# Samsung Electronics Co., Ltd. Samsung Galaxy Devices on Android 12 – Fall Security Target

Version: 0.4
2022/10/24

***Prepared for:***

SAMSUNG

**Samsung Electronics Co., Ltd.**

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

***Prepared By:***

Gossamer Laboratories

www.gossamersec.com

# Table of Contents

# 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 12 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)

***<u>Acronyms and Terminology</u>***

| | |
|---|---|
| AA | Assurance Activity |
| ACVP | Automated Cryptographic Validation Protocol |
| BAF | Biometric Authentication Factor |
| CAVP | Cryptographic Algorithm Validation Program |
| CC | Common Criteria |
| CCEVS | Common Criteria Evaluation and Validation Scheme |
| EAR | Entropy Analysis Report |
| GUI | Graphical User Interface |
| MDM | Mobile Device Management |
| NFC | Near Field Communication |
| PAD | Presentation Attack Detection |
| PAI | Presentation Attack Instrument |
| 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

### ***Glossary***

| | |
|---|---|
| Device Lock Screen<br>Android Lock Screen | The Device Lock Screen is the Android OS lock screen. |
| File-Based Encryption (FBE) | FBE allowed files to be encrypted with different keys and unlocked individually based on different authentication/access controls. This is implemented as part of the ext4 or f2fs file system (using fscrypt). |
| Firmware Over-the-air (FOTA) | Firmware Over-the-air is a term for the process of updating the firmware (operating system and services) on the device via a wireless connection as opposed to a wired (i.e. USB) connection. |
| Personal Profile | The personal profile is the common Android user on the device, such as when the device is first setup. While the name implies this space is for the end user and not the Enterprise, this can be configured as needed by the MDM. This is in contrast to a work profile. |
| Work Profile | The work profile is a second profile on a device created specifically by the MDM and used to segment enterprise data from what may be personal data. The work profile maintains separate authentication to access any apps or data stored within. |

***Table 1 - Glossary***

### ***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 12 – Fall Security Target

**ST Version** – Version 0.4

**ST Date** –2022/10/24

**Proprietary Documentation** (sections with additional information are marked with KMD in the section title)

**KMD Title** – Samsung Electronics Co., Ltd. Samsung Galaxy Devices on Android 12 - Key Management Description

**KMD Version** – Version 0.4

**KMD Date** – 2022/10/24

## 1.2 TOE Reference

**TOE Identification** – Samsung Galaxy Devices on Android 12 - Fall

**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 12.

## 1.4 TOE Description

The TOE is a mobile device based on Android 12 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 boot and device lock password (swipe, PIN, pattern, accessibility (direction), screen locks are not allowed). Acceptable biometrics vary with the device for the device lock.
- The maximum password failure retry policy should be less than or equal to 30.
- A screen lock password required to decrypt data on boot.
- Revocation checking must be enabled.
- Security and audit logging must be enabled.
- External storage must be encrypted.
- When CC mode has been enabled, the TOE behaves as follows:
    - The TOE sets the system wide Android CC mode property to be enabled.
    - The TOE prevents loading of custom firmware/kernels and requires all updates occur through FOTA.
    - The TOE utilizes ACVP/CAVP approved cryptographic ciphers for TLS.

The TOE includes the ability to create separate profiles part of the Knox Platform. A profile 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 profiles created. Thus, the evaluation includes several Knox-specific claims that apply when these profiles are created.

There are different models of the TOE, the Samsung Galaxy Devices on Android 12, and these models differ in their internal components (as described in the table below). All devices are A64 architecture.

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

| Device Name | Model Number | Chipset Vendor | SoC | Arch | Kernel | Build Number |
|---|---|---|---|---|---|---|
| Galaxy Z Flip4 | SM-F721 | Qualcomm | Snapdragon 8+ Gen 1 (SM8475) | ARMv8 | 5.10.81 | SP2A.220305.013 |
| Galaxy XCover6 Pro | SM-G736B | Qualcomm | Snapdragon 778G (SM7325) | ARMv8 | 5.4.147 | SP1A.210812.016 |
| Galaxy A53 5G | SM-A536 | Samsung | Exynos 1280 (S5E8825) | ARMv8 | 5.10.43 | SP1A.210812.016 |
| Galaxy A52 5G | SM-A526 | Qualcomm | Snapdragon 750G (SM7225) | ARMv8 | 4.19.152 | SP1A.210812.016 |
| Galaxy A71 5G | SM-A716 | Qualcomm | Snapdragon 765G (SM7250) | ARMv8 | 4.19.125 | SP1A.210812.016 |
| Galaxy Tab Active3 | SM-T575 | Samsung | Exynos 9810 | ARMv8 | 4.9.191 | SP1A.210812.016 |

*Table 2 - Evaluated Devices*

In addition to the evaluated devices, the devices in **Error! Reference source not found.** are claimed as e quivalent with a note about the differences between the evaluated device (first column) and the equivalent models (noted in the third column with the differences in the fourth column). Equivalence in this table is determined by the use of identical processors, kernel and build number, and is not made across processor types.

| Evaluated Device | SoC | Equivalent Devices | Differences |
|---|---|---|---|
| Galaxy Z Flip4 | Snapdragon 8+ Gen 1 (SM8475) | Galaxy Z Fold4 5G | Z Fold4 > Z Flip4 in terms of display size |
| Galaxy Xcover6 Pro | Snapdragon 778G (SM7325) | Galaxy Tab Active4 Pro | Tab Active4 Pro is tablet and have bigger screen size |
| Galaxy A53 5G | Exynos 1280 (S5E8825) | N/A | |
| Galaxy A52 5G | Snapdragon 750G (SM7225) | Galaxy A42 5G | A52 5G > A42 5G screen resolution & RAM |
| Galaxy A71 5G | Snapdragon 765G (SM7250) | Galaxy A51 5G | A71 5G > A51 5G in terms of display size |
| Galaxy Tab Active3 | Exynos 9810 | N/A | |

*Table 3 - 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). For each device, there are specific models that are validated. This table lists the specific carrier models that have the validated configuration (covering both evaluated and equivalent devices).

The Differences column in the table denotes the differences between the evaluated device and those listed in the Equivalent column. Except in the case of a different Wi-Fi radios or biometric sensors (in which case the radio or biometric is tested on a different device and so always verified as part of the evaluation), any differences between the evaluated device and claimed equivalent devices are outside the requirements of the evaluation requirements, such as screen size/type/resolution, battery size, position of ports.

| Device Name | Chipset Vendor | Chipset Name | Base Model Number | Carrier Models |
|---|---|---|---|---|
| Galaxy Z Flip4 | Qualcomm | Snapdragon 8+ Gen 1 (SM8475) | SM-F721 | W, B, C, N, U1, U, SC-54C*, SCG17*, |
| Galaxy Z Fold4 5G | Qualcomm | Snapdragon 8+ Gen 1 (SM8475) | SM-F936 | W, B, N, U1, U, SC-55C*, SCG16* |
| Galaxy XCover6 Pro | Qualcomm | Snapdragon 778G (SM7325) | SM-G736 | W, B, U1, U |
| Galaxy Tab Active4 Pro | Qualcomm | Snapdragon 778G (SM7325) | SM-T636 | B, N |
| | | | SM-T638 | U |
| | | | SM-T630 | None |
| Galaxy A53 5G | Samsung | Exynos 1280 (S5E8825) | SM-A536 | W, E, B, N, U1, V, U, SC-53C*, SCG15* |
| | | | SM-S536 | DL |
| Galaxy A52 5G | Qualcomm | Snapdragon 750G (SM7225) | SM-A526 | W, B, U1, U, SC-53B |
| Galaxy A42 5G | Qualcomm | Snapdragon 750G (SM7225) | SM-A426 | B, N, U1, U |
| | | | SM-S426 | DL |
| Galaxy A71 5G | Qualcomm | Snapdragon 765G (SM7250) | SM-A716 | U1, U, V |
| Galaxy A51 5G | Qualcomm | Snapdragon 765G (SM7250) | SM-A516 | V, SC54A*, SCG07*, |
| Galaxy Tab Active3 | Samsung | Exynos 9810 | SM-T577 | U |
| | | | SM-T575 | N, None |
| | | | SM-T570 | None |

*Table 4 - Carrier Models*

### 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 inherent in any mobile deployment.

Data on the TOE is protected through the implementation of Samsung File-Based Encryption (FBE) that utilizes ACVP/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 Knox Platform for Enterprise provides a set of flexible deployment options for work environments. With Knox Platform for Enterprise, it is possible to segment the device into two separate areas, by convention called the personal profile and the work profile. In creating a work profile, the Enterprise establishes a completely separate workspace, with its own authentication, applications and services, and ensure they are kept separate from anything the user may do in the personal profile. Another option for deployment is Knox Separated Apps, a folder where the Enterprise can isolate a group of applications from the rest of the device, restricting access to shared information, while maintaining seamless access to the isolated applications for the user.

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 allow and block listing.

#### *1.4.1.1 Physical Boundaries*

The TOE is a multi-user capable mobile device based on Android 12 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 12:

- 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 ACVP certified algorithms for a wide range of cryptographic functions including the following: 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 work profiles and Knox Separated Apps enhance 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 case characters, 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 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 12 on Galaxy Devices Administrator Guide, version 8.0.1, October 24, 2022

# 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 5, April 2017.
    - Part 2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1 Revision 5, April 2017.
    - Part 3 Extended
- PP-Configuration for Mobile Device Fundamentals, Virtual Private Network (VPN) Clients, and Bluetooth, 15 May 2022 (CFG_MDF-VPNC-BT_V1.0)
    - The PP-Configuration includes the following components:
        - Base-PP: Protection Profile for Mobile Device Fundamentals, Version 3.2, (PP_MD_V3.2)
        - PP-Module: PP-Module for Virtual Private Network (VPN) Clients, Version 2.3, (MOD_VPNC-MDF_V2.3)
        - PP-Module: PP-Module for Bluetooth, Version 1.0, (MOD_BT_V1.0)
- Package Claims:
    - 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 (PP_WLAN_CLI_EP_V1.0)
    - Functional Package: Functional Package for Transport Layer Security (TLS), Version 1.1, (PKG_TLS_V1.1)
- Technical Decisions as of October 7, 2022:

| TD No. | PP | Applied | Rationale |
|---|---|---|---|
| 0194 | PP_WLAN_CLI_EP_V1.0 | Yes | |
| 0244 | PP_WLAN_CLI_EP_V1.0 | Yes | |
| 0439 | PP_WLAN_CLI_EP_V1.0 | Yes | |
| 0442 | PKG_TLS_V1.1 | Yes | |
| 0469 | PKG_TLS_V1.1 | No | Not a server |
| 0470 | PP_WLAN_CLI_EP_V1.0 | Yes | |
| 0492 | PP_WLAN_CLI_EP_V1.0 | Yes | |
| 0499 | PKG_TLS_V1.1 | Yes | |
| 0513 | PKG_TLS_V1.1 | Yes | |
| 0517 | PP_WLAN_CLI_EP_V1.0 | Yes | |
| 0588 | PKG_TLS_V1.1 | No | Not a server |
| 0596 | PP_MD_V3.2/ | Yes | |
| 0600 | PP_MD_V3.2/<br>MOD_BT_V1.0 | Yes | |
| 0622 | MOD_VPNC_V2.3 | No | Not using App PP as Base |
| 0623 | PP_MD_V3.2 | Yes | |
| 0640 | MOD_BT_V1.0 | Yes | |

| TD No. | PP | Applied | Rationale |
|---|---|---|---|
| 0643 | PP_MD_V3.2 | Yes | |
| 0646 | PP_MD_V3.2 | Yes | |
| 0650 | PP_MD_V3.2 | Yes | |
| 0653 | PP_MD_V3.2 | Yes | |
| 0658 | PP_MD_V3.2 | Yes | |
| 0663 | PP_MD_V3.2 | Yes | |

***Table 5 - Technical Decisions***

## 2.1 Conformance Rationale

The ST conforms to the requirements in CFG_MDF-VPNC-BT_V1.0, PP_WLAN_CLI_EP_V1.0 and PKG_TLS_V1.1. For simplicity, this shall be referenced as MDF/BT/WLAN/TLS/VPNC. As explained previously, the security problem definition, security objectives, and security requirements are defined in the.

# 3 Security Objectives.

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

In general, the MDF/BT/WLAN/TLS/VPNC has defined Security Objectives appropriate for Mobile Devices and as such are applicable to the Samsung Galaxy Devices on Android 12 TOE.

## 3.1 Security Objectives for the Operational Environment

- **OE.CONFIG** TOE administrators will configure the Mobile Device security functions correctly to create the intended security policy.
- **OE.NOTIFY** The Mobile User will immediately notify the administrator if the Mobile Device is lost or stolen.
- **OE.PRECAUTION** The Mobile User exercises precautions to reduce the risk of loss or theft of the Mobile Device.
- **OE.DATA_PROPER_USER** Administrators take measures to ensure that mobile device users are adequately vetted against malicious intent and are made aware of the expectations for appropriate use of the device.
- **OE.NO_TOE_BYPASS** (PP_WLAN_CLI_EP_V1.0) Information cannot flow between external and internal networks located in different enclaves without passing through the TOE.
- **OE.TRUSTED_ADMIN** (PP_WLAN_CLI_EP_V1.0) TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.
- **OE.NO_TOE_BYPASS** (MOD_VPN_CLI_V2.3) Information cannot flow onto the network to which the VPN client's host is connected without passing through the TOE.
- **OE.PHYSICAL** (MOD_VPN_CLI_V2.3) 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** (MOD_VPN_CLI_V2.3) 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 MDF/BT/WLAN/TLS/VPNC. The MDF/BT/WLAN/TLS/VPNC defines the following extended SFRs and SARs and since they are not redefined in this ST the MDF/BT/WLAN/TLS/VPNC should be consulted for more information concerning those CC extensions.

| Requirement Class | Requirement Component |
|---|---|
| **FCS: Cryptographic support** | FCS_CKM_EXT.1: Cryptographic Key Support |
| | FCS_CKM_EXT.2: Cryptographic Key Random Generation |
| | FCS_CKM_EXT.3: Cryptographic Key Generation |
| | FCS_CKM_EXT.4: Key Destruction |
| | FCS_CKM_EXT.5: TSF Wipe |
| | FCS_CKM_EXT.6: Salt Generation |
| | FCS_CKM_EXT.8: Bluetooth Key Generation |
| | FCS_HTTPS_EXT.1: HTTPS Protocol |
| | FCS_IPSEC_EXT.1: IPsec |
| | FCS_IV_EXT.1: Initialization Vector Generation |
| | FCS_RBG_EXT.1: Random Bit Generation |
| | FCS_RBG_EXT.2: Random Bit Generator State Preservation |
| | FCS_SRV_EXT.1: Cryptographic Algorithm Services |
| | FCS_SRV_EXT.2: Cryptographic Algorithm Services |
| | FCS_STG_EXT.1: Cryptographic Key Storage |
| | FCS_STG_EXT.2: Encrypted Cryptographic Key Storage |
| | FCS_STG_EXT.3: Integrity of Encrypted Key Storage |
| | PKG_TLS_V1.1: FCS_TLS_EXT.1: TLS Protocol |
| | PKG_TLS_V1.1: FCS_TLSC_EXT.1: TLS Client Protocol |
| | FCS_TLSC_EXT.2: TLS Client Support for Mutual Authentication |
| | FCS_TLSC_EXT.2/WLAN: TLS Client Support for Supported Groups Extension (EAP-TLS for WLAN) |
| | FCS_TLSC_EXT.4: TLS Client Support for Renegotiation |
| | FCS_TLSC_EXT.5: TLS Client Support for Supported Groups Extension |
| **FDP: User data protection** | FDP_ACF_EXT.1: Access Control for System Services |
| | FDP_ACF_EXT.2: Access Control for System Resources |
| | FDP_ACF_EXT.3: Security Attribute Based Access Control |
| | FDP_DAR_EXT.1: Protected Data Encryption |
| | FDP_DAR_EXT.2: Sensitive Data Encryption |
| | FDP_IFC_EXT.1: Subset Information Flow Control |
| | FDP_IFC_EXT.1: Subset Information Flow Control |
| | FDP_PBA_EXT.1: Storage of Critical Biometric Parameters |
| | FDP_STG_EXT.1: User Data Storage |
| | FDP_UPC_EXT.1: Inter-TSF User Data Transfer Protection |
| **FIA: Identification and authentication** | FIA_AFL_EXT.1: Authentication Failure Handling |
| | FIA_BLT_EXT.1: Bluetooth User Authorization |
| | FIA_BLT_EXT.2: Bluetooth Mutual Authentication |
| | FIA_BLT_EXT.3: Rejection of Duplicate Bluetooth Connections |

| Requirement Class | Requirement Component |
|---|---|
| | FIA_BLT_EXT.4: Secure Simple Pairing |
| | FIA_BLT_EXT.6: Trusted Bluetooth Device User Authorization |
| | FIA_BLT_EXT.7: Untrusted Bluetooth Device User Authorization |
| | FIA_BMG_EXT.1: Accuracy of Biometric Authentication |
| | FIA_PAE_EXT.1: Port Access Entity Authentication |
| | FIA_PMG_EXT.1: Password Management |
| | FIA_PSK_EXT.1: Pre-Shared Key Composition |
| | FIA_TRT_EXT.1: Authentication Throttling |
| | FIA_UAU_EXT.1: Authentication for Cryptographic Operation |
| | FIA_UAU_EXT.2: Timing of Authentication |
| | FIA_UAU_EXT.4: Secondary User Authentication |
| | FIA_X509_EXT.1: Validation of Certificates |
| | FIA_X509_EXT.1/WLAN: X.509 Certificate Validation |
| | FIA_X509_EXT.2: X.509 Certificate Authentication |
| | FIA_X509_EXT.3: Request Validation of Certificates |
| **FMT: Security management** | FMT_MOF_EXT.1: Management of Security Functions Behavior |
| | FMT_SMF_EXT.1: Specification of Management Functions |
| | FMT_SMF_EXT.2: Specification of Remediation Actions |
| | FMT_SMF_EXT.3 Current Administrator |
| **FPT: Protection of the TSF** | FPT_AEX_EXT.1: Application Address Space Layout Randomization |
| | FPT_AEX_EXT.2: Memory Page Permissions |
| | FPT_AEX_EXT.3: Stack Overflow Protection |
| | FPT_AEX_EXT.4: Domain Isolation |
| | FPT_AEX_EXT.5: Kernel Address Space Layout Randomization |
| | FPT_AEX_EXT.6: Write or Execute Memory Page Permissions |
| | FPT_BBD_EXT.1: Application Processor Mediation |
| | FPT_JTA_EXT.1: JTAG Disablement |
| | FPT_KST_EXT.1: Key Storage |
| | FPT_KST_EXT.2: No Key Transmission |
| | FPT_KST_EXT.3: No Plaintext Key Export |
| | FPT_NOT_EXT.1: Self-Test Notification |
| | FPT_TST_EXT.1: TSF Cryptographic Functionality Testing |
| | FPT_TST_EXT.2: TSF Integrity Checking |
| | FPT_TUD_EXT.1: Trusted Update: TSF Version Query |
| | FPT_TUD_EXT.2: TSF Update Verification |
| **FTA: TOE access** | FTA_SSL_EXT.1: TSF- and User-initiated Locked State |
| | FTA_WSE_EXT.1: Wireless Network Access |
| | FTP_ITC_EXT.1: Trusted Channel Communication |
| **ALC: Life Cycle Support** | ALC_TSU_EXT.1: Timely Security Updates |

***Table 6 - Extended SFRs and SARs***

# 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 MDF/BT/WLAN/TLS/VPNC. The refinements and operations already performed in the MDF/BT/WLAN/TLS/VPNC are not identified (e.g., highlighted) here, rather the requirements have been copied from the PP_MD_V3.2/MOD_BT_V1.0/PP_WLAN_CLI_EP_V1.0/ PKG_TLS_V1.1/MOD_VPN_CLI_V2.3 and any residual operations have been completed herein. Of particular note, the MDF/BT/WLAN/TLS/VPNC 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 MDF/BT/WLAN/TLS/VPNC, 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 MDF/BT/WLAN/TLS/VPNC that serve to ensure corresponding evaluations will yield more practical and consistent assurance than the assurance requirements alone. The MDF/BT/WLAN/TLS/VPNC 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 12 TOE.

| Requirement Class | Requirement Component |
|---|---|
| **FAU: Security Audit** | FAU_GEN.1: Audit Data Generation |
| | MOD_BT_V1.0: FAU_GEN.1/BT Audit Data Generation (Bluetooth) |
| | PP_WLAN_CLI_EP_V1.0: FAU_GEN.1/WLAN: Audit Data Generation (Wireless LAN) |
| | FAU_SAR.1: Audit Review |
| | FAU_STG.1: Audit Storage Protection |
| | FAU_STG.4: Prevention of Audit Data Loss |
| **FCS: Cryptographic support** | FCS_CKM.1: Cryptographic Key Generation |
| | PP_WLAN_CLI_EP_V1.0: FCS_CKM.1/WLAN: Cryptographic Key Generation (Symmetric Keys for WPA2 Connections) |
| | MOD_VPN_CLI_V2.3: FCS_CKM.1/VPN: Cryptographic Key Generation (IKE) |
| | FCS_CKM.2/UNLOCKED: Cryptographic Key Establishment |
| | FCS_CKM.2/LOCKED: Cryptographic Key Establishment |
| | PP_WLAN_CLI_EP_V1.0: FCS_CKM.2/WLAN: Cryptographic Key Distribution (Group Temporal Key for WLAN) |
| | FCS_CKM_EXT.1: Cryptographic Key Support |
| | FCS_CKM_EXT.2: Cryptographic Key Random Generation |
| | FCS_CKM_EXT.3: Cryptographic Key Generation |
| | FCS_CKM_EXT.4: Key Destruction |
| | FCS_CKM_EXT.5: TSF Wipe |
| | FCS_CKM_EXT.6: Salt Generation |

| Requirement Class | Requirement Component |
|---|---|
| | MOD_BT_V1.0: FCS_CKM_EXT.8: Bluetooth Key Generation |
| | FCS_COP.1/ENCRYPT: Cryptographic operation |
| | FCS_COP.1/HASH: Cryptographic operation |
| | FCS_COP.1/SIGN: Cryptographic operation |
| | FCS_COP.1/KEYHMAC: Cryptographic operation |
| | FCS_COP.1/CONDITION: Cryptographic operation |
| | FCS_HTTPS_EXT.1: HTTPS Protocol |
| | MOD_VPN_CLI_V2.3: FCS_IPSEC_EXT.1: IPsec |
| | FCS_IV_EXT.1: Initialization Vector Generation |
| | FCS_RBG_EXT.1: Random Bit Generation |
| | FCS_RBG_EXT.2: Random Bit Generator State Preservation |
| | FCS_SRV_EXT.1: Cryptographic Algorithm Services |
| | FCS_SRV_EXT.2: Cryptographic Algorithm Services |
| | FCS_STG_EXT.1: Cryptographic Key Storage |
| | FCS_STG_EXT.2: Encrypted Cryptographic Key Storage |
| | FCS_STG_EXT.3: Integrity of Encrypted Key Storage |
| | PKG_TLS_V1.1: FCS_TLS_EXT.1: TLS Protocol |
| | PKG_TLS_V1.1: FCS_TLSC_EXT.1: TLS Client Protocol |
| | PP_WLAN_CLI_EP_V1.0: FCS_TLSC_EXT.1/WLAN: TLS Client Protocol (EAP-TLS for WLAN) |
| | PKG_TLS_V1.1: FCS_TLSC_EXT.2: TLS Client Support for Mutual Authentication |
| | PP_WLAN_CLI_EP_V1.0: FCS_TLSC_EXT.2/WLAN: TLS Client Support for Supported Groups Extension (EAP-TLS for WLAN) |
| | PKG_TLS_V1.1: FCS_TLSC_EXT.4: TLS Client Support for Renegotiation |
| | PKG_TLS_V1.1: FCS_TLSC_EXT.5: TLS Client Support for Supported Groups Extension |
| **FDP: User data protection** | FDP_ACF_EXT.1: Access Control for System Services |
| | FDP_ACF_EXT.2: Access Control for System Resources |
| | FDP_ACF_EXT.3: Security Attribute Based Access Control |
| | FDP_DAR_EXT.1: Protected Data Encryption |
| | FDP_DAR_EXT.2: Sensitive Data Encryption |
| | FDP_IFC_EXT.1: Subset Information Flow Control |
| | MOD_VPN_CLI_V2.3: FDP_IFC_EXT.1: Subset Information Flow Control |
| | FDP_PBA_EXT.1: Storage of Critical Biometric Parameters |
| | MOD_VPN_CLI_V2.3: FDP_RIP.2: Full Residual Information Protection |
| | FDP_STG_EXT.1: User Data Storage |
| | FDP_UPC_EXT.1/APPS: Inter-TSF User Data Transfer Protection (Applications) |
| | FDP_UPC_EXT.1/BT: Inter-TSF User Data Transfer Protection (Bluetooth) |
| **FIA: Identification and authentication** | FIA_AFL_EXT.1: Authentication Failure Handling |
| | MOD_BT_V1.0: FIA_BLT_EXT.1: Bluetooth User Authorization |
| | MOD_BT_V1.0: FIA_BLT_EXT.2: Bluetooth Mutual Authentication |

| Requirement Class | Requirement Component |
|---|---|
| | MOD_BT_V1.0: FIA_BLT_EXT.3: Rejection of Duplicate Bluetooth Connections |
| | MOD_BT_V1.0: FIA_BLT_EXT.4: Secure Simple Pairing |
| | MOD_BT_V1.0: FIA_BLT_EXT.6: Trusted Bluetooth Device User Authorization |
| | MOD_BT_V1.0: FIA_BLT_EXT.7: Untrusted Bluetooth Device User Authorization |
| | FIA_BMG_EXT.1(1): Accuracy of Biometric Authentication |
| | FIA_BMG_EXT.1(2): Accuracy of Biometric Authentication |
| | PP_WLAN_CLI_EP_V1.0: FIA_PAE_EXT.1: Port Access Entity Authentication |
| | FIA_PMG_EXT.1: Password Management |
| | MOD_VPN_CLI_V2.3: FIA_PSK_EXT.1: Pre-Shared Key Composition |
| | FIA_TRT_EXT.1: Authentication Throttling |
| | FIA_UAU.5: Multiple Authentication Mechanisms |
| | FIA_UAU.6: Re-Authentication |
| | FIA_UAU.7: Protected Authentication Feedback |
| | FIA_UAU_EXT.1: Authentication for Cryptographic Operation |
| | FIA_UAU_EXT.2: Timing of Authentication |
| | FIA_UAU_EXT.4: Secondary User Authentication |
| | FIA_X509_EXT.1: Validation of Certificates |
| | PP_WLAN_CLI_EP_V1.0: FIA_X509_EXT.1/WLAN: X.509 Certificate Validation |
| | FIA_X509_EXT.2: X.509 Certificate Authentication |
| | PP_WLAN_CLI_EP_V1.0: FIA_X509_EXT.2/WLAN: X.509 Certificate Authentication (EAP-TLS for WLAN) |
| | FIA_X509_EXT.3: Request Validation of Certificates |
| **FMT: Security management** | FMT_MOF_EXT.1: Management of Security Functions Behavior |
| | FMT_SMF_EXT.1: Specification of Management Functions |
| | MOD_BT_CLI_V1.0: FMT_SMF_EXT.1/BT: Specification of Management Functions (Bluetooth) |
| | PP_WLAN_CLI_EP_V1.0: FMT_SMF_EXT.1/WLAN: Specification of Management Functions (WLAN Client) |
| | MOD_VPN_CLI_V2.3: FMT_SMF.1/VPN: Specification of Management Functions - VPN |
| | FMT_SMF_EXT.2: Specification of Remediation Actions |
| | FMT_SMF_EXT.3: Current Administrator |
| **FPT: Protection of the TSF** | FPT_AEX_EXT.1: Application Address Space Layout Randomization |
| | FPT_AEX_EXT.2: Memory Page Permissions |
| | FPT_AEX_EXT.3: Stack Overflow Protection |
| | FPT_AEX_EXT.4: Domain Isolation |
| | FPT_AEX_EXT.5: Kernel Address Space Layout Randomization |
| | FPT_AEX_EXT.6: Write or Execute Memory Page Permissions |
| | FPT_BBD_EXT.1: Application Processor Mediation |
| | FPT_JTA_EXT.1: JTAG Disablement |
| | FPT_KST_EXT.1: Key Storage |

| Requirement Class | Requirement Component |
|---|---|
| | FPT_KST_EXT.2: No Key Transmission |
| | FPT_KST_EXT.3: No Plaintext Key Export |
| | FPT_NOT_EXT.1: Self-Test Notification |
| | FPT_STM.1: Reliable time stamps |
| | FPT_TST_EXT.1: TSF Cryptographic Functionality Testing |
| | PP_WLAN_CLI_EP_V1.0: FPT_TST_EXT.1/WLAN: TSF Cryptographic Functionality Testing (WLAN Client) |
| | MOD_VPN_CLI_V2.3: FPT_TST_EXT.1/VPN: TSF Self-Test (VPN Client) |
| | FPT_TST_EXT.2/PREKERNEL: TSF Integrity Checking (Pre-Kernel) |
| | FPT_TST_EXT.2/POSTKERNEL: TSF Integrity Checking (Post-Kernel) |
| | FPT_TUD_EXT.1: Trusted Update: TSF Version Query |
| | FPT_TUD_EXT.2: TSF Update Verification |
| | FPT_TUD_EXT.3: Application Signing |
| | FPT_TUD_EXT.6: Trusted Update Verification |
| **FTA: TOE access** | FTA_SSL_EXT.1: TSF- and User-initiated Locked State |
| | FTA_TAB.1: Default TOE Access Banners |
| | PP_WLAN_CLI_EP_V1.0: FTA_WSE_EXT.1: Wireless Network Access |
| | MOD_BT_V1.0: FTP_BLT_EXT.1: Bluetooth Encryption |
| | MOD_BT_V1.0: FTP_BLT_EXT.2: Persistence of Bluetooth Encryption |
| | MOD_BT_V1.0: FTP_BLT_EXT.3/BR: Bluetooth Encryption Parameters (BR/EDR) |
| | MOD_BT_V1.0: FTP_BLT_EXT.3/LE: Bluetooth Encryption Parameters (LE) |
| | FTP_ITC_EXT.1: Trusted Channel Communication |

***Table 7 - TOE Security Functional Requirements***

## 5.1.1 Security Audit (FAU)

### *5.1.1.1 FAU_GEN.1: Audit Data Generation, MOD_BT_V1.0: FAU_GEN.1/BT Audit Data Generation (Bluetooth) & PP_WLAN_CLI_EP_V1.0: FAU_GEN.1/WLAN: Audit Data Generation (Wireless LAN)*

**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 OS
5. Insertion or removal of removable media
6. Specifically defined auditable events in Table 12 (from Table 1 of the PP_MD_V3.2);
7. [***Audit records reaching [95] percentage of audit capacity***]
8. [***Specifically defined auditable events as listed in Table 12 - Audit Events from Table 2 of the PP_MD_V3.2***]
9. [***Auditable events as listed in Table 13 – Bluetooth Audit Events from Table 2 of the MOD_BT_V1.0***]

10. [***Auditable events as listed in Table 14 – WLAN Client Audit Events from Table 2 of the PP_WLAN_CLI_EP_V1.0***]
11. ***[Auditable events as listed in** Error! Reference source not found. **from Table 3 of the PKG_TLS_V1.1]***

**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 (of the PP_MD_V3.2)
6. [***Additional information in Table 2 (of the PP_MD_V3.2)***]
7. [*Additional information in the Auditable Events table in Table 2 of the MOD_BT_V1.0*]

### *5.1.1.2 FAU_SAR.1: Audit Review*

**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 FAU_STG.1: Audit Storage Protection*

**FAU_STG.1.1**

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

**FAU_STG.1.2**

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

### *5.1.1.4 FAU_STG.4: Prevention of Audit Data Loss*

**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 FCS_CKM.1: Cryptographic Key Generation*

**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 [
  - ***Diffie-Hellman group 14 that meet the following: RFC 3526***],

and [

- ***RSA schemes using cryptographic key sizes of 2048-bit or greater that meet FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3***

].

### 5.1.2.2 PP_WLAN_CLI_EP_V1.0: FCS_CKM.1/WLAN: Cryptographic Key Generation (Symmetric Keys for WPA2 Connections)

**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 [***PRF-704***] and specified cryptographic key sizes [128 bits] and [***256 bits***] 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 MOD_VPN_CLI_V2.3: FCS_CKM.1/VPN: Cryptographic Key Generation (IKE)

**FCS_CKM.1.1/VPN**

The TSF shall [***invoke platform-provided functionality***] 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 FCS_CKM.2/UNLOCKED: Cryptographic Key Establishment

**FCS_CKM.2.1/UNLOCKED**

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 Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography",**

[

- ***Finite field-based key establishment schemes that meets the following: NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"***
- RSA-based key establishment schemes that meet the following [
  - ***NIST Special Publication 800-56B, "Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography"***

] and

- ***Key establishment schemes using Diffie-Hellman group 14 that meets the following: RFC 3526***

].

### *5.1.2.5 FCS_CKM.2/LOCKED: Cryptographic Key Establishment*

**FCS_CKM.2.1/LOCKED**

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 PP_WLAN_CLI_EP_V1.0: FCS_CKM.2/WLAN: Cryptographic Key Distribution (GTK)*

**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 FCS_CKM_EXT.1: Cryptographic Key Support*

**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 the 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 FCS_CKM_EXT.2: Cryptographic Key Random Generation*

**FCS_CKM_EXT.2.1**

All DEKs shall be [

- ***randomly generated (for SD card encryption)***
- ***from the combination of a randomly generated DEK with another DEK or salt in a way that preserves the effective entropy of each factor by [concatenating the keys and using a KDF (as described in SP 800-108)] (for FBE)***

] with entropy corresponding to the security strength of AES key sizes of [***256***] bits.

### *5.1.2.9 FCS_CKM_EXT.3: Cryptographic Key Generation*

**FCS_CKM_EXT.3.1**

The TSF shall use [

- ***asymmetric KEKs of [128-bit] 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 according to FCS_COP.1.1/CONDITION

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 [concatenating the keys and using a KDF (as described in SP 800-108) (for FBE), encrypting one key with another]***].

### 5.1.2.10 FCS_CKM_EXT.4: Key Destruction

**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 FCS_CKM_EXT.5: TSF Wipe

**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***
- ***Overwriting all Protected Data according to the following rules:***

- ***For 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 flash memory, that is not wear-leveled, the destruction shall be executed [by a block erase that erases the reference to memory that stores data as well as the data itself].***
- ***For 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.5.2**

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

### 5.1.2.12 FCS_CKM_EXT.6: Salt Generation

**FCS_CKM_EXT.6.1**

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

### 5.1.2.13 MOD_BT_V1.0: FCS_CKM_EXT.8: Bluetooth Key Generation

**FCS_CKM_EXT.8.1**

The TSF shall generate public/private ECDH key pairs every [**connection attempt**].

### 5.1.2.14 FCS_COP.1/ENCRYPT: Cryptographic operation

**FCS_COP.1.1/ENCRYPT**

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***].

### 5.1.2.15 FCS_COP.1/HASH: Cryptographic operation

**FCS_COP.1.1/HASH**

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.

### 5.1.2.16 FCS_COP.1/SIGN: Cryptographic operation

**FCS_COP.1.1/SIGN**

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***].

### *5.1.2.17 FCS_COP.1/KEYHMAC: Cryptographic operation*

**FCS_COP.1.1/KEYHMAC**

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".

### *5.1.2.18 FCS_COP.1/CONDITION: Cryptographic operation*

**FCS_COP.1.1/CONDITION**

The TSF shall perform conditioning in accordance with a specified cryptographic algorithm HMAC-[***SHA-256***] using a salt, and [***PDKDF2 with [8192] iterations, [key stretching with scrypt)]***], and output cryptographic key sizes [***256***] that meet the following: [***NIST SP 800-132 (PBKDF2), no standard (scrypt)***].

### *5.1.2.19 FCS_HTTPS_EXT.1: HTTPS Protocol*

**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 as defined in the Package for Transport Layer Security.

**FCS_HTTPS_EXT.1.3**

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

### *5.1.2.20 MOD_VPN_CLI_V2.3: FCS_IPSEC_EXT.1: IPsec*

**FCS_IPSEC_EXT.1.1**

The TSF shall implement the IPsec architecture as specified in RFC 4301.

**FCS_IPSEC_EXT.1.2**

The TSF shall implement [***tunnel mode***].

**FCS_IPSEC_EXT.1.3**

The TSF 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 TSF 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***].

**FCS_IPSEC_EXT.1.5**

The TSF 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], [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), RFC 8784, RFC 8247 and [no other RFCs for hash functions]***].

**FCS_IPSEC_EXT.1.6**

The TSF shall ensure the encrypted payload in the [***IKEv1***[1]***, 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***][2].

**FCS_IPSEC_EXT.1.7**

The TSF 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 TSF shall ensure that all IKE protocols implement DH groups 19 (256-bit Random ECP), 20 (384-bit Random ECP), and [***24 (2048-bit MODP with 256-bit POS), 14 (2048-bit MODP)***].

**FCS_IPSEC_EXT.1.9**

The TSF 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 TSF 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)**].

**FCS_IPSEC_EXT.1.11**

The TSF 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**

[1] The A53, Z Flip4, and XCover 6 Pro devices do not support IKEv1.

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

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.

#### *5.1.2.21 FCS_IV_EXT.1: Initialization Vector Generation*

**FCS_IV_EXT.1.1**

The TSF shall generate IVs in accordance with PP_MD_V3.2 Table 12: References and IV Requirements for NIST-approved Cipher Modes.

#### *5.1.2.22 FCS_RBG_EXT.1: Random Bit Generation*

**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 [***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.

#### *5.1.2.23 FCS_RBG_EXT.2: Random Bit Generator State Preservation*

**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.

#### *5.1.2.24 FCS_SRV_EXT.1: Cryptographic Algorithm Services*

**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/LOCKED
- The following algorithms in FCS_COP.1/ENCRYPT: AES-CBC, [***AES-GCM***]
- All mandatory and selected algorithms in FCS_COP.1/SIGN
- All mandatory and selected algorithms in FCS_COP.1/HASH
- All mandatory and selected algorithms in FCS_COP.1/KEYHMAC
- [***No other cryptographic operations***].

### 5.1.2.25 FCS_SRV_EXT.2: Cryptographic Algorithm Services

**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/ENCRYPT
- Algorithms in FCS_COP.1/SIGN

by keys stored in the secure key storage.

### 5.1.2.26 FCS_STG_EXT.1: Cryptographic Key Storage

**FCS_STG_EXT.1.1**

The TSF shall provide [***mutable hardware (only available on Z Flip4 and ZFold 4 devices), 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, the administrator***].

**FCS_STG_EXT.1.4**

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

**FCS_STG_EXT.1.5**

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

### 5.1.2.27 FCS_STG_EXT.2: Encrypted Cryptographic Key Storage

**FCS_STG_EXT.2.1**

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

- ***Protected by the REK with [***
  - ***encryption by a REK,***
  - ***encryption by a KEK that is derived from a REK],***
- ***Protected by the REK and the password with [***
  - ***encryption by a REK and the password-derived KEK,***
  - ***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 SecurityLogAgent properties (related to SE Android)***] 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]***

].

### 5.1.2.28 FCS_STG_EXT.3: Integrity of Encrypted Key Storage

**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/HASH) of the stored key that is encrypted by a key protected by FCS_STG_EXT.2***
- ***a keyed hash (FCS_COP.1/KEYHMAC) 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.

### 5.1.2.29 PKG_TLS_V1.1: FCS_TLS_EXT.1: TLS Protocol

**FCS_TLS_EXT.1.1**

The product shall implement [

- ***TLS as a client***

].

### 5.1.2.30 PKG_TLS_V1.1: FCS_TLSC_EXT.1: TLS Client Protocol

**FCS_TLSC_EXT.1.1**

The product shall implement TLS 1.2 (RFC 5246) and [***no earlier TLS versions***] as a client that supports the cipher suites [

- ***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***]

and also supports functionality for [

- ***mutual authentication***
- ***session renegotiation***

].

**FCS_TLSC_EXT.1.2**

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

**FCS_TLSC_EXT.1.3**

The product shall not establish a trusted channel if the server certificate is invalid [

- ***with no exceptions***

].

(TD0442 applied)

#### 5.1.2.31 PP_WLAN_CLI_EP_V1.0: FCS_TLSC_EXT.1/WLAN: Extensible Authentication Protocol-Transport Layer Security

**FCS_TLSC_EXT.1.1/WLAN**

The TSF shall implement [***TLS 1.0 (RFC 2246), 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: [

- ***TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246,***
- ***TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,***
- ***TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288,***
- ***TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,***
- ***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_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/WLAN.

**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.

(TD0492, TD0517 applied)

#### 5.1.2.32 PKG_TLS_V1.1: FCS_TLSC_EXT.2: TLS Client Support for Mutual Authentication

**FCS_TLSC_EXT.2.1**

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

#### 5.1.2.33 PP_WLAN_CLI_EP_V1.0: FCS_TLSC_EXT.2/WLAN: TLS Client Protocol

**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)

#### *5.1.2.34 PKG_TLS_V1.1: FCS_TLSC_EXT.4: TLS Client Support for Renegotiation*

**FCS_TLSC_EXT.4.1**

The product shall support secure renegotiation through use of the 'renegotiation_info' TLS extension in accordance with RFC 5746.

#### *5.1.2.35 PKG_TLS_V1.1: FCS_TLSC_EXT.5: TLS Client Support for Supported Groups Extension*

**FCS_TLSC_EXT.5.1**

The product shall present the Supported Groups Extension in the Client Hello with the FCS_TLSS_EXT.1.1 supported groups: [

- ***secp256r1,***
- ***secp384r1***

].

### 5.1.3 User Data Protection (FDP)

#### *5.1.3.1 FDP_ACF_EXT.1: Access Control for System Services*

**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 FDP_ACF_EXT.2: Access Control for System Resources*

**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 FDP_ACF_EXT.3: Security Attribute Based Access Control*

**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 FDP_DAR_EXT.1: Protected Data Encryption*

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

### 5.1.3.5 FDP_DAR_EXT.2: Sensitive Data Encryption

**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 FDP_IFC_EXT.1: Subset Information Flow Control

**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 as defined in the PP-Module for VPN Client***

] with the exception of IP traffic required to establish the VPN connection and [***captive portal traffic needed for correct functioning of the TOE***], when the VPN is enabled. (TD0596 applied).

### 5.1.3.7 MOD_VPN_CLI_V2.3: FDP_IFC_EXT.1: Subset Information Flow Control

**FDP_IFC_EXT.1.1**

The TSF shall ensure that all IP traffic (other than IP traffic required to establish the VPN connection) flow through the IPsec VPN client.

### 5.1.3.8 FDP_PBA_EXT.1: Storage of Critical Biometric Parameters

**FDP_PBA_EXT.1.1**

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

### 5.1.3.9 MOD_VPN_CLI_V2.3: FDP_RIP.2: Full Residual Information Protection

**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.

### 5.1.3.10 FDP_STG_EXT.1: User Data Storage

**FDP_STG_EXT.1.1**

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

#### *5.1.3.11 FDP_UPC_EXT.1/APPS: Inter-TSF User Data Transfer Protection (Applications)*

**FDP_UPC_EXT.1.1/APPS**

The TSF shall provide a means for non-TSF applications executing on the TOE to use

- mutually authenticated TLS as defined in the Package for Transport Layer Security,
- HTTPS,

and [

- ***IPsec as defined in the PP-Module for VPN Client***

] 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/APPS**

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

#### *5.1.3.12 FDP_UPC_EXT.1/BT: Inter-TSF User Data Transfer Protection (Bluetooth)*

**FDP_UPC_EXT.1.1/BT**

The TSF shall provide a means for non-TSF applications executing on the TOE to use

- Bluetooth BR/EDR in accordance with the PP-Module for Bluetooth,

and [

- ***Bluetooth LE in accordance with the PP-Module for Bluetooth***

] 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/BT**

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 FIA_AFL_EXT.1: Authentication Failure Handling*

**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 [**1-30**] 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 MOD_BT_V1.0: FIA_BLT_EXT.1: Bluetooth User Authorization*

**FIA_BLT_EXT.1.1**

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

### *5.1.4.3 MOD_BT_V1.0: FIA_BLT_EXT.2: Bluetooth Mutual Authentication*

**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 MOD_BT_V1.0: FIA_BLT_EXT.3: Rejection of Duplicate Bluetooth Connections*

**FIA_BLT_EXT.3.1**

The TSF shall discard pairing and session initialization attempts from a Bluetooth device address (BD_ADDR) to which an active session already exists.

### *5.1.4.5 MOD_BT_V1.0: FIA_BLT_EXT.4: Secure Simple Pairing*

**FIA_BLT_EXT.4.1**

The TOE shall support Bluetooth Secure Simple Pairing, both in the host and the controller.

**FIA_BLT_EXT.4.2**

The TOE shall support Secure Simple Pairing during the pairing process.

### *5.1.4.6 MOD_BT_V1.0: FIA_BLT_EXT.6: Trusted Bluetooth Device User Authorization*

**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)***].

### *5.1.4.7 MOD_BT_V1.0: FIA_BLT_EXT.7: Untrusted Bluetooth Device User Authorization*

**FIA_BLT_EXT.7.1**

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

### 5.1.4.8 FIA_BMG_EXT.1(1): Accuracy of Biometric Authentication

**FIA_BMG_EXT.1(1).1**

The one-attempt BAF False Accept Rate (FAR) for [**fingerprint**] shall not exceed [**1:10,000**] with a one-attempt BAF False Reject Rate (FRR) not to exceed 1 in [**33**].

**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.

### 5.1.4.9 FIA_BMG_EXT.1(2): Accuracy of Biometric Authentication

**FIA_BMG_EXT.1(2).1**

The one-attempt BAF False Accept Rate (FAR) for [**hybrid**] shall not exceed [**1:10,000**] with a one-attempt BAF False Reject Rate (FRR) not to exceed 1 in [**33**].

**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.

### 5.1.4.10 PP_WLAN_CLI_EP_V1.0: FIA_PAE_EXT.1: Port Access Entity Authentication

**FIA_PAE_EXT.1.1**

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

### 5.1.4.11 FIA_PMG_EXT.1: Password Management

**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.

### 5.1.4.12 MOD_VPN_CLI_V2.3: FIA_PSK_EXT.1: Pre-Shared Key Composition

**FIA_PSK_EXT.1.1**

The TSF shall be able to use pre-shared keys for IPsec.

**FIA_PSK_EXT.1.2**

The TSF 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 TSF shall condition the text-based pre-shared keys by using [***[the entered string as ASCII Hex]***], [***be able to [accept] bit-based pre-shared keys***].

### 5.1.4.13 FIA_TRT_EXT.1: Authentication Throttling

**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.

### 5.1.4.14 FIA_UAU.5: Multiple Authentication Mechanisms

**FIA_UAU.5.1**

The TSF shall provide password and [***fingerprint, 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 lock 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 device lock screen (not at the first lock screen after reboot/power-up)**
- **Hybrid**
  - **For work environments unlock and hybrid authentication factor configured by the user**

].

### 5.1.4.15 FIA_UAU.6: Re-Authentication

**FIA_UAU.6.1**

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

**FIA_UAU.6.2**

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 FIA_UAU.7: Protected Authentication Feedback

**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 FIA_UAU_EXT.1: Authentication for Cryptographic Operation

**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 FIA_UAU_EXT.2: Timing of Authentication*

**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)***
- ***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***
- ***Auto rotate***
- ***Sound (on, mute, vibrate)***
- ***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 FIA_UAU_EXT.4: Secondary User Authentication*

**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 FIA_X509_EXT.1: Validation of Certificates*

**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, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.

- The TSF shall validate that any CA certificate includes caSigning purpose in the key usage field
- The TSF shall validate the revocation status of the certificate using [***OCSP as specified in RFC 6960 (for TLS, HTTPS, EAP-TLS, IPsec), CRL as specified in RFC 5759 (for TLS, HTTPS)***].
- 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
  - 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. [conditional]
  - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the EKU field.
  - OCSP certificates presented for OCSP responses shall have the OCSP Signing purpose (id-dp 9 with OID 1.3.6.1.5.5.7.3.9) in the EKU field. [conditional]

**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 PP_WLAN_CLI_EP_V1.0: FIA_X509_EXT.1/WLAN: X.509 Certificate Validation*

**FIA_X509_EXT.1.1/WLAN**

The TSF shall validate certificates for EAP-TLS 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 extendedKeyUsage field according to the following rules:
  - 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
  - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the extendedKeyUsage field.

**FIA_X509_EXT.1.2/WLAN**

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.

(TD0439 applied)

### *5.1.4.22 FIA_X509_EXT.2: X.509 Certificate Authentication*

**FIA_X509_EXT.2.1**

The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for mutually authenticated TLS as defined in the Package for Transport Layer Security, HTTPS, [***IPsec in accordance with the PP-Module for VPN Client***], 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***].

#### 5.1.4.23 PP_WLAN_CLI_EP_V1.0: FIA_X509_EXT.2/WLAN: X.509 Certificate Authentication (EAP-TLS)

**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.

(TD0517 applied)

#### 5.1.4.24 FIA_X509_EXT.3: Request Validation of Certificates

**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 FMT_MOF_EXT.1: Management of Security Functions Behavior

**FMT_MOF_EXT.1.1**

The TSF shall restrict the ability to perform the functions in column 4 of Table 8 - Security Management Functions to the user.

**FMT_MOF_EXT.1.2**

The TSF shall restrict the ability to perform the functions in column 6 of Table 8 - Security Management Functions to the administrator when the device is enrolled and according to the administrator-configured policy.

(TD0658 applied)

#### 5.1.5.2 FMT_SMF_EXT.1: Specification of Management Functions

**FMT_SMF_EXT.1.1**

The TSF shall be capable of performing the functions in column 2 of Table 8 - Security Management Functions.

| **Management Function** <br> **Mandatory = M** <br> **Implemented = I** | **Implemented** | **User Only** | **Admin** | **Admin Only** |
|---|---|---|---|---|
| 1. configure password policy:<br>a. minimum password length<br>b. minimum password complexity<br>c. maximum password lifetime<br>The administrator can configure the required password characteristics (minimum length, complexity, and lifetime) using the MDM APIs.<br>There are distinct settings for the passwords used to unlock the personal and work profiles.<br>Length: an integer value of characters (0 = no minimum)<br>Complexity: Unspecified, Something, Numeric, Alphabetic, Alphanumeric, Complex.<br>Lifetime: an integer value of days (0 = no maximum) | M | | M | M |
| 2. configure session locking policy:<br>a. screen-lock enabled/disabled<br>b. screen lock timeout<br>c. number of authentication failures<br>The administrator can configure the session locking policy using the MDM APIs. There are distinct settings for personal and work profile inactivity.<br>The user can also adjust each of the session locking policies for the personal and work profile; however, if set by the administrator, the user can only set a more restrictive policy (e.g., setting the device to allow fewer authentication failures than configured by the administrator).<br>Screen lock timeout: an integer number of minutes before the TOE locks (0 = no lock timeout)<br>Authentication failures: an integer number (0 = no limit) | M | | M | M |
| 3. enable/disable the VPN protection:<br>a. across device<br>[<br>***b. on a per-app basis,***<br>***c. on a per-group of applications processes basis***<br>]<br>The user can configure and then enable the TOE's VPN to protect traffic across the entire device.<br>The administrator (through an MDM Agent that utilizes the MDM APIs) can restrict the TOE's ability to connect to a VPN.<br>The administrator can configure per-app and per-work profile VPN connections with the work profile MDM APIs. | M | | I | I |
| 4. enable/disable [**NFC, Bluetooth, Wi-Fi, and cellular radios**]<br>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. The TOE's radios operate at frequencies of 2.4 GHz (NFC/Bluetooth), 2.4/5 GHz (Wi-Fi), and 850 MHz (4G/LTE). | M | | I | I |

| Management Function<br>Mandatory = M<br>Implemented = I | Implemented | User Only | Admin | Admin Only |
|---|---|---|---|---|
| 5. enable/disable [**camera, microphone**]:<br>a. across device<br>[<br>***d. no other method***<br>]<br>An administrator may configure the TOE (through an MDM agent utilizing the 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.<br>The administrator can also disable the use of the camera or microphone inside a work profile without affecting access to those devices when outside the work profile. | M |  | I | I |
| 6. transition to the locked state<br>Both users and administrators (using the MDM APIs) can transition the TOE into a locked state. | M |  | M | - |
| 7. TSF wipe of protected data<br>Both users and administrators (using the MDM APIs) can force the TOE to perform a full wipe (factory reset) of data. | M |  | M |  |
| 8. configure application installation policy by:<br>[<br>***a. restricting the sources of applications***<br>***b. specifying a set of allowed applications based on [application name, developer signature] (an application whitelist)***<br>***c. denying installation of applications***<br>]<br>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 Play Store. There are distinct settings for disabling the installation of applications for the personal and work profile. | M |  | M | M |
| 9. import keys/secrets into the secure key storage<br>Both users and administrators (using the 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<br>Both users and administrators (using the MDM APIs) can destroy secret keys in the secure key storage. | M |  | I |  |
| 11. import X.509v3 certificates into the Trust Anchor Database<br>Both users and administrators (using the MDM APIs) can import X.509v3 certificates into the Trust Anchor Database. | M |  | M |  |

| **Management Function** <br> **Mandatory = M** <br> **Implemented = I** | **Implemented** | **User Only** | **Admin** | **Admin Only** |
|---|---|---|---|---|
| 12. remove imported X.509v3 certificates and [***default X.509v3 certificates***] in the Trust Anchor Database <br> Both users and administrators (using the MDM APIs) can remove imported X.509v3 certificates from the Trust Anchor Database as well as disable any of the TOE's default Root CA certificates (in the latter case, the CA certificate still resides in the TOE's read-only system partition; however, the TOE will treat that Root CA certificate and any certificate chaining to it as untrusted). | M | | I | |
| 13. enroll the TOE in management <br> 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 | I | | |
| 14. remove applications <br> Both users and administrators (using the MDM APIs) can uninstall user and administrator installed applications in the personal profile and applications inside a work profile. | M | | M | |
| 15. update system software <br> 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 work profile software). | M | | M | |
| 16. install applications <br> Both users and administrators (using the MDM APIs) can install applications in the personal profile and applications inside a work profile. Only administrators (using the MDM APIs) can specify to install applications in a Knox Separated Apps folder. | M | | M | |
| 17. remove Enterprise applications <br> Both users and administrators (using the MDM APIs) can uninstall user and administrator installed applications in the personal profile and applications inside a work profile. Applications installed within the work profile are marked as Enterprise (work) applications | M | | M | |
| 18. enable/disable display notification in the locked state of: <br> [ <br> ***f. all notifications*** <br> ] <br> TOE users can configure the TOE to allow or disallow notifications while in a locked state. | M | | | |
| 19. enable data-at rest protection <br> The TOE always encrypts user data storage. | M | | | |

| **Management Function**<br>**Mandatory = M**<br>**Implemented = I** | **Implemented** | **User Only** | **Admin** | **Admin Only** |
|---|---|---|---|---|
| 20. enable removable media's data-at-rest protection<br>The administrator (using the MDM APIs) can configure a removable media encryption policy (on supported devices[3]). Once enabled, the device will prompt a user to encrypt a newly inserted external SD cards (and the device will then encrypt any files present after which the device can use the SD Card). If the user chooses not to encrypt the newly inserted external SD card, then the device cannot access the SD Card. If the administrator has set the policy to force encrypt removable media, then the settings option to decrypt the SD Card is greyed out and the user cannot decrypt the SD Card. | M | | I | I |
| 21. enable/disable location services:<br>a. across device<br>[<br>***d. no other method***<br>]<br>The administrator (using the MDM APIs) can disable location services.<br>Unless disabled by the administrator, TOE users can enable and disable location services. | M | | I | I |
| 22. enable/disable the use of [***Fingerprint Authentication Factor, Hybrid Authentication Factor***]<br>The TOE supports disabling Biometric authentication for the both the TOE's normal device lock screen and for the TOE's work profile lock screen. The TOE's normal device lock screen supports biometrics, which the administrator can disable. The TOE's work profile supports hybrid authentication (combination of password and biometrics), which the administrator can also disable. | M | | I | I |
| 23. **configure whether to allow/disallow establishment of [assignment: configurable trusted channel in FTP_ITC_EXT.1.1/FDP_UPC_EXT.1.1/APPS] if the peer/server certificate is deemed invalid.** | | | | |
| 24. enable/disable all data signaling over [**assignment**: *list of externally accessible hardware ports*] | | | | |
| 25. enable/disable [**assignment**: list of protocols where the device acts as a server] | | | | |
| 26. enable/disable developer modes<br>The administrator (using the MDM APIs) can disable Developer Mode.<br>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<br>The TOE work profile provides the ability to remove only Enterprise data versus user data. | I | | | |
| 29. approve [**selection:** *import, removal*] by applications of X.509v3 certificates in the Trust Anchor Database | | | | |

[3] Samsung Galaxy Z Flip4, Z Fold4 devices do not support removable media.

| **Management Function**<br>**Mandatory = M**<br>**Implemented = I** | **Implemented** | **User Only** | **Admin** | **Admin Only** |
|---|---|---|---|---|
| 30. configure whether to allow/disallow establishment of a trusted channel 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<br>The administrator (using the 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<br>The administrator (using the MDM APIs) can define 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 | | | | |
| 38. retrieve TSF-software integrity verification values | | | | |
| 39. enable/disable<br>[<br>***a. USB mass storage mode***<br>]<br>The administrator (using the 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:** *locally connected system, remote system*] | | | | |
| 41. enable/disable<br>[<br>***a. Hotspot functionality authenticated by [pre-shared key],***<br>***b. USB tethering authenticated by [passcode]***<br>]<br>The administrator (using the MDM APIs) can disable the wireless hotspot and USB tethering.<br>If enabled by the administrator (and supported by the device), TOE users can configure:<br>• A Wi-Fi hotspot with a pre-shared key<br>• A Bluetooth hotspot with another device that has been successfully completed a Bluetooth pairing (establishing a pre-shared key)<br>• A USB tethering connection when the user has first authenticated to the device (the tethering connection will be maintained when the device locks, but initial setup requires the user to actively approve the connection via authentication to the device) | I | | I | I |

| **Management Function**<br>**Mandatory = M**<br>**Implemented = I** | **Implemented** | **User Only** | **Admin** | **Admin Only** |
|---|---|---|---|---|
| 42. approve exceptions for sharing data between [***groups of applications***]<br>The TOE work profile and Knox Separated Apps folder provide separation between groups of application processes along with the ability to control the ability to share data between these groups | I | | | I |
| 43. place applications into application process groups based on [**creating a Knox Separated Apps folder**] | I | | I | I |
| 44. unenroll the TOE from management | I | | I | I |
| 45. Enable/disable the Always On VPN protection<br>a. across device<br>[<br>***b. on a per-app basis,***<br>***c. on a per-group of applications processes basis***]<br>The user can be required to use an Always On VPN | I | | I | I |
| 46. revoke Biometric template | | | | |
| 47. additional management functions<br>[<br>• **enable/disable USB host storage**<br>• **disable CC Mode**<br>• **enable/disable manual Date/Time changes**<br>• **enable/disable applications (including pre-installed)**<br>]<br>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<br>I<br>I<br>I | | I<br>I<br>I<br>I | I<br>I<br>I<br>I |

***Table 8 - Security Management Functions***

(TD0646 applied)

### *5.1.5.3 MOD_BT_CLI_V1.0: FMT_SMF_EXT.1/BT: Specification of Management Functions (Bluetooth)*

**FMT_SMF_EXT.1.1/BT**

The TSF shall be capable of performing the following Bluetooth management functions:

| **Management Function**<br>**Mandatory = M**<br>**Implemented = I** | **Implemented** | **User Only** | **Admin** | **Admin Only** |
|---|---|---|---|---|
| BT1. Configure the Bluetooth trusted channel.<br>• Disable/enable the Discoverable (for BR/EDR) and Advertising (for LE) modes; | M | | I | I |
| BT2. Change the Bluetooth device name (separately for BR/EDR and LE); | | | | |
| BT3. Provide separate controls for turning the BR/EDR and LE radios on and off; | | | | |

| **Management Function** **Mandatory = M** **Implemented = I** | **Implemented** | **User Only** | **Admin** | **Admin Only** |
|---|---|---|---|---|
| BT4. Allow/disallow the following additional wireless technologies to be used with Bluetooth: [***Wi-Fi, NFC***];<br>Wi-Fi Direct services to transfer files can be enabled/disabled by the admin. NFC can be disabled and so unable to be used for Bluetooth pairing. | I |  | I | I |
| BT5. Configure allowable methods of Out of Band pairing (for BR/EDR and LE); |  |  |  |  |
| BT6. Disable/enable the Discoverable (for BR/EDR) and Advertising (for LE) modes separately; |  |  |  |  |
| BT7. Disable/enable the Connectable mode (for BR/EDR and LE); |  |  |  |  |
| BT8. Disable/enable the Bluetooth [**assignment:** list of Bluetooth service and/or profiles available on the OS (for BR/EDR and LE)]; |  |  |  |  |
| BT9. Specify minimum level of security for each pairing (for BR/EDR and LE); |  |  |  |  |

***Table 9 - Bluetooth Security Management Functions***

### *5.1.5.4 PP_WLAN_CLI_EP_V1.0: FMT_SMF_EXT.1/WLAN: Specification of Management Functions (WLAN Client)*

**FMT_SMF_EXT.1.1/WLAN**

The TSF shall be capable of performing the following management functions:

| **Management Function** **Mandatory = M** **Implemented = I** | **Implemented** | **User Only** | **Admin** | **Admin Only** |
|---|---|---|---|---|
| WL1. configure security policy for each wireless network:<br>a. [***specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s)***]<br>b. security type<br>c. authentication protocol<br>d. client credentials to be used for authentication; | M |  | M | I |
| WL2. specify wireless networks (SSIDs) to which the TSF may connect; | M |  | M | I |
| WL3. enable/disable certificate revocation list checking; |  |  |  |  |
| WL4. disable ad hoc wireless client-to-client connection capability; |  |  |  |  |
| WL5. 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); |  |  |  |  |
| WL6. disable roaming capability |  |  |  |  |
| WL7. enable/disable IEEE 802.1X pre-authentication |  |  |  |  |
| WL8. enable/disable and configure PMK caching<br>a. set the amount of time (in minutes) for which PMK entries are cached<br>b. set the maximum number of PMK entries that can be cached |  |  |  |  |

***Table 10 - WLAN Security Management Functions***

### 5.1.5.5 MOD_VPN_CLI_V2.3: FMT_SMF.1/VPN: Specification of Management Functions - VPN

**FMT_SMF.1.1/VPN**

The TSF shall be capable of performing the following management functions**:**

| Management Function | Mandatory = M<br>Implemented = I | Implemented | User Only | Admin | Admin Only |
|---|---|---|---|---|---|
| VPN1. | Specify VPN gateways to use for connections | I | | | |
| VPN2. | Specify IPsec VPN Clients to use for connections | | | | |
| VPN3. | Specify IPsec-capable network devices to use for connections | | | | |
| VPN4. | Specify client credentials to be used for connections | I | | | |
| VPN5. | Configure the reference identifier of the peer | I | | | |
| VPN6. | [any additional VPN management functions] | | | | |

***Table 11 - VPN Security Management Functions***

### 5.1.5.6 FMT_SMF_EXT.2: Specification of Remediation Actions

**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***

] upon unenrollment and [***no other triggers***].

### 5.1.5.7 FMT_SMF_EXT.3: Current Administrator

**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 FPT_AEX_EXT.1: Application Address Space Layout Randomization

**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 FPT_AEX_EXT.2: Memory Page Permissions

**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 FPT_AEX_EXT.3: Stack Overflow Protection

**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 FPT_AEX_EXT.4: Domain Isolation

**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 FPT_AEX_EXT.5: Kernel Address Space Layout Randomization

**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 [**6**] unpredictable bits.

### 5.1.6.6 FPT_AEX_EXT.6: Write or Execute Memory Page Permissions

**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 FPT_BBD_EXT.1: Application Processor Mediation

**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 FPT_JTA_EXT.1: JTAG Disablement

**FPT_JTA_EXT.1.1**

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

### 5.1.6.9 FPT_KST_EXT.1: Key Storage

**FPT_KST_EXT.1.1**

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

### 5.1.6.10 FPT_KST_EXT.2: No Key Transmission

**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 FPT_KST_EXT.3: No Plaintext Key Export*

**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 FPT_NOT_EXT.1: Self-Test Notification*

**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 FPT_STM.1: Reliable time stamps*

**FPT_STM.1.1**

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

#### *5.1.6.14 FPT_TST_EXT.1: TSF Cryptographic Functionality Testing*

**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 PP_WLAN_CLI_EP_V1.0: FPT_TST_EXT.1/WLAN: TSF Cryptographic Functionality Testing*

**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 MOD_VPN_CLI_V2.3: FPT_TST_EXT.1/VPN: TSF Self-Test (VPN Client)*

**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 FPT_TST_EXT.2/PREKERNEL: TSF Integrity Checking (Pre-Kernel)*

**FPT_TST_EXT.2.1/PREKERNEL**

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 FPT_TST_EXT.2/POSTKERNEL: TSF Integrity Checking (Post-Kernel)*

**FPT_TST_EXT.2.1/POSTKERNEL**

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 FPT_TUD_EXT.1: Trusted Update: TSF Version Query*

**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 FPT_TUD_EXT.2: TSF Update Verification*

**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***].

### *5.1.6.21 FPT_TUD_EXT.3: Application Signing*

**FPT_TUD_EXT.3.1**

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

### *5.1.6.22 FPT_TUD_EXT.6: Trusted Update Verification*

**FPT_TUD_EXT.6.1**

The TSF shall verify that software updates to the TSF are a current or later version than the current version of the TSF.

### 5.1.7 TOE Access (FTA)

#### *5.1.7.1 FTA_SSL_EXT.1: TSF- and User-initiated Locked State*

**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 FTA_TAB.1: Default TOE Access Banners*

**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 PP_WLAN_CLI_EP_V1.0: FTA_WSE_EXT.1: Wireless Network Access*

**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.

(TD0470 applied)

### 5.1.8 Trusted Path/Channels (FTP)

#### *5.1.8.1 MOD_BT_V1.0: FTP_BLT_EXT.1: Bluetooth Encryption*

**FTP_BLT_EXT.1.1**

The TSF shall enforce the use of encryption when transmitting data over the Bluetooth trusted channel for BR/EDR and [***LE***].

**FTP_BLT_EXT.1.2**

The TSF shall use key pairs per FCS_CKM_EXT.8 for Bluetooth encryption.

#### *5.1.8.2 MOD_BT_V1.0: FTP_BLT_EXT.2: Persistence of Bluetooth Encryption*

**FTP_BLT_EXT.2.1**

The TSF shall [restart encryption] if the remote device stops encryption while connected to the TOE.

#### *5.1.8.3 MOD_BT_V1.0: FTP_BLT_EXT.3/BR: Bluetooth Encryption Parameters (BR/EDR)*

**FTP_BLT_EXT.3.1/BR**

The TSF shall set the minimum encryption key size to [**128 bits**] for [*BR/EDR*] and not negotiate encryption key sizes smaller than the minimum size.

#### *5.1.8.4 MOD_BT_V1.0: FTP_BLT_EXT.3/LE: Bluetooth Encryption Parameters (LE)*

**FTP_BLT_EXT.3.1/LE**

The TSF shall set the minimum encryption key size to [**128 bits**] for [*LE*] and not negotiate encryption key sizes smaller than the minimum size.

#### *5.1.8.5 FTP_ITC_EXT.1: Trusted Channel Communication*

**FTP_ITC_EXT.1.1**

The TSF shall use

- 802.11-2012 in accordance with the PP-Module for WLAN Clients,
- 802.1X in accordance with the PP-Module for WLAN Clients,
- EAP-TLS in accordance with the PP-Module for WLAN Clients,
- mutually authenticated TLS as defined in the Package for Transport Layer Security

and [

- ***IPsec in accordance with the PP-Module for VPN Client,***
- ***HTTPS***

] protocols 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***].

#### *5.1.8.6 PP_WLAN_CLI_EP_V1.0: FTP_ITC_EXT.1/WLAN: Trusted Channel Communication*

**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 and provides assured identification of its end points and protection of the channel data from modification or disclosure.

**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 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.

## 5.2.1 Development (ADV)

### 5.2.1.1 ADV_FSP.1: Basic Functional Specification

**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 AGD_OPE.1: Operational User Guidance

**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 AGD_PRE.1: Preparative Procedures*

**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 ALC_CMC.1: Labelling of the TOE*

**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 ALC_CMS.1: TOE CM Coverage*

**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 ALC_TSU_EXT.1: Timely Security Updates*

**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 ATE_IND.1: Independent Testing - sample*

**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 AVA_VAN.1: Vulnerability Survey*

**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**
TOE provides several distinct mechanisms for Security Audit logging – one introduced via KNOX APIs, and two introduced from mainline AOSP – Security Log (https://developer.android.com/reference/android/app/admin/SecurityLog) and generic Log (https://developer.android.com/reference/android/util/Log). Some events might only be audited by one of these systems, so the client is encouraged to monitor both.

The following table enumerates the events that the TOE audits including the mechanisms logging it. Requirements marked with "(O)" are from the Table 2: Additional Auditable Events of the PP_MD_V3.2.

| Requirement | Audit Event | Content | Mechanism |
|---|---|---|---|
| **FAU_GEN.1** | Start-up and shutdown of the audit functions | | Generic Log |
| **FAU_GEN.1** | All administrative actions | | KNOX, Security Log |
| **FAU_GEN.1** | Start-up and shutdown of the OS and kernel | | Security Log |
| **FAU_GEN.1** | Insertion or removal of removable media | | Security Log, generic Log |
| **FCS_CKM.1** | Failure of key generation activity for authentication keys. | | Security Log |
| **FCS_STG_EXT.1** | Import or destruction of key. | Identity of key. Role and identity of requestor. | Security Log |
| **FCS_STG_EXT.3** | Failure to verify integrity of stored key. | Identity of key being verified. | Security Log |
| **FDP_DAR_EXT.2** | Failure to encrypt/decrypt data. | | KNOX |
| **FDP_STG_EXT.1** | Addition or removal of certificate from Trust Anchor Database. | Subject name of certificate. | Security Log |
| **FIA_UAU.6 (O)** | User changes Password Authentication Factor. | | KNOX |

| Requirement | Audit Event | Content | Mechanism |
|---|---|---|---|
| **FIA_X509_EXT.1** | Failure to validate X.509v3 certificate. | Reason for failure of validation. | KNOX |
| **FIA_X509_EXT.2 (O)** | Failure to establish connection to determine revocation status. | | KNOX |
| **FMT_SMF_EXT.1** | Change of settings. | Role of user that changed setting. Value of new setting. | KNOX |
| **FMT_SMF_EXT.1 (O)** | Success or failure of function. | Role of user that performed function. Function performed. Reason for failure. | KNOX |
| **FMT_SMF_EXT.1 (O)** | Initiation of software update. | Version of update. | KNOX |
| **FMT_SMF_EXT.1 (O)** | Initiation of application installation or update. | Name and version of application. | KNOX |
| **FPT_TST_EXT.1** | Initiation of self-test. Failure of self-test. | [*none*] | KNOX |
| **FPT_TST_EXT.2(1)** | Start-up of TOE. [*none*] | No additional Information. [***no additional information***] | Security Log |
| **FPT_TUD_EXT.2 (O)** | Success or failure of signature verification for software updates. | | KNOX |
| **FPT_TUD_EXT.2 (O)** | Success or failure of signature verification for applications. | | KNOX |
| **FTA_TAB.1 (O)** | Change in banner setting. | | KNOX |

***Table 12 - Audit Events***

### MOD_BT_V1.0: FAU_GEN.1/BT

The following table enumerates the events that the TOE audits from Table 2: Auditable Events of the MOD_BT_V1.0. Requirements marked with "(O)" are optional in the table.

| Requirement | Audit Event | Content | Mechanism |
|---|---|---|---|
| **FIA_BLT_EXT.1** | Failed user authorization of Bluetooth device. | User authorization decision (e.g., user rejected connection, incorrect pin entry). | KNOX |
| **FIA_BLT_EXT.2** | Initiation of Bluetooth connection. | Bluetooth address and name of device. | Generic log |
| | Failure of Bluetooth connection. | Reason for failure. | KNOX |
| **FIA_BLT_EXT.3 (O)** | Duplicate connection attempt. | BD_ADDR of connection attempt. | Generic log |

***Table 13 – Bluetooth Audit Events***

### PP_WLAN_CLI_EP_V1.0: FAU_GEN.1/WLAN

The following table enumerates the events that the TOE audits from Table 2: Auditable Events of the PP_WLAN_CLI_EP_V1.0.

| Requirement | Audit Event | Content | Mechanism |
|---|---|---|---|
| **FCS_TLSC_EXT.1/ WLAN** | Failure to establish an EAP-TLS session. | Reason for failure. | KNOX |
| | Establishment/termination of an EAP-TLS session. | Non-TOE endpoint of connection. | KNOX |
| **FPT_TST_EXT.1/ WLAN** | Execution of this set of TSF self-tests. | | KNOX |
| | [*none*]. | [*no additional information*]. | KNOX |
| **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). | KNOX and generic Log |
| **FTP_ITC.1/WLAN** | All attempts to establish a trusted channel.<br>Detection of modification of channel data. | Identification of the non-TOE endpoint of the channel. | KNOX and generic Log |

***Table 14 – WLAN Client Audit Events***

**PKG_TLS_V1.1: FAU_GEN.1**
No events from PKG_TLS_V1.1 are selected.

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

**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 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 BoringSSL and ECDSA (including P-256, P384 and P-521) keys in in SCrypto. 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 (FFC) | 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 15 - Asymmetric Key Generation per Module*

### PP_WLAN_CLI_EP_V1.0: 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 and PRF704 for WPA2 derivation of 128-bit or 256-bit AES Temporal Key (using the HMAC implementation provided by BoringSSL) and 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 A52 | SM-G78xx | 110242, 110243, 110475, 110613, 110614 |
| Galaxy A42 | SM-A515x | 102133, 102135, 110678, 111052 |
| | SM-S515x | 111054 |
| Galaxy A71 5G | SM-G715x | 97494, 100309, 98205 |
| Galaxy A51 5G | SM-N976x | 100038, 97686 |
| Galaxy Tab Active3 | SM-N975x | 101265, 101555, 101556, 101557, 101558 |
| Galaxy XCover6 Pro | SM-G736x | 119715, 119711, 119667, 119714, 119712 |
| Galaxy Z Flip 4 | SM-F721x | 119746,119748,119749,119750,119752,119755,119847 |
| Galaxy ZFold 4 | SM-F936x | 119631,119976,119977,119978,119979,119980,119981,120027,120147 |

*Table 16 - W-Fi Alliance Certificates*

### MOD_VPN_CLI_V2.3: 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/UNLOCKED

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/LOCKED**
The TOE uses ECDH with a P-256 curve key establishment for protection of application sensitive data received while the device is locked.

**PP_WLAN_CLI_EP_V1.0: 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 its 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** ***(KMD)***
The TOE supports Data Encryption Key (DEK) generation using its approved RBGs for use in SD card encryption. The TOE RBGs are capable of generating 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. For FBE, the TOE supports using a SP800-108 KDF to concatenate keys together to generate unique DEKs. The keys used in the SP800-108 KDF are generated by the approved RBGs. The TOE can also generate 128-bit asymmetric keys used for sensitive data protection.

**FCS_CKM_EXT.3** ***(KMD)***
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 FBE, the SD Card encryption, and work profile encryption (normal and sensitive).

The TOE generates a number of different KEKs. In addition to the TSF KEKs, applications may request key generation (through either the Android APIs or SCrypto APIs within the TEE), and the TOE utilizes its

BoringSSL/SCrypto CTR_DRBG and Kernel Crypto HMAC_DRBG to satisfy those requests. The requesting application ultimately chooses whether to use that key as a DEK or a KEK, but it is worth mentioning here, as an application can utilize such a key as a KEK, should it choose.

**FCS_CKM_EXT.4 *(KMD)***
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 FBE 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 DEKs used to encrypt the data partition using a block erase and read verify command to ensure that the UFS blocks containing the encrypted DEKs (FBE and SD card) are now reported as empty. After the encrypted keys have been erased, the TOE will delete the entire user partition with a block erase command and then 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 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 17 - Salt Creation***

**FCS_CKM_EXT.8**
The TOE will generate new ECDH key pairs for every pairing attempt.

**FCS_COP.1/ENCRYPT**
**FCS_COP.1/HASH**
**FCS_COP.1/SIGN**
**FCS_COP.1/KEYHMAC**
**FCS_COP.1/CONDITION**

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

The BoringSSL v1.6 library (with both Processor Algorithm Accelerators (PAA) and without PAA) 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/ENCRYPT | A2351 |
| CVL ECC - P-256/384/521 | SP 800-56A | FCS_CKM.2(1)<br>FCS_CKM_EXT.3 | A2351 |
| DRBG CTR – 256 | SP 800-90A | FCS_RBG_EXT.1 | A2351 |

| Algorithm | NIST Standard | SFR Reference | Cert# |
|---|---|---|---|
| ECDSA PKG/PKV/SigGen/SigVer - P-256/384/521 | FIPS 186-4 | FCS_CKM.1<br>FCS_CKM.2(1)<br>FCS_COP.1/SIGN | A2351 |
| HMAC SHA-1/256/384/512 | FIPS 198-1 & 180-4 | FCS_COP.1/KEYHMAC | A2351 |
| RSA KeyGen/SigGen/SigVer – 2048/3072/4096 | FIPS 186-4 | FCS_CKM.1<br>FCS_COP.1/SIGN | A2351 |
| SHS SHA-1/256/384/512 | FIPS 180-4 | FCS_COP.1/HASH | A2351 |

***Table 18 - BoringSSL Cryptographic Algorithms***

The Samsung Crypto Extension v1.0 library provides the following algorithms.

| Algorithm | NIST Standard | SFR Reference | Cert# |
|---|---|---|---|
| KBKDF | SP 800-108 | FCS_CKM_EXT.3 | A933 |

***Table 19 - Samsung Crypto Extension Cryptographic Algorithms***

The evaluated devices utilize the following kernels for the Samsung Kernel Cryptographic Module (Kernel Crypto).

| Device | Kernel Version | Kernel Crypto Version |
|---|---|---|
| A53 | 5.10 | 2.2 |
| XCover6 Pro | 5.4 | 2.2 |
| Z Flip 4/Z Fold4 | 5.10 | 2.2 |
| A52 5G/A42 5G | 4.19 | 2.2 |
| A71 5G/A51 5G | 4.19 | 2.1 |
| Tab Active3 | 4.9 | 1.9 |

***Table 20 - Kernel Versions***

The Samsung Kernel Cryptographic ("Kernel Crypto") Module provides the following algorithms.

| Algorithm | NIST Standard | SFR Reference | Cert# |
|---|---|---|---|
| AES 128/256 CBC | FIPS 197, SP 800-38A | FCS_COP.1/ENCRYPT | A1456, A1455 |
| | | | A970, A969 |
| | | | A503, A502 |
| | | | 5184, 5183 |
| HMAC SHA-1/256 | FIPS 198-1 & 180-4 | FCS_COP.1/KEYHMAC | A1456, A1455 |
| | | | A970, A969 |
| | | | A503, A502 |
| | | | 3440, 3439 |
| DRBG SHA-256 HMAC_DRBG | SP 800-90A | FCS_RBG_EXT.1 | A1456, A1455 |
| | | | A970, A969 |
| | | | A503, A502 |
| | | | 1958 |
| SHS SHA-1/256 | FIPS 180-4 | FCS_COP.1/HASH | A1456, A1455 |
| | | | A970, A969 |
| | | | A503, A502 |
| | | | 4188, 4187 |

***Table 21 - Samsung Kernel Cryptographic Algorithms***

The evaluated devices utilize the Samsung SCrypto Cryptographic Module for cryptographic operations within the TEE on each device. The following table lists the TEE operating systems for each device.

| Device | TEE OS Version | SCrypto Version |
|---|---|---|
| A 53 5G | TEEGRIS 4.2.1 | 2.6 |
| A52 5G/A42 5G/ A51 5G | QSEE 5.10 | 2.5 |
| A71 5G | QSEE 5.8 | 2.5 |
| *All* Z Fold4/<br>Z Flip 4G (Qualcomm) | QSEE 5.18 | 2.6 |
| Tab Active3 | TEEGRIS 4.0 | 2.5 |
| XCover6 Pro | QSEE 5.11 | 2.5 |

***Table 22 - TEE Environments***

The Samsung SCrypto TEE library provides the following algorithms. Note that the TOE only performs RSA signing/decryption (using the private key) in the TEE, and performs public key verification/encryption in the normal world using BoringSSL.

| Algorithm | NIST Standard | SFR Reference | Cert# |
|---|---|---|---|
| AES CBC/GCM 128/256 | FIPS 197, SP 800-38A/D | FCS_COP.1/ENCRYPT | A915, A889, C1360 |
| DRBG AES-256 CTR_DRBG | SP 800-90A | FCS_RBG_EXT.1 | A915, A889, C1360 |
| ECDSA PKG/PKV/SigGen/SigVer P-256/384/521 | FIPS 186-4 | FCS_CKM.1<br>FCS_COP.1/SIGN | A915, A889, C1360 |
| HMAC SHA-1/256/384/512 | FIPS 198-1 & 180-4 | FCS_COP.1/KEYHMAC | A915, A889, C1360 |
| RSA KeyGen and SigGen (no verification) 2048 bits | FIPS 186-4 | FCS_CKM.1FCS_CKM.2 (1) FCS_COP.1(3) | A915, A889, C1360 |
| SHS SHA-1/256/384/512 | FIPS 180-4 | FCS_COP.1/HASH | A915, A889, C1360 |
| KBKDF | SP 800-108 | FCS_CKM_EXT.3 | A915, A889, C1360 |

***Table 23 - SCrypto TEE Cryptographic Algorithms***

The Chipset hardware for storage encryption has various modules that provide cryptographic functions. The modules and versions are listed here. Only discrete modules are listed here.

| Device | Flash Crypto |
|---|---|
| A53 | Samsung FMP v4.0.1 |
| A52 5G/A42 5G | Qualcomm ICE v3.1.0 |
| A71 5G/A51 5G | Qualcomm ICE v3.1.0 |
| Tab Active3 | Samsung FMP v2.0 (HW FX6_V4.0) |
| *All* Z Fold4/*Z Flip 4 5G/*(Qualcomm) | Qualcomm ICE v3.2.1 |

***Table 24 - Hardware Components***

The Samsung Flash Memory Protector ("FMP") Driver Module provides the following software algorithms.

| Algorithm | NIST Standard | SFR Reference | Cert# |
|---|---|---|---|
| SHS SHA-256 (Exynos FMP) | FIPS 180-4 | FCS_COP.1/HASH | A2724, A505 |
| HMAC SHA-256 (Exynos FMP) | FIPS 198-1 & 180-4 | FCS_COP.1/KEYHMAC | A2724, A505 |

***Table 25 - FMP Driver Algorithms***

The storage encryption modules provide the following algorithms.

| Algorithm | NIST Standard | SFR Reference | Cert# |
|---|---|---|---|
| XTS-AES 256 (Exynos FMP) | FIPS 197, SP 800-38E | FCS_COP.1/ENCRYPT | A2723, A504 |
| XTS-AES 128/256 (Qualcomm) | FIPS 197, SP 800-38E | FCS_COP.1/ENCRYPT | A2217, A2216 |
| | | | A1658, A1659 |
| | | | A1010, A1009 |
| | | | C1553, C1552 |

***Table 26 - Storage Hardware Algorithms***

The devices contain unique Wi-Fi chipsets based on the model of the device. The chipsets are listed here.

| Device | Wi-Fi Chipset |
|---|---|
| Galaxy Z Fold4 5G | Qualcomm WCN6856 |
| Galaxy XCover6 Pro | Qualcomm WCN6750 |
| A53 5G | Samsung S5N5C20X00-6030 |
| A52 5G/A42 5G | Qualcomm WCN3988 |
| A71 5G/A51 5G | Qualcomm WCN3998 |
| Tab Active3 | Broadcom BCM4361 |

***Table 27 - Wi-Fi Hardware Components***

The Wi-Fi chipset hardware provides the following algorithms.

| Algorithm | NIST Standard | SFR Reference | Cert# |
|---|---|---|---|
| AES 128 CCM (Qualcomm Wi-Fi) | FIPS 197, SP 800-38C | FCS_COP.1(1) | 5663, 4748, 4143 |
| AES 128 CCM (BCM Wi-Fi) | FIPS 197, SP 800-38C | FCS_COP.1(1) | 4152 |
| AES 128 CCM (Samsung Wi-Fi) | FIPS 197, SP 800-38C | FCS_COP.1(1) | C1623 |

***Table 28 - Wi-Fi Chip Algorithms***

Several devices provide a secure processor for mutable hardware key storage. The devices and chipsets are listed here.

| Device | Mutable Storage Chipset |
|---|---|
| *All Flip4/Fold4* (Qualcomm) | Secure Processing Unit 5.5 |

***Table 29 - Mutable Key Storage Components***

The chipsets for the mutable hardware key storage provide the following algorithms.

| Algorithm | NIST Standard | SFR Reference | Cert# |
|---|---|---|---|
| AES CBC/GCM 128/256 | FIPS 197, SP 800-38A/D | FCS_COP.1/ENCRYPT | A2707, A2706 |

| Algorithm | NIST Standard | SFR Reference | Cert# |
|---|---|---|---|
| DRBG Hash_DRBG (Qualcomm) | SP 800-90A | FCS_RBG_EXT.1 | A2545 |
| ECDSA SigGen/SigVer (P-256/384) | FIPS 186-4 | FCS_COP.1/SIGN | A2707 |
| HMAC SHA-256/384/512 (Qualcomm) | FIPS 198-1 & 180-4 | FCS_COP.1/KEYHMAC | A2706 |
| RSA SigGen/SigVer - 2048 | FIPS 186-4 | FCS_CKM.1<br>FCS_CKM.2(1)<br>FCS_COP.1/SIGN | A2707 |
| SHS SHA-256 | FIPS 180-4 | FCS_COP.1/HASH | A2706 |

**Table 30 - Mutable Key Storage Cryptographic Algorithms**

The SoC hardware provides the following algorithms.

| Algorithm | NIST Standard | SFR Reference | Cert# |
|---|---|---|---|
| KBKDF (Exynos) | SP 800-108 | FCS_CKM_EXT.3 | A2536, 196 |
| AES 128/256 CBC/GCM (Exynos) | FIPS 197, SP 800-38A/D | FCS_COP.1(1) | A2536, 5367 |
| SHS SHA-256 (Exynos) | FIPS 180-4 | FCS_COP.1(2) | A2536, 4309 |
| HMAC SHA-256 (Exynos) | FIPS 198-1 & 180-4 | FCS_COP.1(4) | A2536, 3555 |
| AES 128/256 CBC (Qualcomm) | FIPS 197, SP 800-38A | FCS_COP.1(1) | A2045, A805, A242 |
| DRBG SHA-256 Hash_DRBG (Qualcomm) | SP 800-90A | FCS_RBG_EXT.1 | A50, A2065 |
| SHS SHA-256 (Qualcomm) | FIPS 180-4 | FCS_COP.1(2) | A50, A2065 |

***Table 31 - SoC Cryptographic Algorithms***

The TOE’s application processors include hardware entropy implementations that supply random data within the TEE and to the Linux kernel RNG (primary input pool).

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 Android 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 using a combination of functions to increase the memory required for derivation to thwart attacks. This combination of functions is embedded in the scrypt algorithm. Scrypt uses three steps to condition the password:

1. One PBKDF2 operation (NIST SP 800-132)
2. Several rounds of ROMix operations
3. One final PBKDF2 operation (NIST SP 800-132)

This value is used as input for decrypting other keys that are tied to successful user authentication.

To unlock the user's keystore, the value generated in the previous step us conditioned further using PBKDF2 (NIST SP 800-132). The key derivation function uses the value derived from the initial password conditioning and a randomly generated 128-bit salt in 8192 HMAC-SHA-256 iterations to generate a 256-bit KEK.

In both cases of conditioning, 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 30 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, Crypto Extensions library and Chipset hardware implementations. These implementations have been tested upon Android 12 running atop both Exynos and Qualcomm ARMv8 chipsets. The TOE's Exynos processor devices include the Samsung Flash Memory Protector, while the TOE's Qualcomm processor devices include hardware XTS-AES (Inline Crypto Engine, ICE) implementations for internal storage 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.

**MOD_VPN_CLI_V2.3: 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, IKEv1 main mode (aggressive mode is not supported) 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 SPD configuration and 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 supports main mode when this is required by the Gateway.

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 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. 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 or 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 (IKEv1 does not support using ECDSA certificates) 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 MOD_VPN_CLI_V2.3 is supported by the VPN Client. The VPN Client processes the pre-shared keys by using the entered string as ASCII Hex 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 *(KMD)***
The TOE generates IVs for data storage encryption and for key storage encryption. The TOE uses XTS-AES and AES-CBC mode for data encryption and AES-GCM for key storage.

**FCS_RBG_EXT.1 *(KMD)***
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 API) to random data drawn from its AES-256 CTR_DRBG
2. An AES-256 CTR_DRBG provided by SCrypto in the TEE
3. A SHA-256 HMAC_DRBG provided by Kernel Crypto in the Android kernel (/dev/random or get_random_bytes())
4. A hardware SHA-256 Hash_DRBG provided by the Qualcomm Application Processor hardware

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 fill primary input pool continuously with random data that has full entropy, and in turn, the TOE draws from this input pool to seed both its AES-256 CTR_DRBG and its SHA-256 HMAC_DRBG. The TOE seeds each of its software DRBGs using 384-bits of data from the primary input pool, thus ensuring 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. The SHA-256 Hash_DRBG in the Qualcomm AP is used to generate the REK on first boot.

**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**
**FCS_SRV_EXT.2*(KMD)***
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 Android API methods and through the kernel. The vendor also developed testing applications to enable execution of the NIST algorithm test suite in order to verify the correctness of the algorithm implementations.

**FCS_STG_EXT.1**
The TOE provides users, administrators 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 or administrator (via the MDM) 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 user of the TOE can elect to delete keys from the keystore, as well as to wipe the entire device securely. The administrator can only delete keys from the keystore that have been deployed to the TOE by the administrator.

The TOE affords applications control (control over use and destruction) of keys that they create or import, and only the 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 | | |
| Administrator | Yes | Yes | | |
| Mobile application | Yes | Yes | | |
| Common application developer | | | Yes | Yes |

***Table 32 - Key Management Matrix***

On the devices with support for mutable hardware storage (see Table 29 - Mutable Key Storage Components), the TOE provides both software-based (the Android Keystore) and mutable hardware-based (the StrongBox Keystore) key storage. The key storage is implemented in a separate hardware chipset that is part of the SoC, providing additional protection beyond that normally provided by the Android Keystore which is implemented in the TEE. The hardware module implements its own cryptographic services internally to the module.

By default, all key storage is handled with the Android Keystore, but an application developer can use the StrongBox keystore by setting a preference to use the hardware keystore instead. When the preference is set to the hardware keystore, the keys will be stored in the mutable hardware and all operations on the keys will be handled inside the hardware.

**FCS_STG_EXT.2 *(KMD)***
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.

The TOE also stores the SecurityLogAgent (properties related to the SE Android configuration) in the same manner as the long-term trusted channel key materials.

**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.

**PKG_TLS_V1.1: TCS_TLS_EXT.1**
**PKG_TLS_V1.1: FCS_TLSC_EXT.1**
**PKG_TLS_V1.1: FCS_TLSC_EXT.2**
**PKG_TLS_V1.1: FCS_TLSC_EXT.4**
**PKG_TLS_V1.1: FCS_TLSC_EXT.5**

The TOE provides mobile applications (through its Android API) the use of TLS version 1.2 only as a client, including support for the selected ciphersuites in the selections in section 5.1.2.30. The TOE supports Common Name (CN) and Subject Alternative Name (SAN) (DNS and IP address) as reference identifiers. The TOE supports client (mutual) authentication and session renegotiation. The TOE inherently (without requiring any configuration) supports the supported ciphersuites and evaluated elliptic curves (P-256 and P-384); neither the user nor the administrator need configure anything in order for the TOE to support these ciphersuites or curves. The TOE supports the use of wildcards in X.509 reference identifiers (CN and SAN), and the TOE supports certificate pinning through Android's Network security configuration. This configuration allows a mobile application to specify one or more certificate public key hashes (SHA-1 or SHA-256) along with the domain and optionally an expiry. With such a configuration, the application will only establish a TLS connection if one of the public keys in the certificate path matches a "pinned" key hash. After the optional expiry, Android disregards the pinned certificates and performs no pinning (to prevent connectivity issues in apps that have not been updated).

**PP_WLAN_CLI_EP_V1.0: FCS_TLSC_EXT.1/WLAN**
**PP_WLAN_CLI_EP_V1.0: FCS_TLSC_EXT.2/WLAN**
The TSF supports TLS versions 1.2, 1.1 and 1.0 with client (mutual) authentication and supports the ciphersuites in the selections in section 5.1.2.31 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**
**FDP_ACF_EXT.2**
**FDP_ACF_EXT.3** ***(KMD)***
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 signature permission is the `android.permission.LOCATION_HARDWARE`, which allows an application to use location features in hardware (such as the geofencing API). The device grants this permission to requesting applications that either have been signed with the same developer key used to sign the android application declaring the permissions or that reside in the "system" directory within Android, which for Android 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 (Android 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 [Android 5.0] 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  shows a series of functional categories centered on 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 33 - Access Control Categories***

Only applications with a common application developer are able to allow sharing of data between the applications. Common applications are those signed by a common certificate or key by the developer that have permissions to allow data sharing in their manifest. Application data can only be shared under this scenario.

The TOE provides the ability to create a Knox Separated Apps folder. A Knox Separated Apps folder provides an administrator with the ability to isolate applications from the broader system. Access to applications placed into the folder does not require separate authentication. Applications within the folder are restricted from accessing services provided outside the folder, such as sharing services (intents) and data storage.

**FDP_DAR_EXT.1**
The TOE provides encryption of all data (which includes both user data and TSF data) stored on the data partition and on external media (such as an SD Card) of the TOE.

The TOE uses FBE to encrypt data using XTS-AES-256 using a unique File Content Encryption Key (FCEK) for each file. File metadata (such as filenames) is encrypted separately with AES-CBC-CTS with a unique File Name Encryption Key (FNEK) for each file. FBE supports two separate classes of protection, credentialed and device. While each class is encrypted, the difference is whether the encryption is chained to the user's credentials. By default, all data is stored in the credential class, and applications that will store data in the device class are defined during the installation of the application. Device class data can be accessed as soon as the device has started, including prior to the first user authentication.

For the protection of data stored on external media (SD Card[4]), the TOE also provides AES-256-CBC encryption of protected data stored using FEKs. The TOE encrypts each individual file stored on the SD Card, generating a unique FEK for each file.

The TOE's system executables, libraries, and their configuration data reside in a read-only file system outside the data partition.

**FDP_DAR_EXT.2** ***(KMD)***

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. This functionality is controlled by the administrator through the apps that support sensitive data storage, and is only available for apps that have been designed to support sensitive data. When an application stores data as sensitive data, the file will be marked as sensitive in the metadata. Based on the environment lock-state, sensitive data protected by this mechanism will be encrypted when locked (either directly by the user or 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 receive and store data securely while the environment is locked (such as an email application).

**FDP_IFC_EXT.1**
**MOD_VPN_CLI_V2.3: 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 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_PBA_EXT.1** ***(KMD)***

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

[4] Samsung Galaxy Flip & Fold devices do not support removable media.

match determination). The image is provided to the biometric service to check the enrolled templates for a match to the captured image.

**MOD_VPN_CLI_V2.3: FDP_RIP.2**

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 certificates (individually stored in `/system/etc/security/cacerts`) and any additional user or admin/MDM loaded certificates. The user can disable the built-in certificates or add new certificates using the TOE's Android user interface [Settings->Security-> Trusted Credentials]. The admin is able to load new certificates using the MDM. 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/APPS**

The TOE provides APIs allowing non-TSF applications (mobile applications) the ability to establish a secure channel using IPsec, TLS, and HTTPS. Mobile applications can use the following Android APIs for IPsec, TLS and HTTPS 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.SSLContex`

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

`javax.net.ssl.HttpsURLConnection`

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

**FDP_UPC_EXT.1/BT**

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

`android.bluetooth`

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

## 6.4 Identification and Authentication

**FIA_AFL_EXT.1**
The TOE maintains two separate lock screens: the device (Android) lock screen and a lock screen for the work profile. Each lock screen maintains individually stored (in separate Flash locations) failed login attempt counters.

The TOE maintains, for each lock screen, 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. For the device lock screen, this would mean a full wipe of all data on the device (a factory reset), while for the work profile lock screen this would remove all data associated with the work profile. 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. An administrator can adjust the number of failed logins to a value between one and 30 through an MDM.

For users with biometrics enabled, biometric authentication attempts are maintained along with the password attempts. In all cases, biometric or hybrid authentication mechanisms are non-critical and cannot be the authentication method that triggers an action (device or work profile wipe).

The maximum number of incorrect password authentication attempts can be configured to a value between 1 and 30. The maximum number of biometric attempts is 10 (this cannot be configured separately and this limit only applies to the device lock screen, not the work profile lock screen). The user can attempt 10 biometric attempts followed by the maximum number of 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. Alternatively, the user might enter 14 incorrect passwords, and then ten failed biometric authentication attempts followed by a final incorrect password attempt.

The TOE's work profile provides its own lock screen, which allows password authentication or hybrid authentication (biometric and password). The hybrid authentication method requires the user to authenticate with a biometric and a password in sequential order. To login, the user must first enter his or her Knox biometric and only upon successfully verifying the user's biometric, the TOE prompts the User for their Knox password, and the user must enter their password. Knox will count the number of incorrect passwords attempted, and wipe the work profile (and its associated data) after the user reaches the configured number of incorrect attempts.

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.

**MOD_BT_V1.0: 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).

**MOD_BT_V1.0: 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).

**MOD_BT_V1.0: 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.

**MOD_BT_V1.0: 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.

**MOD_BT_V1.0: FIA_BLT_EXT.6**
The TOE requires that OPP and MAP profile connections have explicit user authorization before granting access to trusted remote devices on a per service basis (as opposed to a per app basis).

**MOD_BT_V1.0: FIA_BLT_EXT.7**
The TOE requires that untrusted devices have explicit user authorization before granting access to untrusted remote devices for all Bluetooth profiles on a per service basis (as opposed to a per app basis).

**FIA_BMG_EXT.1(1)**
**FIA_BMG_EXT.1(2)** ***(KMD)***
The TOE provides fingerprint biometric authentication. Table 34 - Device biometric sensor shows which biometric subsystems are available on each device.

| Device | Sensors |
|---|---|
| A53 5G /A52 5G/A42 5G/A71 5G/A51 5G/Tab Active3 | Fingerprint-ID |
| Z Fold4 | Fingerprint-U |
| Z Flipx/ XCover6 Pro | Fingerprint-I |

***Table 34 - Device biometric sensor***

In the evaluated configuration, the maximum number of authentication attempts is 40 before a wipe event is triggered. This is broken down into a maximum of 10 biometric attempts and 30 password attempts that 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.

$$SAFAR_{password} = 1 - \left(1 - {}^{1}\!/_{93^{4}}\right)^{30} = 4.010 * 10^{-7}$$

The FAR for fingerprint is 1:10,000 and the FRR is 3%. The SAFAR when a fingerprint is in use is 1:1,000 based on a maximum of 10 attempts of the biometric as noted in the worst-case scenario.

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

For all SAFAR calculations the password is considered a critical factor for all combined factor SAFAR$_{any}$ calculations as detailed in PP_MD_V3.1 section H.4. The values are accepted because they are within the 1% margin allowed by PP_MD_V3.1.

For devices which support fingerprint biometrics, SAFAR$_{fingerprint}$ can be used for the calculation of a worst-case scenario.

$$SAFAR_{any-fingerprint} = 1 - \left(1 - SAFAR_{ff}\right) * (1 - SAFAR_{password})$$

$$SAFAR_{any-ff} = 1.00022 \ * 10^{-3}$$

The Knox Platform for Enterprise provides hybrid (multi-factor: password and biometric) authentication. When using the hybrid authentication mechanism, the user must enter a correct biometric factor and then a correct password in order to successfully unlock the work profile.

The SAFAR when a hybrid mechanism is in use is equal to SAFAR$_{password}$. The maximum number of biometric attempts when using hybrid is 50, but eventually the biometric factor must be entered correctly and then one must always enter a correct password. Given the potential maximum 50 biometric attempts, although that could take a very long time given some imposed delays, that factor is discounted in calculating the SAFAR. 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 affect 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.

**PP_WLAN_CLI_EP_V1.0: 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 supports user passwords consisting of basic Latin characters (upper and lower case, numbers, and the special characters noted in the selection (see section 5.1.4.11)) for authentication to all lock screens (see Table 32). On devices with a boot lock screen, the TOE uses the same configuration for both the boot and device lock screens (i.e. there is one setting that applies to both). The TOE can support a minimum password length of as few as four (4) characters and a maximum of no more than sixteen (16) 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).

### MOD_VPN_CLI_V2.3: FIA_PSK_EXT.1

The TOE supports the use of pre-shared keys (the TOE allows 1 to 64 character PSKs) for IPsec VPNs. 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 MOD_VPN_CLI_V2.3 is supported by the TOE. The TOE does not perform any processing on pre-shared keys. The TOE simply uses the pre-shared key that was entered by the user or administrator.

### 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 allows the following authentication methods at the different lock screens in CC mode. The available biometrics are dependent on the lock screen being used, MDM configuration and enrolled templates. Table 32 shows which authentication methods are available at each lock screen.

| Device Lock | Work Profile Lock |
|---|---|
| Password, Biometrics per (34) | Password, Hybrid |

*Table 32 - Allowed Lock Screen Authentication Methods*

The TOE prohibits other authentication mechanisms such as pattern or swipe. Use of Smart Lock mechanisms (on-body detection, trusted places, trusted devices, trusted face, and trusted voice) and PIN can be blocked through management controls. Upon restart or power-up the user can only use a password for authentication at the first lock screen. Once past this initial authentication screen the user is able to use one of the configured methods at the device lock screen to login and the work profile lock screen.

### FIA_UAU.6

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's two lock screens (device lock screen, and work profile lock screen), by default, briefly display the most recently entered password character and then obscures the character by replacing the displayed character with a dot symbol. The user can configure the TOE's behavior for the device lock screen so that it does not briefly display the last typed character; however, the TOE always briefly displays the last entered character for the work profile lock screen. Additionally, the TOE's device lock screen does not provide any feedback other than a notification of a failed biometric (fingerprint) authentication attempt (“not recognized”). Similarly, the TOE’s work profile lock screen, when configured for hybrid authentication, displays only an indication (“no match”) of a failed biometric attempt.

**FIA_UAU_EXT.1**

The TOE's Key Hierarchy requires the user's password in order to derive the sHEK in order to decrypt other KEKs and DEKs. Thus, until it has the user's password, the TOE cannot decrypt the DEK utilized by FBE 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_X509_EXT.1**

**PP_WLAN_CLI_EP_V1.0: FIA_X509_EXT.1/WLAN**

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 presented in a peer 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**

**PP_WLAN_CLI_EP_V1.0: FIA_X509_EXT.2/WLAN**

The TOE uses X.509v3 certificates as part of EAP-TLS, TLS, HTTPS and IPsec authentication. The TOE comes with a built-in set of default Trusted Credentials (Android's set of trusted CA certificates). 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 or HTTPS 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. When establishing an EAP-TLS connection, the TOE does not check for certificate revocation as the revocation servers are not available until after the EAP-TLS connection is established. 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. When establishing a connection to a VPN server, the VPN first compares 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.

The TOE supports both CRL and OCSP configurations for revocation checking. Only one of these will be used at a time (even if both are configured). OCSP takes precedence over CRL if a certificate provides information for checking both. The process of checking the revocation status of a certificate depends on the configuration on the device set by the admin.

If OCSP is configured (and if the Authority Information Access, AIA, extension is present), OCSP will be used to check the status. If the certificate lacks AIA, or if OCSP is not configured on the device 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 or OCSP server, or does not receive a positive confirmation from the OCSP server, 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 (if it is configured on the device).

**FIA_X509_EXT.3**
The TOE's Android operating system provides applications the `java.security.cert.CertPathValidator` 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_EXT.1**
**FMT_SMF_EXT.1**
The TOE provides the management functions described in Table 8 - Security Management Functions. The table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted. It is worth noting that the TOE's ability to specify authorized application repositories takes the form of allowing enterprise applications (i.e., restricting applications to only those applications installed by an MDM Agent).

**MOD_BT_CLI_V1.0: FMT_SMF_EXT.1/BT**
The TOE provides the management functions described in Table 9 - Bluetooth Security Management Functions. The table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted.

Wi-Fi Direct can be blocked to prevent Bluetooth High Speed access between devices and NFC access can be blocked to prevent out of band pairing.

**PP_WLAN_CLI_EP_V1.0: FMT_SMF_EXT.1/WLAN**
The TOE provides the management functions described in Table 10 - WLAN Security Management Functions. The table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted.

**MOD_VPN_CLI_V2.3: FMT_SMF.1/VPN**
The TOE provides the management functions described in Table 11 - VPN Security Management Functions. 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.

**FMT_SMF_EXT.2**
A user can unenroll an MDM agent from the device in one of two ways. First, a user can revoke the MDM agent's administrative privileges through the Settings app (Settings->Security & Location->Device admin apps) and then uninstall the agent. This method assumes that the MDM agent does block the user from revoking the agent's administrator privileges. When unenrolled in this fashion, the device will remove the work profile, the work profile's applications and data. In effect, this translates to the selections of wiping all sensitive data (all work profile data), removing all Enterprise applications (all work profile applications), removing all device-stored Enterprise application data (all work profile application data), and removing Enterprise secondary authentication data (Knox password and/or fingerprint).

In the case where an MDM agent blocks the user from revoking the agent's administrative privileges, a user can only unenroll by wiping the entire device. By doing this, the user causes the device to wipe all protected data (wiping all data, including both the user's protected data and any Enterprise/work profile data) as well as remove MDM policies and disabling CC mode (as the device returns to factory defaults).

**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.

## 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 12 operating system utilizes 5/10/5.4/4.19/4.9 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 sets the ARM eXecute Never (XN) bit on memory pages and the MMU circuitry of the TOE's ARMv8 Application Processor enforces the XN bits. From Android's security documentation (https://source.android.com/security/), Android 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 hash of the Secure Boot Public Key (a 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 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 signature on either kernel image (primary kernel image or recovery kernel image). The signing key type and hash type used are listed in Table 36 - Secure Boot Public Keys.

| Device | Signing Key | Hash |
|---|---|---|
| A53 5G /A52 5G/A42 5G | ECDSA P384 | SHA-256 |
| A71 5G/A51 5G/Tab Active3 | RSA 2048 | SHA-256 |
| All other devices | RSA 2048 | SHA-256 |

***Table 36 - Secure Boot Public Keys***

Note that when configured for Common Criteria mode, the TOE only accepts updates to the TOE via FOTA; 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 FOTA update signatures (which contain a PKCS 2.1 PSS RSA 2048 w/ SHA-256 signature).

The TOE protects access to the REK and derived HEK to only trusted applications[5] within the TEE (TrustZone). The TOE key manager includes a TEE module that 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

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

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 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 six (6) unpredictable bits. This ensures that at each boot, the location of kernel data structures including the core kernel begins at a random physical address, mapping the core kernel at a random virtual address in the vmalloc area, loading kernel modules at a random virtual address in the vmalloc area, and mapping system memory at a random virtual address in the linear area.

**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). While the AP and CP are part of the same SoC, they are separate physical components within the SoC with no shared components between them (such as memory) and communicate via a non-shared bus (i.e. no other chips can communicate via this bus). 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** ***(KMD)***
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 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 *(KMD)***
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: fscrypt (on devices with FBE), 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_EXT.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 *(KMD)***
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 timeserver) 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 |

| Algorithm | Implemented in | Description |
|---|---|---|
| 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 33 - Power-up Cryptographic Algorithm Self-Tests***

**PP_WLAN_CLI_EP_V1.0: FPT_TST_EXT.1/WLAN**
**MOD_VPN_CLI_V2.3: 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/PREKERNEL**
**FPT_TST_EXT.2/POSTKERNEL**

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.

Once the kernel has been loaded, the TOE 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 file system 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 *(KMD)***

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 the Download mode of the bootloader). After verifying an update's FOTA signature, the TOE will then install those updates to the TOE.

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.

**FPT_TUD_EXT.3**

The TOE requires that all applications be signed before they can be installed. Android uses a file format called APK to package the application installation. The contents of the APK are hashed and then signed, with the resulting APK Signing Block then inserted into the APK or saved as a separate file (depending on the APK Signature Scheme in use).

Once an application has been downloaded, the Signing Block information is used to verify the software before installation. The app contents are verified against the hash values that have been calculated and signed before approving the installation.

In addition to the hash check, there are several rules related to the contents of the Signing Block which can be found here: https://source.android.com/security/apksigning/v3.

**FPT_TUD_EXT.6 *(KMD)***
The TOE maintains a monotonic version counter for the TOE software. Before a new update can be installed, the version of the new software is verified to the version counter. If the new image is older than the current version, the update will be rejected and not installed; only version numbers that are equal to or greater than the currently set number can be installed on the device.

**ALC_TSU_EXT**
Samsung utilizes industry best practices to ensure their devices are patched to mitigate security flaws. Samsung provides a web portal for reporting potential security issues (https://security.samsungmobile.com/securityReporting.smsb) with instructions about how to securely contact and communicate with Samsung. As an Android OEM, also works with Google on reported Android issues (https://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 (https://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. 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 5.1.4.18 before authenticating. However, 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.

On power up the TOE boots to the device lock screen. On the first boot, the user can only make emergency calls, receive calls, enter their password or see notifications from apps that do not require user authentication (apps that have requested the use of Device Encrypted `storage during installation).

**FTA_TAB.1**

The TOE can be configured by an administrator to display a message on the lock screen using an MDM.

**PP_WLAN_CLI_EP_V1.0: FTA_WSE_EXT.1**
The TOE allows an administrator to specify (using 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

**MOD_BT_V1.0: FTP_BLT_EXT.1**
**MOD_BT_V1.0: FTP_BLT_EXT.3/BR**
**MOD_BT_V1.0: FTP_BLT_EXT.3/LE**
The TOE provides support for both Bluetooth BR/EDR and Bluetooth LE connections. The TSF ensures that all connections are encrypted using keys of at least 128-bits (regardless of the type of connection).

**MOD_BT_V1.0: FTP_BLT_EXT.2**
Since the TOE requires an encrypted connection between itself and another Bluetooth device, if the remote device stops encryption, the TSF will force the termination of the connection. A new connection should re-establish the encrypted channel (and if not, the connection will not be successful).

**FTP_ITC_EXT.1**
**PP_WLAN_CLI_EP_V1.0: 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, HTTPS 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 and HTTPS via published APIs that 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 Work Profile Functionality

This section specifically enumerates the functionality specifically provided when a work profile has been created.

**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 profile within the device. Within a work profile a group of applications can be installed, and access to those applications is then restricted to usage solely within the work profile. The work profile boundary restricts the ability of sharing data such that applications outside the work profile cannot see, share or even copy data to those inside the work profile and vice versa. Exceptions to the boundary (such as allowing a copy operation) must be configured by the administrator via policy. Furthermore, the work profile boundary policy can control access to hardware features, such as the camera or microphone, and restrict the ability of applications within the work profile to access those services.

**FIA_AFL_EXT.1**

The work profile maintains, in flash, the number of failed logins since the last successful login, and upon reaching the maximum number of incorrect logins, the work profile performs a full wipe of data protected by Knox (i.e. data inside the work profile). 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 thirty through an MDM.

**FIA_UAU_EXT.4 *(KMD)***

The TOE requires a separate password or hybrid authentication for its work profile, 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 authentication and enrolled a biometric) in order to access any of the Enterprise application data.

**FIA_UAU.5**

The work profile 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**

The work profile requires the user to enter their password in order to unlock the work profile. Additionally, the work profile requires the user to confirm their current password when accessing the "Knox Settings -> Knox unlock method" menu in the work profile user interface. Only after entering their current user password can the user then elect to change their password.

**FIA_UAU.7**

The work profile allows the user to enter the user's password from the lock screen. The work profile 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 work profile obscures the character by replacing the character with a dot symbol.

The work profile can also be configured for hybrid authentication. In this case the user is prompted to first scan their biometric successfully before being prompted to enter their password. If the biometric attempt is unsuccessful, the user is prompted again to enter the biometric. To successfully login, both a valid biometric and the correct password must be entered.

**FMT_MOF.1**
**FMT_SMF_EXT.1**

The work profile grants the admin additional controls over the work profile beyond those available to the device as a whole. The primary additional control is over sharing data to and from the work profile.

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

In general, the management functions for the work profile are the same as those of the device as a whole. Specific differences of the impact of a work profile function are noted in Table 8 - Security Management Functions.

**FTA_SSL_EXT.1**

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

The work profile timeout is independent of the TOE timeout and as such can be set to different values.