto, Kernel Crypto, Chipset hardware	Comparison of known answer to calculated valued
ECDH key agreement	BoringSSL	Comparison of known answer to calculated valued
DRBG random bit generation	BoringSSL, SCrypto, Kernel Crypto	Comparison of known answer to calculated valued
ECDSA sign/verify	BoringSSL, SCrypto	Sign operation followed by verify
HMAC-SHA	BoringSSL, SCrypto, Kernel Crypto	Comparison of known answer to calculated valued
RSA sign/verify	BoringSSL, SCrypto	Comparison of known answer to calculated valued
SHA hashing	BoringSSL, SCrypto, Kernel Crypto	Comparison of known answer to calculated valued

Table 18 Power-up Cryptographic Algorithm Self-Tests

FPT_TST_EXT.1/WLAN & FPT_TST_EXT.1/VPN: The TOE platform performs the previously mentioned self-tests to ensure the integrity of the WLAN client (wpa_supplicant) and the VPN client (libcharon.so) in addition to the cryptographic libraries used by the clients.

FPT_TST_EXT.2(1)/FPT_TST_EXT.2(2): The TOE ensures a secure boot process in which the TOE verifies the digital signature of the bootloader software for the Application Processor (using a public key whose hash resides in the processor's internal fuses) before transferring control. The bootloader, in turn, verifies the signature of the Linux kernel (either the primary or the recovery kernel) it loads.

The TOE also uses dm-verity in EIO mode to protect the integrity of the system partition. After verifying the digital signature of the dm-verity hash tree (using the same public key that verifies the kernel image), the TOE will reject the loading of filesystem blocks where the integrity does not match, and return an I/O error (as if the block were unreadable).

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 security software version) and hardware (model and version). Additionally, the TOE provides users the ability to review the currently installed apps (including 3rd party "built-in" applications) and their version.

FPT_TUD_EXT.2: When in CC mode, the TOE verifies all updates to the TOE software using a public key (FOTA public key) chaining ultimately to the Secure Boot Public Key (SBPK), a hardware protected key whose SHA-256 hash resides inside the application processor (note that when not in CC mode, the TOE allows updates to the TOE software through ODIN mode of the bootloader). After verifying an update's FOTA signature, the TOE will then install those updates to the TOE. The TOE will check a new image to ensure that the image is not older than the current image, and if so, the TOE will reject the new image and not update the TOE software.

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

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

ALC_TSU_EXT: Samsung utilizes industry best practices to ensure their devices are patched to mitigate security flaws. Samsung provides customers with a developer web portal (http://developer.samsung.com/notice/How-to-Use-the-Forum) in which one can ask questions as well as inform Samsung of potential issues with their software (as

Samsung moderators monitor the forums), and as an Android OEM, also works with Google on reported Android issues (http://source.android.com/source/report-bugs.html) to ensure customer devices are secure.

Samsung will create updates and patches to resolve reported issues as quickly as possible, at which point the update is provided to the wireless carriers. The delivery time for resolving an issue depends on the severity, and can be as rapid as a few days before the carrier handoff for high priority cases. The wireless carriers perform additional tests to ensure the updates will not adversely impact their networks and then plan device rollouts once that testing is complete. Carrier updates usually take at least two weeks to as much as two months (depending on the type and severity of the update) to be rolled out to customers. However, the Carriers also release monthly Maintenance Releases in order to address security-critical issues, and Samsung itself maintains a security blog (http://security.samsungmobile.com) in order to disseminate information directly to the public.

Samsung communicates with the reporting party to inform them of the status of the reported issue. Further information about updates is handled through the carrier release notes. Issues reported to Google directly are handled through Google's notification processes.

6.7 TOE access

FTA_SSL_EXT.1: The TOE transitions to its locked state either immediately after a user initiates a lock by pressing the power button or after a 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 allows a user to perform the functions listed in section 5.1.7.1 before authenticating. But without authenticating first, a user cannot perform any related actions based upon these notifications (for example, they cannot respond to emails, calendar appointment requests, or text messages) other than answering an incoming phone call.

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

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

FTA_WSE_EXT.1: The TOE allows an administrator to specify (through the use of an MDM) a list of wireless networks (SSIDs) to which the user may direct the TOE to connect. 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 user may scan for available wireless network (or directly enter a specific wireless network), and then connect. Once a user has connected to a wireless network, the TOE will automatically reconnect to that network when in range and the user has enabled the TOE's Wi-Fi radio.

6.8 Trusted path/channels

FTP_ITC_EXT.1/FTP_ITC_EXT.1/WLAN: 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 IPsec. The TOE permits itself and applications to initiate communications via the trusted channel, and the TOE initiates communication 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. The TOE also meets the PP-Module for Virtual Private Network (VPN) Clients.

6.9 Knox Workspace Container Functionality

To differentiate the functionality provided specifically by a Knox Workspace container, this section enumerates the functionality provided by the container.

FDP_ACF_EXT.1.2: The TOE, through a combination of Android's multi-user capabilities and Security Enhancements (SE) for Android, provides the ability to create an isolated container within the device. Within a container a group of applications can be installed, and access to those applications is then restricted to usage solely

within the container. The container boundary restricts the ability of sharing data such that applications outside the container cannot see, share or even copy data to those inside the container and vice versa. Exceptions to the boundary (such as allowing a copy operation) must be configured by the administrator via policy. Furthermore, the container boundary policy can control access to hardware features, such as the camera or microphone, and restrict the ability of applications within the container to access those services.

FIA_AFL_EXT.1: The Knox Workspace container maintains, in Flash, the number of failed logins since the last successful login, and upon reaching the maximum number of incorrect logins, the Knox Workspace container performs a full wipe of data protected by Knox (i.e. data inside the container). An administrator can adjust the number of failed logins for the hybrid and password login mechanism from the default of ten failed logins to a value between one and fifty through an MDM.

FIA_UAU.5: The Knox Workspace container allows the user to authenticate using a password, or a hybrid method requiring both a biometric and the password at the same time. The TOE prohibits other authentication mechanisms, such as pattern, PIN, or biometric by themselves.

FIA_UAU.6(1)/FIA_UAU6(2): The Knox Workspace container requires the user to enter their password in order to unlock the Knox Workspace container. Additionally the Knox Workspace container requires the user to confirm their current password when accessing the "Knox Settings -> Knox unlock method" menu in the Knox Workspace container's user interface. Only after entering their current user password can the user then elect to change their password.

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

FMT_MOF.1/FMT_SMF_EXT.1: The Knox Workspace container grants the admin additional controls over the container beyond those available to the device as a whole. The primary additional control is over sharing data to and from the container.

The control over the camera and microphone within the Knox Workspace container only affects access to those resources inside the container, not outside the container. If either of these is disabled outside the container then they will not be available within the container, even if they are enabled.

In general the management functions for the Knox Workspace container are the same as those of the device as a whole. Specific differences of the impact of a Knox Workspace container function are noted in **Table 5 Security Management Functions**.

FTA_SSL_EXT.1: The Knox Workspace container transitions to its locked state either immediately after a user initiates a lock by pressing the container lock button from the notification bar or after a configurable period of inactivity, and as part of that transition, the Knox Workspace container will display a lock screen to obscure the previous contents. When the Knox Workspace container is locked, it can still display calendar appointments and other notifications allowed by the administrator to be shown outside the container (in the notification area). But without authenticating first to the Knox Workspace container, a user cannot perform any related actions based upon these container notifications (they cannot respond to emails, calendar appointments, or text messages).

The Knox Workspace container timeout is independent of the TOE timeout and as such can be set to different values.

7. TSF Inventory

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

Name	Path	Security Function
app_process	system/bin	APP
charon	system/bin	VPN
dalvikvm	system/bin	VM
gatekeeperd	system/bin	Keystore/Key Mgmt
icd	system/bin	Integrity
keystore	system/bin	Keystore
libcharon.so	system/lib64	VPN
libcrypto.so	system/lib, system/lib64	Crypto
libjavacrypto.so	system/lib, system/lib64	Crypto JNI
libkeystore_binder.so	system/lib, system/lib64	Keystore
libmdf.so	system/lib, system/lib64, system/vendor/lib, system/vendor/lib64	CCMode
libsec_ode_keymanager.so	system/lib64	DAR
libsecure_storage.so	system/vendor/lib, system/vendor/lib64	DAR
libsecure_storage_jni.so	system/vendor/lib, system/vendor/lib64	DAR
mcDriverDaemon	system/vendor/bin	Trustzone Daemon
mdf_fota	system/bin	Trustzone Daemon
qseecomd	system/vendor/bin	DAR
secure_storage_daemon	system/bin	FOTA
time_daemon	system/vendor/bin	Time
vold	system/bin	DAR
wpa_supplicant	system/vendor/bin/hw	WLAN

Table 19 TSF Files Inventory