OPPO Find X5 Pro on ColorOS 12.1

Security Target

Version:	1.4
Status:	Release
Last Update:	2022-03-16

Revision History

Revision	Date	Author(s)	Changes to Previous Revision
1.4	2022-03-16	Di Li	Released version.

Table of Contents

List of Tables

List of Figures

1 Introduction

This document is the Common Criteria (CC) Security Target (ST) for the OPPO Find X5 Pro on ColorOS 12.1 product to be evaluated as Mobile Devices in exact compliance with:

- PP-Configuration for Mobile Device Fundamentals and Bluetooth Version 1.0, dated 15 April 2021 [CFGMDFBT10]
- Extended Package for Wireless LAN Client Version 1.0, dated 11 February 2016 [WLANCEP10]
- Functional Package for TLS Version 1.1, dated 12 February 2019 [PKGTLS11]

This section contains the Security Target (ST) and Target of Evaluation (TOE) identifications, TOE overview, and TOE description.

The Security Target contains the following additional sections:

- Conformance Claims (Section 2)
- Security Problem Definition (Section 3)
- Security Objectives (Section 4)
- Extended Components Definition (Section 5)
- Security Requirements (Section 6)
- TOE Summary Specification (Section 7)

1.1 Security Target Identification

ST Title:	OPPO Find X5 Pro on ColorOS 12.1 Security Target
Version:	1.4
Status:	Release
Date:	2022-03-16
Sponsor:	Guangdong OPPO Mobile Telecommunications Corp., Ltd
Developer:	Guangdong OPPO Mobile Telecommunications Corp., Ltd
Keywords:	MDFPP32, Common Criteria, mobile device, TLS, HTTPS, Bluetooth, X509 certificate

1.2 TOE Identification

The TOE is OPPO Find X5 Pro on ColorOS 12.1.

The features provided by the TOE is shown below:

Features	OPPO Find X5 Pro
Processor	Qualcomm Snapdragon® 8 Gen 1 Mobile Platform
RAM	12 GB RAM
Storage	256 GB internal memory (non-expandable)
Display	AMOLED 6.70", 20.1:9 ratio (3216 x 1440)
Camera	Rear Tri Cameras with LED Flash. - Main camera:50 MP all pixel omni-directional PDAF f/1.7

	- Ultra-wide angle camera:50 MP all pixel omni-directional PDAF - Telephoto camera: 13 MP, and AF supported Front Camera: 32 MP Lens f/2.4
Communications	5G,4G LTE Network / Mobile Hotspot / Bluetooth 5.2 / Wi-Fi Tethering / Wi-Fi Direct / USB and Bluetooth Tethering / NFC / Media Server / Screen Sharing(Miracast) / GPS, A-GPS, BeiDou Navigation Satellite System, GLONASS, and QZSS positioning systems
Battery	2*2500mAh (typical), two series-connected cells, equivalent to a total capacity of 5000mAh
Biometric	Optical in-display fingerprint readers

1.3 TOE Overview

1.3.1 TOE Type

The TOE Type is personally-owned mobile phone for both personal and enterprise use.

1.3.2 TOE Usage

The TOE is OPPO Find X5 Pro mobile phone running with ColorOS 12.1.

The TOE's OS manages the device hardware and provides the technologies with a rich API set required to implement native applications, it also provides the capability to approve or reject an application based upon the API access that the application requires (or to grant applications access at runtime).

The TOE provides a built-in Mobile Device Management (MDM) framework API, giving management features that may be utilized by external MDM solutions (not part of this evaluation), allowing enterprises to use profiles to control some of the device settings. Security management capabilities are also provided to users via the user interface of the device and to administrators through the installation of Configuration Profiles on the device by using MDM solutions.

The TOE provides cryptographic services for the encryption of data-at-rest within the TOE, for secure communication channels, for protection of Configuration Profiles, and for use by apps. These cryptographic services can also be used to establish a trusted channel to other IT entities.

User data protection is provided by encrypting all the user and mobile application data stored in the user's data partition, restricting access by apps and by restricting access until the user has been successfully authenticated.

User identification and authentication is provided by a user defined passphrase (and supplemented by biometric technologies) where the minimum length of the passphrase, passphrase rules, and the maximum number of consecutive failed authentication attempts can be configured by an administrator. Any kind of Smart Lock mechanism shall be disabled in the CC configuration of the TOE.

The TOE protects itself by having its own code and data protected from unauthorized access (using hardware provided memory protection features), by encrypting internal user and TOE Security Functionality (TSF) data using TSF protected keys and encryption/decryption functions, by self-tests, by ensuring the integrity and authenticity of TSF updates and downloaded apps, and by locking the TOE upon user request or after a defined time of user inactivity.

1.3.3 Required non-TOE Hardware/Software/Firmware

The TOE consists of its hardware and the ColorOS, other components that running with TOE, e.g., user application, wireless AP, authentication server for EAP-TLS mutual authentication, MDM client and server, and mobile data network, are considered as non-TOE components, but they are still required by the TOE to perform administrative management functions or other operational functions for the end user or the administrator.

1.3.4 Major Security Features

This section summarizes the security functions provided by the TOE:

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

1.4 TOE Description

Table 1: The Detailed Description of Evaluated Devices

Device Name	Model Number	Chipset Vendor	CPU	OS Version	Build Number	Kernel Version
OPPO Find X5 Pro	CPH2305	Qualcomm	Snapdragon 8 Gen 1	ColorOS 12.1	CPH2305_11_A.11	Android: 12 Linux kernel: 5.10

1.4.1 Physical Boundaries

The TOE's physical boundary is the physical perimeter of its enclosure. The TOE runs ColorOS as its operating system on the Qualcomm Snapdragon 8 Gen 1 processor (refer to as Application Processor). The TOE does not include the user applications that run on top of the operating system, but does include controls that limit application behavior. Further, the device provides support for downloadable MDM agents to be installed to limit or permit different functionality of the device. There is no built-in MDM agent pre-installed on the device.

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

User documentation listed in Section 1.4.3 is also included in the TOE scope.

1.4.2 Logical Boundaries

This section describes the logical security features offered by the TOE listed in Section 1.3.4.

1.4.2.1 Security audit

The TOE implements a security log and logcat that are each stored in a circular memory buffer. An MDM agent can read/fetch the security logs, can retrieve logcat logs, and then handle appropriately (potentially storing the log to Flash or transmitting its contents to the MDM server). These log methods meet the logging requirements outlined by FAU_GEN.1 in MDFPP32.

1.4.2.2 Cryptographic support

The TOE provides cryptographic services via the following two cryptographic modules:

- BoringSSL ae2bb641735447496bed334c495e4868b981fe32
- Application Processor

BoringSSL is a fork of OpenSSL which is built into shared libraries of ColorOS. The cryptographic functions provided by BoringSSL include symmetric key generation, encryption and decryption, asymmetric key generation and key establishment, cryptographic hashing, and keyed-hash message authentication. The TOE also provides below functions which are used to implement security protocols and the encryption of data-at-rest:

- Random number generation
- Data encryption and decryption
- Signature generation/verification
- Message digest
- Message authentication
- Key generation
- Key wrapping

Application Processor provides a set of FIPS 140-2 certified hardware cryptographic modules, the cryptographic functions provided by Application Processor include symmetric key generation, encryption and decryption, cryptographic hashing, and keyed-hash message authentication. The TOE also provides below functions which are used to implement security protocols and the encryption of data-at-rest:

- Random number generation
- Data encryption and decryption
- Message digest
- Message authentication
- Key generation
- Key derivation

Many of above listed cryptographic functions are also accessible as services to applications running on the TOE allowing application developers to ensure their application meets the required criteria to remain compliant to MDFPP32 standards.

1.4.2.3 User data protection

The TOE controls access to system services by hosted applications, including protection of the Trust Anchor Database. Additionally, User data in files is protected using cryptographic functions, ensuring this data remains protected even if the device gets lost or is stolen. Data is protected such that only the app that owns the data can access it. The TOE's evaluated configuration supports Android Enterprise profiles to provide additional separation between application and application data belonging to the Enterprise profile. Please see the Admin Guide for additional details regarding how to set up and use Enterprise profiles.

1.4.2.4 Identification and authentication

Except for answering calls, making emergency calls, using the cameras, using the flashlight, using the quick settings, and checking notifications, users need to authenticate using a passcode or a biometric (fingerprint / face). The user is required to use the passcode authentication mechanism under the following conditions.

- Turn on or restart the device
- Unlock the device for the first time after reboot
- Update software
- Erase the device
- View or change passcode settings
- Install enterprise profiles

The passcode can be configured for a minimum length, for dedicated passcode policies, and for a maximum lifetime. When entered, passcodes are obscured and the frequency of entering passcodes is limited as well as the number of consecutive failed attempts of entering the passcode.

The TOE also enters a locked state after a (configurable) time of user inactivity, and the user is required to either enter his passcode or use biometric authentication (fingerprint) to unlock the TOE.

External entities connecting to the TOE via a secure protocol (Extensible Authentication Protocol Transport Layer Security (EAP-TLS), Transport Layer Security (TLS)) can be authenticated using X.509 certificates.

1.4.2.5 Security management

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

1.4.2.6 Protection of the TSF

Some of the functions the TOE implements to protect the TSF and TSF data are as follows:

- Protection of cryptographic keys - they are not accessible or exportable using the application processor's hardware.
- Protection of REKs - The TOE disallows all read access to the Root Encryption Key and retains all keys derived from the REK within its the Trusted Execution Environment (TEE). Application software can only use keys derived from the REK by reference and receive the result.
- The TOE 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 to isolate the address spaces of applications from one another to protect those applications.
- Digital signature protection of the TSF image - all updates to the TSF need to be digitally signed.
- Software/firmware integrity self-test upon start-up - the TOE will not go operational when this test fails.
- Digital signature verification for apps.
- Access to defined TSF data and TSF services only when the TOE is unlocked.
- The TOE provides its own timing mechanism to ensure that reliable time information is available (e.g., for log accountability).

1.4.2.7 TOE access

The TSF provides functions to lock the TOE upon request by user or after an administrator configurable time of inactivity.

The TOE also has the capability to display an administrator specified (using the TOE's MDM API) advisory message (banner) when the user unlocks the TOE for the first use after reboot.

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

1.4.2.8 Trusted path/channels

The TOE supports the use of the following cryptographic protocols that define a trusted channel between itself and another trusted IT product.

- IEEE 802.11-2012
- IEEE 802.11ac-2013 (a.k.a. Wi-Fi 5)
- IEEE 802.11ax (a.k.a. Wi-Fi 6)
- IEEE 802.1X

- EAP-TLS (1.0, 1.1, 1.2)
- TLS (1.1, 1.2)
- HTTPS
- Bluetooth (5.0)

1.4.3 TOE Documentation

Reference	Document Name	Version
[CC_GUIDE]	OPPO Find X5 Pro on ColorOS 12.1 Administrator Guidance	1.1

2 CC Conformance Claim

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
- Package Claims:
  - Extended Package for Wireless LAN Client Version 1.0, dated 08 February 2016 [WLANCEP10], package conformant
- PP Claims:
  - Exact conformance: PP-Configuration for Mobile Device Fundamentals and Bluetooth Version 1.0, dated 15 April 2021 [CFGMDFBT10]

    Note this PP-Configuration is composed of the Mobile Device Fundamentals PP and the Bluetooth PP-Module. The Mobile Device Fundamentals PP is Functional Package for TLS Version 1.1 conformant, which implies that the TOE is also conformant to this functional package.
- Technical Decisions, all applicable technical decisions until 2022-03-16:

TD No.	Applied	Rationale
TD0194 – WLANCEP10	Yes	Impacts required audit events
TD0244 – WLANCEP10	Yes	Allows additional TLSC curves
TD0439 – WLANCEP10	Yes	Adds FIA_X509_EXT.1/WLAN
TD0442 – PKGTLS11	Yes	FCS_TLSC_EXT.1.1 apply
TD0469 – PKGTLS11	No	TOE does not support TLS in server mode
TD0470 – WLANCEP10	Yes	FCS_SMF_EXT.1.1/WLAN & FTA_WSE_EXT.1 apply
TD0492 – WLANCEP10	Yes	FCS_TLSC_EXT.1.1/WLAN applies
TD0499 – PKGTLS11	Yes	FCS_TLSC_EXT.1.2 applies
TD0513 – PKGTLS11	Yes	FCS_TLSC_EXT.1.3 applies
TD0517 – WLANCEP10	Yes	FCS_TLSC_EXT.1.1/WLAN and FIA_X509_EXT.2/WLAN apply
TD0588 – PKGTLS11	No	TOE does not support TLS in server mode
TD0596 – MDFPP32	Yes	VPN Traffic Permitted in FDP_IFC_EXT.1

TD0600 – MDFPP32	No	Conformance claim sections updated to allow for MOD_VPNC_V2.3
TD0623 – MDFPP32	Yes	FIA_X509_EXT.2.1 apply

2.1 Conformance Rationale

The ST conforms to MDFPP32/WLANCEP10/MODBT10/PKGTLS11. The security problem definition, security objectives, and security requirements have been drawn from the PPs.

3 Security Problem Definition

The security problem definition has been taken from **MDFPP32** and **WLANCEP10**. It is reproduced here for the convenience of the reader. PP-Module for Bluetooth does not specify any additional threats, organizational security policies or assumptions.

3.1 Threats

T.NETWORK_EAVESDROP (MDFPP32)

An attacker is positioned on a wireless communications channel or elsewhere on the network infrastructure. Attackers may monitor and gain access to data exchanged between the Mobile Device and other endpoints.

T.NETWORK_ATTACK (MDFPP32)

An attacker is positioned on a wireless communications channel or elsewhere on the network infrastructure. Attackers may initiate communications with the Mobile Device or alter communications between the Mobile Device and other endpoints in order to compromise the Mobile Device. These attacks include malicious software update of any applications or system software on the device. These attacks also include malicious web pages or email attachments, which are usually delivered to devices over the network.

T.PHYSICAL_ACCESS (MDFPP32)

An attacker, with physical access, may attempt to access user data on the Mobile Device including credentials. These physical access threats may involve attacks, which attempt to access the device through external hardware ports, impersonate the user authentication mechanisms, through its user interface, and also through direct and possibly destructive access to its storage media. Note: Defending against device re-use after physical compromise is out of scope for this Protection Profile.

T.MALICIOUS_APP (MDFPP32)

Applications loaded onto the Mobile Device may include malicious or exploitable code. This code could be included intentionally or unknowingly by the developer, perhaps as part of a software library. Malicious apps may attempt to exfiltrate data to which they have access. They may also conduct attacks against the platform's system software, which will provide them with additional privileges and the ability to conduct further malicious activities. Malicious applications may be able to control the device's sensors (GPS, camera, microphone) to gather intelligence about the user's surroundings even when those activities do not involve data resident or transmitted from the device. Flawed applications may give an attacker access to perform network-based or physical attacks that otherwise would have been prevented.

T.PERSISTENT_PRESENCE (MDFPP32)

Persistent presence on a device by an attacker implies that the device has lost integrity and cannot regain it. The device has likely lost this integrity due to some other threat vector, yet the continued access by an attacker constitutes an on-going threat in itself. In this case, the device and its data may be controlled by an adversary as well as by its legitimate owner.

T.TSF_FAILURE (WLANCEP10)

Security mechanisms of the TOE generally build up from a primitive set of mechanisms (e.g., memory management, privileged modes of process execution) to more complex sets of mechanisms. Failure of the primitive mechanisms could lead to a compromise in more complex mechanisms, resulting in a compromise of the TSF.

T.UNAUTHORIZED ACCESS (WLANCEP10)

A user may gain unauthorized access to the TOE data and TOE executable code. A malicious user, process, or external IT entity may masquerade as an authorized entity in order to gain unauthorized access to data or TOE resources. A malicious user, process, or external IT entity may misrepresent itself as the TOE to obtain identification and authentication data.

T.UNDETECTED_ACTIONS (WLANCEP10)

Malicious remote users or external IT entities may take actions that adversely affect the security of the TOE. These actions may remain undetected and thus their effects cannot be effectively mitigated.

3.2 Assumptions

A.CONFIG (MDFPP32)

It is assumed that the TOE's security functions are configured correctly in a manner to ensure that the TOE security policies will be enforced on all applicable network traffic flowing among the attached networks.

A.NOTIFY (MDFPP32)

It is assumed that the mobile user will immediately notify the administrator if the Mobile Device is lost or stolen.

A.PRECAUTION (MDFPP32)

It is assumed that the mobile user exercises precautions to reduce the risk of loss or theft of the Mobile Device.

A.PROPER_USER (MDFPP32)

Mobile Device users are not willfully negligent or hostile, and use the device within compliance of a reasonable Enterprise security policy.

A.NO_TOE_BYPASS (WLANCEP10)

Information cannot flow between the wireless client and the internal wired network without passing through the TOE.

A.TRUSTED_ADMIN (WLANCEP10)

TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.

3.3 Organizational Security Policies

There are no OSPs for the Mobile Device.

4 Security Objectives

The security objectives for the TOE have been taken from **MDFPP32** and **WLANCEP10**. It is reproduced here for the convenience of the reader. MDFPP32 offers additional information about the identified security objectives as well as a security objectives rationale, but that has not been reproduced here and MDFPP32 should be consulted if there is interest in that material. PP-Module for Bluetooth does not define any additional security objectives.

4.1 Security Objectives for the TOE

The security objectives for the Mobile Device are defined as follows. They are reproduced here for the convenience of the reader.

O.PROTECTED_COMMS (MDFPP32)

To address the network eavesdropping (T.EAVESDROP) and network attack (T.NETWORK) threats described in Section 3.1 Threats, concerning wireless transmission of Enterprise and user data and configuration data between the TOE and remote network entities, conformant TOEs will use a trusted communication path. The TOE will be capable of communicating using one (or more) of these standard protocols: IPsec, DTLS, TLS, HTTPS, or Bluetooth. The protocols are specified by RFCs that offer a variety of implementation choices. Requirements have been imposed on some of these choices (particularly those for cryptographic primitives) to provide interoperability and resistance to cryptographic attack.

While conformant TOEs must support all of the choices specified in the ST including any optional SFRs defined in this PP, they may support additional algorithms and protocols. If such additional mechanisms are not evaluated, guidance must be given to the administrator to make clear the fact that they were not evaluated.

O.STORAGE (MDFPP32)

To address the issue of loss of confidentiality of user data in the event of loss of a Mobile Device (T.PHYSICAL), conformant TOEs will use data-at-rest protection. The TOE will be capable of encrypting data and keys stored on the device and will prevent unauthorized access to encrypted data.

O.CONFIG (MDFPP32)

To ensure a Mobile Device protects user and enterprise data that it may store or process, conformant TOEs will provide the capability to configure and apply security policies defined by the user and the Enterprise Administrator. If Enterprise security policies are configured these must be applied in precedence of user specified security policies.

O.AUTH (MDFPP32)

To address the issue of loss of confidentiality of user data in the event of loss of a Mobile Device (T.PHYSICAL), users are required to enter an authentication factor to the device prior to accessing protected functionality and data. Some non-sensitive functionality (e.g., emergency calling, text notification) can be accessed prior to entering the authentication factor. The device will automatically lock following a configured period of inactivity in an attempt to ensure authorization will be required in the event of the device being lost or stolen.

Authentication of the endpoints of a trusted communication path is required for network access to ensure attacks are unable to establish unauthorized network connections to undermine the integrity of the device.

Repeated attempts by a user to authorize to the TSF will be limited or throttled to enforce a delay between unsuccessful attempts.

O.INTEGRITY (MDFPP32)

To ensure the integrity of the Mobile Device is maintained conformant TOEs will perform self-tests to ensure the integrity of critical functionality, software/firmware and data has been maintained. The user shall be notified of any failure of these self-tests. This will protect against the threat T.PERSISTENT.

To address the issue of an application containing malicious or flawed code (T.FLAWAPP), the integrity of downloaded updates to software/firmware will be verified prior to installation/execution of the object on the Mobile Device. In

addition, the TOE will restrict applications to only have access to the system services and data they are permitted to interact with. The TOE will further protect against malicious applications from gaining access to data they are not authorized to access by randomizing the memory layout.

O.PRIVACY (MDFPP32)

In a BYOD environment (use cases 3 and 4), a personally-owned mobile device is used for both personal activities and enterprise data. Enterprise management solutions may have the technical capability to monitor and enforce security policies on the device. However, the privacy of the personal activities and data must be ensured. In addition, since there are limited controls that the enterprise can enforce on the personal side, separation of personal and enterprise data is needed. This will protect against the T.FLAWAPP and T.PERSISTENT threats.

O.AUTH_COMM (WLANCEP10)

The TOE will provide a means to ensure that it is communicating with an authorized Access Point and not some other entity pretending to be an authorized Access Point, and will provide assurance to the Access Point of its identity.

O.CRYPTOGRAPHIC_FUNCTIONS (WLANCEP10)

The TOE shall provide or use cryptographic functions (i.e., encryption/decryption and digital signature operations) to maintain the confidentiality and allow for detection of modification of data that are transmitted outside the TOE and its host environment.

O.SYSTEM_MONITORING (WLANCEP10)

The TOE will provide the capability to generate audit data.

O.TOE_ADMINISTRATION (WLANCEP10)

The TOE will provide mechanisms to allow administrators to be able to configure the TOE.

O.TSF_SELF_TEST (WLANCEP10)

The TOE will provide the capability to test some subset of its security functionality to ensure it is operating properly.

O.WIRELESS_ACCESS_POINT_CONNECTION (WLANCEP10)

The TOE will provide the capability to restrict the wireless access points to which it will connect.

4.2 Security Objectives for the Operational Environment

OE.CONFIG (MDFPP32)

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

OE.NOTIFY (MDFPP32)

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

OE.PRECAUTION (MDFPP32)

The mobile device user exercises precautions to reduce the risk of loss or theft of the Mobile Device.

OE.DATA_PROPER_USER (MDFPP32)

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 (WLANCEP10)

Information cannot flow between external and internal networks located in different enclaves without passing through the TOE.

OE.TRUSTED_ADMIN (WLANCEP10)

TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.

5 Extended Components Definition

All of the extended requirements in this ST have been drawn from MDFPP32, WLANCEP10, PKGTLS11 and MODBT10. MDFPP32, WLANCEP10, PKGTLS11 and MODBT10 define the following extended requirements and, since they are not redefined in this ST, MDFPP32, WLANCEP10, PKGTLS11 and MODBT10 should be consulted for more information in regard to those CC extensions.

Extended SFRs:

- MDFPP32: FCS_CKM_EXT.1 Cryptographic Key Support
- MDFPP32: FCS_CKM_EXT.2 Cryptographic Key Random Generation
- MDFPP32: FCS_CKM_EXT.3 Cryptographic Key Generation
- MDFPP32: FCS_CKM_EXT.4 Key Destruction
- MDFPP32: FCS_CKM_EXT.5 TSF Wipe
- MDFPP32: FCS_CKM_EXT.6 Salt Generation
- MODBT10: FCS_CKM_EXT.8 Bluetooth Key Generation
- MDFPP32: FCS_HTTPS_EXT.1 HTTPS Protocol
- MDFPP32: FCS_IV_EXT.1 Initialization Vector Generation
- MDFPP32: FCS_RBG_EXT.1 Random Bit Generation
- MDFPP32: FCS_SRV_EXT.1 Cryptographic Algorithm Services
- MDFPP32: FCS_STG_EXT.1 Cryptographic Key Storage
- MDFPP32: FCS_STG_EXT.2 Encrypted Cryptographic Key Storage
- MDFPP32: FCS_STG_EXT.3 Integrity of Encrypted Key Storage
- PKGTLS11: PKGTLS11: FCS_TLS_EXT.1 TLS Protocol
- PKGTLS11: FCS_TLSC_EXT.1 TLS Client Protocol
- WLANCEP10: FCS_TLSC_EXT.1/WLAN Extensible Authentication Protocol-Transport Layer Security
- PKGTLS11: FCS_TLSC_EXT.2 TLS Client Support for Mutual Authentication
- WLANCEP10: FCS_TLSC_EXT.2/WLAN TLS Client Protocol
- PKGTLS11: FCS_TLSC_EXT.4 TLS Client Support for Renegotiation
- PKGTLS11: FCS_TLSC_EXT.5 TLS Client Support for Supported Groups Extension
- MDFPP32: FDP_ACF_EXT.1 Access Control for System Services
- MDFPP32: FDP_ACF_EXT.2 Access Control for System Resources
- MDFPP32: FDP_DAR_EXT.1 Protected Data Encryption
- MDFPP32: FDP_DAR_EXT.2 Sensitive Data Encryption
- MDFPP32: FDP_IFC_EXT.1 Subset Information Flow Control
- MDFPP32: FDP_PBA_EXT.1: Extended: Storage of Critical Biometric Parameters
- MDFPP32: FDP_STG_EXT.1 User Data Storage
- MDFPP32: FDP_UPC_EXT.1/APPS Inter-TSF User Data Transfer Protection (Applications)
- MDFPP32: FIA_AFL_EXT.1 Authentication Failure Handling
- MODBT10: FIA_BLT_EXT.1 Bluetooth User Authorization

- MODBT10: FIA_BLT_EXT.2 Bluetooth Mutual Authentication
- MODBT10: FIA_BLT_EXT.3 Rejection of Duplicate Bluetooth Connections
- MODBT10: FIA_BLT_EXT.4 Secure Simple Pairing
- MODBT10: FIA_BLT_EXT.6 Trusted Bluetooth Device User Authorization
- MODBT10: FIA_BLT_EXT.7 Untrusted Bluetooth Device User Authorization
- MDFPP32: FIA_BMG_EXT.1/FINGERPRINT: Extended: Accuracy of Biometric Authentication
- MDFPP32: FIA_BMG_EXT.1/FACE: Extended: Accuracy of Biometric Authentication
- WLANCEP10: FIA_PAE_EXT.1 Port Access Entity Authentication
- MDFPP32: FIA_PMG_EXT.1 Password Management
- MDFPP32: FIA_TRT_EXT.1 Authentication Throttling
- MDFPP32: FIA_UAU_EXT.1 Authentication for Cryptographic Operation
- MDFPP32: FIA_UAU_EXT.2 Timing of Authentication
- MDFPP32: FIA_X509_EXT.1 X.509 Validation of Certificates
- MDFPP32: FIA_X509_EXT.2 X.509 Certificate Authentication
- WLANCEP10: FIA_X509_EXT.2/WLAN X.509 Certificate Authentication (EAP-TLS)
- MDFPP32: FIA_X509_EXT.3 Request Validation of Certificates
- MDFPP32: FMT_MOF_EXT.1 Management of Security Functions Behavior
- MDFPP32: FMT_SMF_EXT.1 Specification of Management Functions
- MODBT10: FMT_SMF_EXT.1/BT Specification of Management Functions
- WLANCEP10: FMT_SMF_EXT.1/WLAN Specification of Management Functions (Wireless LAN)
- MDFPP32: FMT_SMF_EXT.2 Specification of Remediation Actions
- MDFPP32: FPT_AEX_EXT.1 Application Address Space Layout Randomization
- MDFPP32: FPT_AEX_EXT.2 Memory Page Permissions
- MDFPP32: FPT_AEX_EXT.3 Stack Overflow Protection
- MDFPP32: FPT_AEX_EXT.4 Domain Isolation
- MDFPP32: FPT_JTA_EXT.1 JTAG Disablement
- MDFPP32: FPT_KST_EXT.1 Key Storage
- MDFPP32: FPT_KST_EXT.2 No Key Transmission
- MDFPP32: FPT_KST_EXT.3 No Plaintext Key Export
- MDFPP32: FPT_NOT_EXT.1 Self-Test Notification
- MDFPP32: FPT_TST_EXT.1 TSF Cryptographic Functionality Testing
- WLANCEP10: FPT_TST_EXT.1/WLAN TSF Cryptographic Functionality Testing (Wireless LAN)
- MDFPP32: FPT_TST_EXT.2/PREKERNEL TSF Integrity Checking (Pre-Kernel)
- MDFPP32: FPT_TUD_EXT.1 Trusted Update: TSF Version Query
- MDFPP32: FPT_TUD_EXT.2 TSF Update Verification
- MDFPP32: FPT_TUD_EXT.3 Application Signing

- MDFPP32: FTA_SSL_EXT.1 TSF- and User-initiated Locked State
- WLANCEP10: FTA_ WSE_EXT.1 Wireless Network Access
- MODBT10: FTP_BLT_EXT.1 Bluetooth Encryption
- MODBT10: FTP_BLT_EXT.2 Persistence of Bluetooth Encryption
- MODBT10: FTP_BLT_EXT.3/BR Bluetooth Encryption Parameters (BR/EDR)
- MODBT10: FTP_BLT_EXT.3/LE Bluetooth Encryption Parameters (LE)
- MDFPP32: FTP_ITC_EXT.1 Trusted Channel Communication
- WLANCEP10: FTP_ITC_EXT.1/WLAN Trusted Channel Communication (Wireless LAN)

Extended SARs:

- ALC_TSU_EXT.1: Timely Security Updates

6 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 MDFPP32, WLANCEP10, MODBT10 and PKGTLS11. The refinements and operations already performed in above listed PPs are not identified (e.g., highlighted) here, rather the requirements have been copied from above listed PPs and any residual operations have been completed herein. Of particular note, above listed PPs made a number of refinements and completed some of the SFR operations defined in the Common Criteria (CC) and that above listed PPs should be consulted to identify those changes if necessary.

The SARs are also drawn from MDFPP32 which includes all the SARs for EAL 1 augmented with ALC_TSU_EXT.1. However, the SARs are effectively refined since requirement-specific 'Assurance Activities' are defined in MDFPP32, WLANCEP10, MODBT10 and PKGTLS11 that serve to ensure corresponding evaluations will yield more practical and consistent assurance than the EAL 1 assurance requirements alone. MDFPP32 should be consulted for the assurance activity definitions. WLANCEP10 and MODBT10 do not define any SARs beyond those defined within the base MDFPP32.

Conventions

The following conventions have been applied in this document:

- Security Functional Requirements – Part 1 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.
    - 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 (e.g., "… **all objects**"), and strikethrough for deletions (e.g., "… ~~some legacy protocol~~ …").
    - Iteration operation: is indicated by appending the SFR name with a slash and unique identifier suggesting the purpose of the operation, e.g., "/EXAMPLE1."

6.1 TOE Security Functional Requirements

The following table identifies the SFRs that are implemented by TOE.

Table 2: TOE Security Functional Components

Requirement Class	Requirement Component
FAU: Security Audit	MDFPP32: FAU_GEN.1 Audit Data Generation
	MODBT10: FAU_GEN.1/BT Audit Data Generation (Bluetooth)
	WLANCEP10: FAU_GEN.1/WLAN Audit Data Generation (Wireless LAN)
	MDFPP32: FAU_STG.1 Audit Storage Protection
	MDFPP32: FAU_STG.4 Prevention of Audit Data Loss
FCS: Cryptographic support	MDFPP32: FCS_CKM.1 Cryptographic key generation

	WLANCEP10: FCS_CKM.1/WLAN Cryptographic key generation (Symmetric Keys for WPA2 Connections)
	MDFPP32: FCS_CKM.2/UNLOCKED Cryptographic Key Establishment
	MDFPP32: FCS_CKM.2/LOCKED Cryptographic Key Establishment
	WLANCEP10: FCS_CKM.2/WLAN Cryptographic Key Distribution (GTK)
	MDFPP32: FCS_CKM_EXT.1 Cryptographic Key Support
	MDFPP32: FCS_CKM_EXT.2 Cryptographic Key Random Generation
	MDFPP32: FCS_CKM_EXT.3 Cryptographic Key Generation
	MDFPP32: FCS_CKM_EXT.4 Key Destruction
	MDFPP32: FCS_CKM_EXT.5 TSF Wipe
	MDFPP32: FCS_CKM_EXT.6 Salt Generation
	MODBT10: FCS_CKM_EXT.8 Bluetooth Key Generation
	MDFPP32: FCS_COP.1/ENCRYPT Cryptographic Operation
	MDFPP32: FCS_COP.1/HASH Cryptographic Operation
	MDFPP32: FCS_COP.1/SIGN Cryptographic Operation
	MDFPP32: FCS_COP.1/KEYHMAC Cryptographic Operation
	MDFPP32: FCS_COP.1/CONDITION Cryptographic Operation
	MDFPP32: FCS_HTTPS_EXT.1 HTTPS Protocol
	MDFPP32: FCS_IV_EXT.1 Initialization Vector Generation
	MDFPP32: FCS_RBG_EXT.1 Random Bit Generation
	MDFPP32: FCS_SRV_EXT.1 Cryptographic Algorithm Services
	MDFPP32: FCS_STG_EXT.1 Cryptographic Key Storage
	MDFPP32: FCS_STG_EXT.2 Encrypted Cryptographic Key Storage
	MDFPP32: FCS_STG_EXT.3 Integrity of Encrypted Key Storage
	PKGTLS11: PKGTLS11: FCS_TLS_EXT.1 TLS Protocol
	PKGTLS11: FCS_TLSC_EXT.1 TLS Client Protocol

	WLANCEP10: FCS_TLSC_EXT.1/WLAN Extensible Authentication Protocol-Transport Layer Security
	PKGTLS11: FCS_TLSC_EXT.2 TLS Client Support for Mutual Authentication
	WLANCEP10: FCS_TLSC_EXT.2/WLAN TLS Client Protocol
	PKGTLS11: FCS_TLSC_EXT.4 TLS Client Support for Renegotiation
	PKGTLS11: FCS_TLSC_EXT.5 TLS Client Support for Supported Groups Extension
FDP: User data protection	MDFPP32: FDP_ACF_EXT.1 Access Control for System Services
	MDFPP32: FDP_ACF_EXT.2 Access Control for System Resources
	MDFPP32: FDP_DAR_EXT.1 Protected Data Encryption
	MDFPP32: FDP_DAR_EXT.2 Sensitive Data Encryption
	MDFPP32: FDP_IFC_EXT.1 Subset Information Flow Control
	MDFPP32: FDP_PBA_EXT.1: Extended: Storage of Critical Biometric Parameters
	MDFPP32: FDP_STG_EXT.1 User Data Storage
	MDFPP32: FDP_UPC_EXT.1/APPS Inter-TSF User Data Transfer Protection (Applications)
FIA: Identification and authentication	MDFPP32: FIA_AFL_EXT.1 Authentication Failure Handling
	MODBT10: FIA_BLT_EXT.1 Bluetooth User Authorization
	MODBT10: FIA_BLT_EXT.2 Bluetooth Mutual Authentication
	MODBT10: FIA_BLT_EXT.3 Rejection of Duplicate Bluetooth Connections
	MODBT10: FIA_BLT_EXT.4 Secure Simple Pairing
	MODBT10: FIA_BLT_EXT.6 Trusted Bluetooth Device User Authorization
	MODBT10: FIA_BLT_EXT.7 Untrusted Bluetooth Device User Authorization
	MDFPP32: FIA_BMG_EXT.1/FINGERPRINT: Extended: Accuracy of Biometric Authentication
	MDFPP32: FIA_BMG_EXT.1/FACE: Extended: Accuracy of Biometric Authentication
	WLANCEP10: FIA_PAE_EXT.1 Port Access Entity Authentication

	MDFPP32: FIA_PMG_EXT.1 Password Management
	MDFPP32: FIA_TRT_EXT.1 Authentication Throttling
	MDFPP32: FIA_UAU.5 Multiple Authentication Mechanisms
	MDFPP32: FIA_UAU.6 Re-Authentication
	MDFPP32: FIA_UAU.7 Protected Authentication Feedback
	MDFPP32: FIA_UAU_EXT.1 Authentication for Cryptographic Operation
	MDFPP32: FIA_UAU_EXT.2 Timing of Authentication
	MDFPP32: FIA_X509_EXT.1 X.509 Validation of Certificates
	MDFPP32: FIA_X509_EXT.2 X.509 Certificate Authentication
	WLANCEP10: FIA_X509_EXT.2/WLAN X.509 Certificate Authentication (EAP-TLS)
	MDFPP32: FIA_X509_EXT.3 Request Validation of Certificates
FMT: Security management	MDFPP32: FMT_MOF_EXT.1 Management of Security Functions Behavior
	MDFPP32: FMT_SMF_EXT.1 Specification of Management Functions
	MODBT10: FMT_SMF_EXT.1/BT Specification of Management Functions
	WLANCEP10: FMT_SMF_EXT.1/WLAN Specification of Management Functions (Wireless LAN)
	MDFPP32: FMT_SMF_EXT.2 Specification of Remediation Actions
FPT: Protection of the TSF	MDFPP32: FPT_AEX_EXT.1 Application Address Space Layout Randomization
	MDFPP32: FPT_AEX_EXT.2 Memory Page Permissions
	MDFPP32: FPT_AEX_EXT.3 Stack Overflow Protection
	MDFPP32: FPT_AEX_EXT.4 Domain Isolation
	MDFPP32: FPT_JTA_EXT.1 JTAG Disablement
	MDFPP32: FPT_KST_EXT.1 Key Storage
	MDFPP32: FPT_KST_EXT.2 No Key Transmission
	MDFPP32: FPT_KST_EXT.3 No Plaintext Key Export
	MDFPP32: FPT_NOT_EXT.1 Self-Test Notification

	MDFPP32: FPT_STM.1 Reliable Time Stamps
	MDFPP32: FPT_TST_EXT.1 TSF Cryptographic Functionality Testing
	WLANCEP10: FPT_TST_EXT.1/WLAN TSF Cryptographic Functionality Testing (Wireless LAN)
	MDFPP32: FPT_TST_EXT.2/PREKERNEL TSF Integrity Checking (Pre-Kernel)
	MDFPP32: FPT_TUD_EXT.1 Trusted Update: TSF Version Query
	MDFPP32: FPT_TUD_EXT.2 TSF Update Verification
	MDFPP32: FPT_TUD_EXT.3 Application Signing
FTA: TOE access	MDFPP32: FTA_SSL_EXT.1 TSF- and User-initiated Locked State
	WLANCEP10: FTA_ WSE_EXT.1 Wireless Network Access
FTP: Trusted path/channels	MODBT10: FTP_BLT_EXT.1 Bluetooth Encryption
	MODBT10: FTP_BLT_EXT.2 Persistence of Bluetooth Encryption
	MODBT10: FTP_BLT_EXT.3/BR Bluetooth Encryption Parameters
	MODBT10: FTP_BLT_EXT.3/LE Bluetooth Encryption Parameters
	MDFPP32: FTP_ITC_EXT.1 Trusted Channel Communication
	WLANCEP10: FTP_ITC_EXT.1/WLAN Trusted Channel Communication (Wireless LAN)

6.1.1 Security Audit (FAU)

6.1.1.1 Audit Data Generation (FAU_GEN.1)

MDFPP32: 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 3
7. [*no additional auditable events*]

Note: FAU_GEN.1/BT and FAU_GEN.1/WLAN have been merged into FAU_GEN.1 and Table 3 combined all the mandatory auditable events specified in MDFPP32, Bluetooth PP Module and WLAN Client EP.

MDFPP32: 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 3
6. [***no additional information***]

Table 3: Mandatory Auditable Events

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	None.	
FAU_GEN.1/WLAN	None.	
FAU_STG.1	None.	
FAU_STG.4	None.	
FCS_CKM_EXT.1	**[*None*].**	No additional information.
FCS_CKM_EXT.2	None.	
FCS_CKM_EXT.3	None.	
FCS_CKM_EXT.4	None.	
FCS_CKM_EXT.5	**[*None*].**	No additional information.
FCS_CKM_EXT.6	None.	
FCS_CKM_EXT.8	None.	
FCS_CKM.1	**[*None*].**	No additional information.
FCS_CKM.1/WLAN	None.	
FCS_CKM.2/UNLOCKED	None.	
FCS_CKM.2/LOCKED	None.	
FCS_CKM.2/WLAN	None.	
FCS_COP.1/ENCRYPT	None.	
FCS_COP.1/HASH	None.	

FCS_COP.1/SIGN	None.	
FCS_COP.1/KEYHMAC	None.	
FCS_COP.1/CONDITION	None.	
FCS_IV_EXT.1	None.	
FCS_SRV_EXT.1	None.	
FCS_STG_EXT.1	Import or destruction of key.	Identity of key. Role and identity of requestor.
	[*No other events*]	
FCS_STG_EXT.2	None.	
FCS_STG_EXT.3	Failure to verify integrity of stored key.	Identity of key being verified.
FCS_TLSC_EXT.1/WLAN	Failure to establish an EAP-TLS session.	Reason for failure.
	Establishment/termination of an EAP-TLS session.	Non-TOE endpoint of connection.
FDP_DAR_EXT.1	[*None*].	No additional information.
FDP_DAR_EXT.2	Failure to encrypt/decrypt data.	No additional information
FDP_IFC_EXT.1	None.	
FDP_STG_EXT.1	Addition or removal of certificate from Trust Anchor Database.	Subject name of certificate.
FIA_BLT_EXT.1	Failed user authorization of Bluetooth device.	User authorization decision (e.g., user rejected connection, incorrect pin entry).
	Failed user authorization for local Bluetooth Service.	**Bluetooth address and name of device. Bluetooth profile. Identity of local service with [*service ID*].**
FIA_BLT_EXT.2	Initiation of Bluetooth connection.	Bluetooth address and name of device.
	Failure of Bluetooth connection.	Reason for failure.
FIA_BLT_EXT.4	None.	
FIA_BLT_EXT.6	None.	
FIA_BLT_EXT.7	None.	

FIA_PAE_EXT.1	None.	
FIA_PMG_EXT.1	None.	
FIA_TRT_EXT.1	None.	
FIA_UAU_EXT.1	None.	
FIA_UAU.5	None.	
FIA_UAU.7	None.	
FIA_X509_EXT.1	Failure to validate X.509v3 certificate.	Reason for failure of validation.
FIA_X509_EXT.2/WLAN	None.	
FMT_MOF_EXT.1	None.	
FMT_SMF_EXT.1/WLAN	None.	
FPT_AEX_EXT.1	None.	
FPT_AEX_EXT.2	None.	
FPT_AEX_EXT.3	None.	
FPT_JTA_EXT.1	None.	
FPT_KST_EXT.1	None.	
FPT_KST_EXT.2	None.	
FPT_KST_EXT.3	None.	
FPT_NOT_EXT.1	**[*None*].**	**[*No additional information*].**
FPT_STM.1	None.	
FPT_TST_EXT.1	Initiation of self-test.	
	Failure of self-test.	**[*None*]**
FPT_TST_EXT.1/WLAN **(note: can be performed by TOE or TOE platform)**	Execution of this set of TSF self-tests. [***none***].	[***No additional information***] (Done as part of FPT_TST_EXT.1)
FPT_TST_EXT.2/PREKERNEL	Start-up of TOE.	No additional information.
	[*None*]	**[*No additional information*]**
FPT_TUD_EXT.1	None.	

FTA_SSL_EXT.1	None.	
FTA_WSE_EXT.1	All attempts to connect to access points.	Identity of access point being connected to as well as success and failures (including reason for failure).
FTP_BLT_EXT.1	None.	
FTP_BLT_EXT.2	None.	
FTP_BLT_EXT.3/BR	None.	
FTP_BLT_EXT.3/LE **(if claimed)**	None.	
FTP_ITC_EXT.1/WLAN	All attempts to establish a trusted channel. (TD0194 applied)	Identification of the non-TOE endpoint of the channel.

6.1.1.2 Audit Storage Protection (FAU_STG.1)

MDFPP32: FAU_STG.1.1

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

MDFPP32: FAU_STG.1.2

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

6.1.1.3 Prevention of Audit Data Loss (FAU_STG.4)

MDFPP32: FAU_STG.4.1

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

6.1.2 Cryptographic support (FCS)

6.1.2.1 Cryptographic key generation (FCS_CKM.1)

MDFPP32: FCS_CKM.1.1

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

- *RSA schemes using cryptographic key sizes of 2048-bit or greater that meet FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3,*
- *ECC schemes using [*
    - *"NIST curves" P-384 and [P-256] that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4],*

].

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

WLANCEP10: FCS_CKM.1.1/WLAN

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]***

6.1.2.3 Cryptographic key establishment (FCS_CKM.2)

MDFPP32: FCS_CKM.2.1/UNLOCKED

The TSF shall perform cryptographic key establishment in accordance with a specified cryptographic key establishment method [

- ***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"]***
- ***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"***

].

MDFPP32: FCS_CKM.2.1/LOCKED

The TSF shall perform cryptographic key establishment in accordance with a specified cryptographic key establishment method: [

- ***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"***

] for the purposes of encrypting sensitive data received while the device is locked.

6.1.2.4 Cryptographic Key Distribution (GTK) (FCS_CKM.2/WLAN)

WLANCEP10: FCS_CKM.2.1/WLAN

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.

6.1.2.5 Extended: Cryptographic Key Support (FCS_CKM_EXT.1)

MDFPP32: FCS_CKM_EXT.1.1

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

MDFPP32: FCS_CKM_EXT.1.2

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

MDFPP32: FCS_CKM_EXT.1.3

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

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

MDFPP32: FCS_CKM_EXT.2.1

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

6.1.2.7 Extended: Cryptographic Key Generation (FCS_CKM_EXT.3)

MDFPP32: FCS_CKM_EXT.3.1

The TSF shall use [

- *asymmetric KEKs of [112 bits] security strength*,
- *symmetric KEKs of [256-bit] security strength corresponding to at least the security strength of the keys encrypted by the KEK*

].

MDFPP32: 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 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), encrypting one key with another]*

].

6.1.2.8 Extended: Key Destruction (FCS_CKM_EXT.4)

MDFPP32: 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 block erase that erases the reference to memory that stores data as well as the data itself*].
  - 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.

MDFPP32: FCS_CKM_EXT.4.2

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

6.1.2.9 Extended: TSF Wipe (FCS_CKM_EXT.5)

MDFPP32: 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 PD 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.*

].

MDFPP32: FCS_CKM_EXT.5.2

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

6.1.2.10 Extended: Salt Generation (FCS_CKM_EXT.6)

MDFPP32: FCS_CKM_EXT.6.1

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

6.1.2.11 Extended: Bluetooth Key Generation (FCS_CKM_EXT.8)

MODBT10: FCS_CKM_EXT.8.1

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

6.1.2.12 Cryptographic operation (FCS_COP.1/ENCRYPT)

MDFPP32: 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), and
- [
  - *AES Key Wrap (KW) (as defined in NIST SP 800-38F),*
  - *AES-GCM (as defined in NIST SP 800-38D),*
  - *AES-XTS (as defined in NIST SP 800-38E) mode*
  - *AES-GCMP-256 (as defined in NIST SP800-38D and IEEE 802.11ac-2013)*

]

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

6.1.2.13 Cryptographic operation (FCS_COP.1/HASH)

MDFPP32: 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 bits***] that meet the following: FIPS Pub 180-4.

6.1.2.14 Cryptographic operation (FCS_COP.1/SIGN)

MDFPP32: 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,*
- *ECDSA schemes using "NIST curve" P-384 and [P-256] that meet the following: FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Section 5*

].

6.1.2.15 Cryptographic operation (FCS_COP.1/KEYHMAC)

MDFPP32: 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".

6.1.2.16 Cryptographic operation (FCS_COP.1/CONDITION)

MDFPP32: FCS_COP.1.1/CONDITION

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

6.1.2.17 Extended: HTTPS Protocol (FCS_HTTPS_EXT.1)

MDFPP32: FCS_HTTPS_EXT.1.1

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

MDFPP32: FCS_HTTPS_EXT.1.2

The TSF shall implement HTTPS using TLS as defined in the Package for Transport Layer Security.

MDFPP32: FCS_HTTPS_EXT.1.3

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

6.1.2.18 Extended: Initialization Vector Generation (FCS_IV_EXT.1)

MDFPP32: FCS_IV_EXT.1.1

The TSF shall generate IVs in accordance with Table 13 in MDFPP32: References and IV Requirements for NIST-approved Cipher Modes.

6.1.2.19 Extended: Random Bit Generation (FCS_RBG_EXT.1)

MDFPP32: 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), CTR_DRBG (AES)***].

MDFPP32: 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.

MDFPP32: 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.

6.1.2.20 Extended: Cryptographic Algorithm Services (FCS_SRV_EXT.1)

MDFPP32: 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 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

[

- ***All mandatory and [selected algorithms] in FCS_CKM.1,***

].

6.1.2.21 Extended: Cryptographic Key Storage (FCS_STG_EXT.1)

MDFPP32: FCS_STG_EXT.1.1

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

MDFPP32: 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***].

MDFPP32: 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***].

MDFPP32: 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 [***the administrator, a common application developer***].

MDFPP32: 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 [***the administrator, a common application developer***].

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

MDFPP32: FCS_STG_EXT.2.1

The TSF shall encrypt all DEKs, KEKs, [**WPA2 WiFi PSK, Bluetooth Keys**] and [***all software-based key storage***] by KEKs that are [

- ***Protected by the REK with [***
  - ***encryption by a KEK chaining from a REK,***
  - ***encryption by a KEK that is derived from a REK],***

  ],
- ***Protected by the REK and the password with [***
  - ***encryption by a KEK chaining to a REK and the password-derived or biometric-unlocked KEK***
  - ***encryption by a KEK that is derived from a REK and the password-derived or biometric-unlocked KEK]***

].

MDFPP32: FCS_STG_EXT.2.2

DEKs, KEKs, [***WPA2 WiFi PSK, Bluetooth Keys***] 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]***

].

6.1.2.23 Extended: Integrity of encrypted key storage (FCS_STG_EXT.3)

MDFPP32: FCS_STG_EXT.3.1

The TSF shall protect the integrity of any encrypted DEKs and KEKs and [***[WPA2 WiFi PSK, Bluetooth Keys], all software-based key storage***] by [

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

].

MDFPP32: FCS_STG_EXT.3.2

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

6.1.2.24 Extended: TLS Protocol (FCS_TLS_EXT.1)

PKGTLS11: FCS_TLS_EXT.1.1

The product shall implement [

- ***TLS as a client***

]

6.1.2.25 Extended: TLS Client Protocol (FCS_TLSC_EXT.1)

PKGTLS11: FCS_TLSC_EXT.1.1

The product shall implement TLS 1.2 (RFC 5246) and [T***LS 1.1 (RFC 4346), 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_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***

] and also supports functionality for [

- ***mutual authentication,***
- ***session renegotiation***]. (TD0442 applied)

PKGTLS11: FCS_TLSC_EXT.1.2

The product shall verify that the presented identifier matches the reference identifier according to RFC 6125. (TD0499 applied)

PKGTLS11: FCS_TLSC_EXT.1.3

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

- ***with no exceptions,***

]. (TD0513 applied)

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

WLANCEP10: 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_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]. (TD0492 applied)

WLANCEP10: 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.

WLANCEP10: FCS_TLSC_EXT.1.3/WLAN
The TSF shall use X509 v3 certificates as specified in FIA_X509_EXT.1/WLAN. (TD0517 applied)

WLANCEP10: 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.

WLANCEP10: 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.

WLANCEP10: FCS_TLSC_EXT.1.6/WLAN
Removed by TD0492.

6.1.2.27 Extended: TLS Client Support for Mutual Authentication (FCS_TLSC_EXT.2)

PKGTLS11: FCS_TLSC_EXT.2.1
The product shall support mutual authentication using X.509v3 certificates.

WLANCEP10: 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)

6.1.2.28 Extended: TLS Client Support for Renegotiation (FCS_TLSC_EXT.4)

PKGTLS11: FCS_TLSC_EXT.4.1
The product shall support secure renegotiation through use of the “renegotiation_info” TLS extension in accordance with RFC 5746.

6.1.2.29 Extended: TLS Client Support for Supported Groups Extension (FCS_TLSC_EXT.5)

PKGTLS11: FCS_TLSC_EXT.5.1
The product shall present the Supported Groups Extension in the Client Hello with the supported groups [

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

].

6.1.3 User data protection (FDP)

6.1.3.1 Extended: Access Control for System Services (FDP_ACF_EXT.1)

MDFPP32: FDP_ACF_EXT.1.1

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

MDFPP32: 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 [***a common application developer (for sharing between applications), no one (for sharing between personal and enterprise profiles)***].

6.1.3.2 Extended: Access Control for System Resources (FDP_ACF_EXT.2)

MDFPP32: FDP_ACF_EXT.2.1

The TSF shall provide a separate [***address book, calendar, keychain***] 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 administrator (for address book), no one (for calendar, keychain)***].

6.1.3.3 Extended: Protected Data Encryption (FDP_DAR_EXT.1)

MDFPP32: FDP_DAR_EXT.1.1

Encryption shall cover all protected data.

MDFPP32: FDP_DAR_EXT.1.2

Encryption shall be performed using DEKs with AES in the [***XTS***] mode with key size [***256***] bits.

6.1.3.4 Extended: Sensitive Data Encryption (FDP_DAR_EXT.2)

MDFPP32: FDP_DAR_EXT.2.1

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

MDFPP32: 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.

MDFPP32: 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.

MDFPP32: 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.

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

MDFPP32: FDP_IFC_EXT.1.1

The TSF shall [

- ***provide an interface which allows a VPN client to protect all IP traffic using IPsec***

] with the exception of IP traffic required to establish the VPN connection. (TD0596 applied)

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

MDFPP32: FDP_PBA_EXT.1.1

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

6.1.3.7 Extended: User Data Storage (FDP_STG_EXT.1)

MDFPP32: FDP_STG_EXT.1.1

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

6.1.3.8 Extended: Inter-TSF user data transfer protection (FDP_UPC_EXT.1/APPS)

MDFPP32: FDP_UPC_EXT.1.1/APPS

The TSF 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 [

- ***no other protocol***

] to provide a protected communication channel between the non-TSF application and another IT product that is logically distinct from other communication channels, provides assured identification of its end points, protects channel data from disclosure, and detects modification of the channel data.

MDFPP32: FDP_UPC_EXT.1.2/APPS

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

6.1.4 Identification and authentication (FIA)

6.1.4.1 Authentication failure handling (FIA_AFL_EXT.1)

MDFPP32: FIA_AFL_EXT.1.1

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

MDFPP32: FIA_AFL_EXT.1.2

The TSF shall detect when a configurable positive integer within [**0 - 50**] of [***non-unique***] unsuccessful authentication attempts occur related to last successful authentication for each authentication mechanism.

MDFPP32: FIA_AFL_EXT.1.3

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

MDFPP32: 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.

MDFPP32: 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.

MDFPP32: 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.

6.1.4.2 Extended: Bluetooth User Authorization (FIA_BLT_EXT.1)

MODBT10: FIA_BLT_EXT.1.1

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

6.1.4.3 Extended: Bluetooth Mutual Authentication (FIA_BLT_EXT.2)

MODBT10: FIA_BLT_EXT.2.1

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

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

MODBT10: 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.

6.1.4.5 Extended: Secure Simple Pairing (FIA_BLT_EXT.4)

MODBT10: FIA_BLT_EXT.4.1

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

MODBT10: FIA_BLT_EXT.4.2

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

6.1.4.6 Extended: Trusted Bluetooth Device User Authorization (FIA_BLT_EXT.6)

MODBT10: 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**].

6.1.4.7 Extended: Untrusted Bluetooth Device User Authorization (FIA_BLT_EXT.7)

MODBT10: 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 profiles**].

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

MDFPP32: FIA_BMG_EXT.1.1/FINGERPRINT

The one-attempt BAF False Accept Rate (FAR) for [**fingerprint**] shall not exceed [**1:100,000**] with a one-attempt BAF False Reject Rate (FRR) not to exceed 1 in [**1:20**].

MDFPP32: FIA_BMG_EXT.1.2/FINGERPRINT

The overall System Authentication False Accept Rate (SAFAR) shall be no greater than 1 in [**1:5,000**] within a 1% margin.

MDFPP32: FIA_BMG_EXT.1.1/FACE

The one-attempt BAF False Accept Rate (FAR) for [**face**] shall not exceed [**1:100,000**] with a one-attempt BAF False Reject Rate (FRR) not to exceed 1 in [**1:20**].

MDFPP32: FIA_BMG_EXT.1.2/FACE

The overall System Authentication False Accept Rate (SAFAR) shall be no greater than 1 in [**1:5000**] within a 1% margin.

6.1.4.9 Extended: Port Access Entity Authentication (FIA_PAE_EXT.1)

WLANCEP10: FIA_PAE_EXT.1.1

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

6.1.4.10 Extended: Password Management (FIA_PMG_EXT.1)

MDFPP32: 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.

6.1.4.11 Extended: Authentication Throttling (FIA_TRT_EXT.1)

MDFPP32: 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.

6.1.4.12 Multiple Authentication Mechanisms (FIA_UAU.5)

MDFPP32: FIA_UAU.5.1

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

MDFPP32: FIA_UAU.5.2

The TSF shall authenticate any user's claimed identity according to the [

following rules:

- **unlock the user's Credential encrypted (CE files) and keystore keys**

To authenticate unlocking the device immediately after boot (first unlock after reboot):

- **User passwords are required after reboot to unlock the user's Credential encrypted (CE files) and keystore keys. Fingerprint and face authentication is disabled immediately after boot.**

To authenticate unlocking the device after device lock (not following a reboot):

- **The TOE verifies user credentials (password, fingerprint, or face) via the gatekeeper or fingerprint trusted application (running inside the Trusted Execution Environment, TEE), which compares the entered credential to a derived value or template.**

To change protected settings or issue certain commands:

- **The TOE requires password after a reboot, when changing settings (Screen lock, Fingerprint, Face unlock and Smart Lock settings), and when factory resetting."**

].

6.1.4.13 Re-Authentication (FIA_UAU.6)

MDFPP32: 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.

MDFPP32: 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**].

6.1.4.14 Protected authentication feedback (FIA_UAU.7)

MDFPP32: 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.

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

MDFPP32: 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 [***all software-based key storage***] at startup.

6.1.4.16 Extended: Timing of Authentication (FIA_UAU_EXT.2)

MDFPP32: FIA_UAU_EXT.2.1

The TSF shall allow [*[*

- Take screen shots (stored internally)

- Enter password to unlock

- Make/receive emergency calls

- Take pictures (stored internally) - unless the camera was disabled

- Turn the TOE off

- Restart the TOE

- Enable Airplane mode

- See notifications (note that some notifications identify actions, for example to view a screenshot; however, selecting those notifications highlights the password prompt and require the password to access that data)

- Configure sound, vibrate, or mute

- Set the volume (up and down) for ringtone

- Set the brightness for screen display

- Access notification widgets (without authentication):

o Wi-Fi toggle

o Mobile data toggle

o Bluetooth toggle

o Flashlight toggle

o Auto rotate toggle

o Natural tone display toggle

o 01 Ultra Vision toggle

o PC Connect toggle

o Dark mode toggle

o Eye comfort toggle

o Power saving toggle

o Personal hotspot toggle

o NFC toggle

o Switch data SIM toggle

] on behalf of the user to be performed before the user is authenticated.

MDFPP32: 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.

6.1.4.17 Extended: Validation of certificates (FIA_X509_EXT.1)

MDFPP32: 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***].
- 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 EKU field. [conditional]
    - Client certificate 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-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the EKU field. [conditional]

MDFPP32: 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.

6.1.4.18 Extended: X509 certificate authentication (FIA_X509_EXT.2)

MDFPP32: 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, [***no other protocol***], and [***code signing for system software updates***]. (TD0623 applied)

MDFPP32: FIA_X509_EXT.2.2

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

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

WLANCEP10: 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.

WLANCEP10: FIA_X509_EXT.2.2/WLAN

(removed as per TD0517)

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

MDFPP32: FIA_X509_EXT.3.1

The TSF shall provide a certificate validation service to applications.

MDFPP32: FIA_X509_EXT.3.2

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

6.1.5 Security management (FMT)

6.1.5.1 Extended: Management of security functions behavior (FMT_MOF_EXT.1)

MDFPP32: FMT_MOF_EXT.1.1

The TSF shall restrict the ability to perform the functions in column 3 of Table 4 to the user.

MDFPP32: FMT_MOF_EXT.1.2

The TSF shall restrict the ability to perform the functions in column 5 of Table 4 to the administrator when the device is enrolled and according to the administrator-configured policy.

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

MDFPP32: FMT_SMF_EXT.1.1

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

Table 4: Security Management Functions

Management Function **Status Markers:** **M – Mandatory** **O - Optional/Objective**	**Impl.**	**Users Only**	**Admin**	**Admin Only**
1. configure password policy: a. minimum password length b. minimum password complexity c. maximum password lifetime The administrator can configure the required password characteristics (minimum length, complexity, and lifetime) using the Android MDM APIs. Length: an integer value of characters Complexity: Unspecified, Something, Numeric, Alphabetic, Alphanumeric, Complex. Lifetime: an integer value of seconds (0 = no maximum).	M	-	M	M
2. configure session locking policy: a. screen-lock enabled/disabled b. screen lock timeout c. number of authentication failures	M	-	M	M

The administrator can configure the session locking policy using the Android MDM APIs. Screen lock timeout: an integer number of minutes before the TOE locks (0 = no lock timeout) Authentication failures: an integer number (-2,147,483,648 to 2,147,483,648 [negative integers and zero means no limit]).				
3. enable/disable the VPN protection: a. across device ***[d. no other method]*** Both users (using the TOE's settings UI) and administrator (using the TOE's MDM APIs) can configure a third-party VPN client and then enable the VPN client to protect traffic. The User can set up VPN protection, but if an admin enables VPN protection, the user cannot disable it.	M	O	O	O
4. enable/disable **[Bluetooth,** **NFC, Wi-Fi, cellular (GSM/WCDMA/LTE TDD/ LTE FDD/5G NR)]** The administrator can disable the Bluetooth using the TOE's MDM APIs. Once disabled, a user cannot enable the Bluetooth. The administrator cannot fully disable/restrict NFC, Wi-Fi or cellular voice capabilities. Radios are not used as part of the initialization of the device. Only when the radios are enabled, they are initialed in system service phase of TOE's boot sequence. The TOE's radios operate at frequencies of 13.56 MHz (NFC), 2.4 GHz (Bluetooth), 2.4/5 GHz (Wi-Fi), 850/900/1800/1900MHz (GSM), Bands 1/2/4/5/6/8/19 (WCDMA), Bands 34/38/39/40/41/42 (TDD-LTE), Bands 1/2/3/4/5/7/8/12/13/17/18/19/20/25/26/28/32/66 (LTE FDD), n1/n3/n5/n7/n8/n20/n28/n38/n40/n41/n77/n78/n79 (5G NR).	M M	 O	O	O
5. enable/disable **[microphone, camera]**: a. across device (microphone, camera), ***[b. on a per-app basis (microphone, camera)]*** An administrator can enable/disable the device's microphone and camera via an MDM API. Once the microphone or camera has been disabled, the user cannot re-enable it until the administrator enables it. In the user's settings, a user can view a permission by type (i.e. camera, microphone). The user can access this by going to "Settings" -> "App Permissions" -> Selecting the permission and revoking any applications.	M M	-	O	O
6. transition to the locked state Both users (using the TOE's settings UI) and administrators (using the TOE's MDM APIs) can transition the TOE into a locked state.	M	-	M	-
7. TSF wipe of protected data Both users (using the TOE's settings UI) and administrators (using the TOE's MDM APIs) can force the TOE to perform a full wipe (factory reset) of data.	M	-	M	-
8. configure application installation policy by: ***[a. restricting the sources of applications*** ***c. denying installation of applications]*** The administrator using the TOE's MDM APIs can configure the TOE so that applications cannot be installed and can also block the use of the Google Market Place.	M	-	M	M
9. import keys/secrets into the secure key storage Both users (using the TOE's settings UI) and administrators (using the TOE's MDM APIs) can import secret keys into the secure key storage.	M	O	O	-

10. destroy imported keys/secrets and ***[no other keys/secrets]*** in the secure key storage Both users and administrators (using the TOE's MDM APIs) can destroy secret keys in the secure key storage.	M	O	O	-
11. import X.509v3 certificates into the Trust Anchor Database Both users (using the TOE's settings UI) and administrators (using the TOE's MDM APIs) can import X.509v3 certificates into the Trust Anchor Database.	M	-	M	O
12. remove imported X.509v3 certificates and ***[no other certificates]*** in the Trust Anchor Database Both users (using the TOE's settings UI) and administrators (using the TOE's MDM APIs) can remove imported X.509v3 certificates from the Trust Anchor Database as well as disable any of the TOE's default Root CA certificates (in the latter case, the CA certificate still resides in the TOE's read-only system partition; however, the TOE will treat that Root CA certificate and any certificate chaining to it as untrusted).	M	O	O	-
13. enroll the TOE in management TOE users can enroll the TOE in management according to the instructions specific to a given MDM. Presumably any enrollment would involve at least some user functions (e.g., install an MDM agent application) on the TOE prior to enrollment.	M	O	O	O
14. remove applications Both users (using the TOE's settings UI) and administrators (using the TOE's MDM APIs) can uninstall user and administrator installed applications on the TOE.	M	-	M	O
15. update system software Users can check for updates and cause the device to update if an update is available. An administrator can use MDM APIs to query the version of the TOE and query the installed applications and an MDM agent on the TOE could issue pop-ups, initiate updates, block communication, etc. until any necessary updates are completed.	M	-	M	O
16. install applications Both users and administrators (using the TOE's MDM APIs) can install applications on the TOE.	M	-	M	O
17. remove Enterprise applications An administrator (using the TOE's MDM APIs) can uninstall Enterprise installed applications on the TOE.	M	-	M	-
18. enable/disable display notification in the locked state of: ***[f. all notifications]*** Notifications can be configured to display in the following formats: Users & administrators: show all notification content Users: hide sensitive content Users & administrators: hide notifications entirely If the administrator sets any of the above settings, the user cannot change it.	M	O	O	O
19. enable data-at rest protection The TOE always encrypts its user data storage.	M	O	O	O
20. enable removable media's data-at-rest protection.	M	O	O	O
21. enable/disable location services: a. across device ***[d. no other method]*** The administrator (using the TOE's MDM APIs) can enable or disable location services. An additional MDM API can prohibit TOE users ability to enable and disable location services.	M	O	O	O
22. Enable/disable the use of ***[Biometric Authentication Factor]*** The Biometric Authentication Factor is always enabled and could not be disabled by user.	M	O	O	O

23. configure whether to allow/disallow establishment of a trusted channel if the peer/server certificate is deemed invalid **The TOE will not connect to the Wi-Fi if peer/server certificate is deemed invalid, and there is no configuration interface to override this.** **For TLS connection, the API provides "trustmanager" class as the configuration interface. If user does not define this "trustmanager" class, TOE will not establish the trusted channel.**	M	O	O	O
24. enable/disable all data signaling over [**assignment: list of externally accessible hardware ports**]	O	O	O	O
25. enable/disable ***[Wi-Fi hotspot, USB tethering, and Bluetooth tethering]*** The administrator (using the TOE's MDM APIs) can enable/disable all tethering methods (i.e. all or none disabled). The TOE acts as a server (acting as an access point, a USB Ethernet adapter, and as a Bluetooth Ethernet adapter respectively) in order to share its network connection with another device.	O	O	O	O
26. enable/disable developer modes The administrator (using the TOE's MDM APIs) can disable Developer Mode. Unless disabled by the administrator, TOE users can enable and disable Developer Mode.	O	O	O	O
27. enable/disable bypass of local user authentication N/A – It is not possible to bypass local user auth for this TOE	O	O	O	O
28. wipe Enterprise data An administrator can remove Enterprise applications and their data.	O	O	O	-
29. approve ***[import, removal]*** by applications of X.509v3 certificates in the Trust Anchor Database	O	O	O	O
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	O	O	O	O
31. enable/disable the cellular protocols used to connect to cellular network base stations	O	O	O	O
32. read audit logs kept by the TSF The administrator could read logs that kept by the TSF using "TestDPC -> Request security logs", and user could read logs via "LogKit" tool.	O	O	O	-
33. configure ***[selection: certificate, public-key]*** used to validate digital signature on applications	O	O	O	O
34. approve exceptions for shared use of keys/secrets by multiple applications	O	O	O	O
35. approve exceptions for destruction of keys/secrets by applications that did not import the key/secret	O	O	O	O
36. configure the unlock banner	O	-	O	O
37. configure the auditable items	O	-	O	O
38. retrieve TSF-software integrity verification values	O	O	O	O
39. enable/disable *[* ***a. USB mass storage mode,*** *]*	O	O	O	O
40. enable/disable backup to ***[all applications]*** to ***[remote system]***	O	O	O	O
41. enable/disable ***[*** ***a. Hotspot functionality authenticated by [pre-shared key],*** ***b. USB tethering authenticated by [no authentication]]*** The administrator (using the TOE's MDM APIs) can disable the Wi-Fi hotspot and USB tethering. Unless disabled by the administrator, TOE users can configure the Wi-Fi hotspot with a pre-shared key and can configure USB tethering (with no authentication).	O	O	O	O
42. approve exceptions for sharing data between ***[groups of application]***	O	O	O	O

43. place applications into application process groups based on [**assignment: enterprise configuration settings**]	O	O	O	O
44. Unenroll the TOE from management	O	O	O	O
45. enable/disable the Always On VPN protection: a. across device [**selection:** **b. on a per-app basis,** **c. on a per-group of applications processes basis,** **d. no other method**] Administrator could config Always On VPN protection by using DevicePolicyManage`s API setAlwaysOnVpnPackage. Or the user can access this by going to "Settings" -> "Connection & sharing" -> "VPN"	O	O	O	O
46. Revoke Biometric template	O	O	O	O
47. [**assignment: list of other management functions to be provided by the TSF**]	O	O	O	O

MODBT10: FMT_SMF_EXT.1.1/BT

The TSF shall be capable of performing the functions of Table 5 Bluetooth Management Functions:

Table 5: Bluetooth Management Functions

Management Function **Status Markers:** **M – Mandatory** **O - Optional/Objective**	**Impl.**	**Users Only**	**Admin**	**Admin Only**
BT-1. Configure the Bluetooth trusted channel. • Disable/enable the Discoverable (for BR/EDR) and Advertising (for LE) modes;	M	O	O	O
BT-2. Change the Bluetooth device name (separately for BR/EDR and LE);	O	O	O	O
BT-3. Provide separate controls for turning the BR/EDR and LE radios on and off;	-	-	-	-
BT-4. Allow/disallow the following additional wireless technologies to be used with Bluetooth: [***selection: Wi-Fi, NFC, [assignment: other wireless technologies]***];	-	-	-	-
BT-5. Configure allowable methods of Out of Band pairing (for BR/EDR and LE);	-	-	-	-
BT-6. Disable/enable the Discoverable (for BR/EDR) and Advertising (for LE) modes separately;	-	-	-	-
BT-7. Disable/enable the Connectable mode (for BR/EDR and LE);	-	-	-	-
BT-8. Disable/enable the Bluetooth [assignment: list of Bluetooth service and/or profiles available on the OS (for BR/EDR and LE)].	-	-	-	-
BT-9. Specify minimum level of security for each pairing (for BR/EDR and LE);	-	-	-	-

WLANCEP10: FMT_SMF_EXT.1.1/WLAN

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

- ***configure security policy for each wireless network:***
  - ***[specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s)]***
  - security type
  - authentication protocol
  - client credentials to be used for authentication;
- specify wireless networks (SSIDs) to which the TSF may connect;

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

MDFPP32: 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***] upon unenrollment and [***factory reset***].

6.1.6 Protection of the TSF (FPT)

6.1.6.1 Extended: Application Address Space Layout Randomization (FPT_AEX_EXT.1)

MDFPP32: FPT_AEX_EXT.1.1

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

MDFPP32: FPT_AEX_EXT.1.2

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

6.1.6.2 Extended: Memory Page Permissions (FPT_AEX_EXT.2)

MDFPP32: FPT_AEX_EXT.2.1

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

6.1.6.3 Extended: Stack Overflow Protection (FPT_AEX_EXT.3)

MDFPP32: 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.

6.1.6.4 Extended: Domain Isolation (FPT_AEX_EXT.4)

MDFPP32: FPT_AEX_EXT.4.1

The TSF shall protect itself from modification by untrusted subjects.

MDFPP32: FPT_AEX_EXT.4.2

The TSF shall enforce isolation of address space between applications.

6.1.6.5 Extended: JTAG Disablement (FPT_JTA_EXT.1)

MDFPP32: FPT_JTA_EXT.1.1

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

6.1.6.6 Extended: Key Storage (FPT_KST_EXT.1)

MDFPP32: FPT_KST_EXT.1.1

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

6.1.6.7 Extended: No Key Transmission (FPT_KST_EXT.2)

MDFPP32: FPT_KST_EXT.2.1

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

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

MDFPP32: FPT_KST_EXT.3.1

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

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

MDFPP32: FPT_NOT_EXT.1.1

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

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

6.1.6.10 Reliable time stamps (FPT_STM.1)

MDFPP32: FPT_STM.1.1

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

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

MDFPP32: 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.

6.1.6.12 Extended: TSF Cryptographic Functionality Testing (Wireless LAN) (FPT_TST_EXT.1/WLAN)

WLANCEP10: FPT_TST_EXT.1.1/WLAN

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

WLANCEP10: FPT_TST_EXT.1.2/WLAN

The [***TOE***] 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.

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

MDFPP32: 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***].

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

MDFPP32: FPT_TUD_EXT.1.1

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

MDFPP32: 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.

MDFPP32: FPT_TUD_EXT.1.3

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

6.1.6.15 Extended: TSF Update Verification (FPT_TUD_EXT.2)

MDFPP32: FPT_TUD_EXT.2.1

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

MDFPP32: FPT_TUD_EXT.2.2

The TSF shall [***never update***] the TSF boot integrity [***hash***].

MDFPP32: 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***].

6.1.6.16 Extended: Application Signing (FPT_TUD_EXT.3)

MDFPP32: FPT_TUD_EXT.3.1

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

6.1.7 TOE access (FTA)

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

MDFPP32: FTA_SSL_EXT.1.1

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

MDFPP32: FTA_SSL_EXT.1.2

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

MDFPP32: 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**].

6.1.7.2 Extended: Wireless Network Access (FTA_WSE_EXT.1)

WLANCEP10: 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.

6.1.8 Trusted path/channels (FTP)

6.1.8.1 Extended: Bluetooth Encryption (FTP_BLT_EXT.1)

MODBT10: 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]***.

MODBT10: FTP_BLT_EXT.1.2

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

6.1.8.2 Extended: Persistence of Bluetooth Encryption (FTP_BLT_EXT.2)

MODBT10: FTP_BLT_EXT.2.1

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

6.1.8.3 Extended: Bluetooth Encryption Parameters (FTP_BLT_EXT.3)

MODBT10: 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.

MODBT10: 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.

6.1.8.4 Extended: Trusted channel Communication (FTP_ITC_EXT.1)

MDFPP32: FTP_ITC_EXT.1.1

The TSF shall use

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

and *[*

- ***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.

MDFPP32: FTP_ITC_EXT.1.2

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

MDFPP32: 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***].

6.1.8.5 Extended: Trusted Channel Communication (Wireless LAN)(FTP_ITC_EXT.1/WLAN)

WLANCEP10: FTP_ITC_EXT.1.1/WLAN

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

WLANCEP10: FTP_ITC_EXT.1.2/WLAN

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

6.2 TOE Security Assurance Requirements

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

Table 6: Security Assurance Requirements

Requirement Class	Requirement Component
Security Target (ASE)	ASE_CCL.1: Conformance claims
	ASE_ECD.1: Extended components definition
	ASE_INT.1: ST introduction
	ASE_OBJ.1: Security objectives for the operational environment
	ASE_REQ.1: Stated security requirements
	ASE_SPD.1: Security Problem Definition
	ASE_TSS.1: TOE summary specification
ADV: Development	ADV_FSP.1: Basic functional specification
AGD: Guidance documents	AGD_OPE.1: Operational user guidance
	AGD_PRE.1: Preparative procedures
ALC: Life-cycle support	ALC_CMC.1: Labelling of the TOE
	ALC_CMS.1: TOE CM coverage
	ALC_TSU_EXT.1: Timely Security Updates
ATE: Tests	ATE_IND.1: Independent Testing – Sample
AVA: Vulnerability assessment	AVA_VAN.1: Vulnerability survey

6.2.1 Development (ADV)

6.2.1.1 Basic functional specification (ADV_FSP.1)

ADV_FSP.1.1d

The developer shall provide a functional specification.

ADV_FSP.1.2d

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

ADV_FSP.1.3c

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

ADV_FSP.1.4c

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

ADV_FSP.1.5c

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

ADV_FSP.1.6c

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

ADV_FSP.1.7e

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

ADV_FSP.1.8e

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

6.2.2 Guidance documents (AGD)

6.2.2.1 Operational user guidance (AGD_OPE.1)

AGD_OPE.1.1d

The developer shall provide operational user guidance.

AGD_OPE.1.2c

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.1.3c

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.1.4c

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.1.5c

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.1.6c

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

AGD_OPE.1.7c

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.1.8c

The operational user guidance shall be clear and reasonable.

AGD_OPE.1.9e

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

6.2.2.2 Preparative procedures (AGD_PRE.1)

AGD_PRE.1.1d

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

AGD_PRE.1.2c

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.1.3c

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.1.4e

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

AGD_PRE.1.5e

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

6.2.3 Life-cycle support (ALC)

6.2.3.1 Labelling of the TOE (ALC_CMC.1)

ALC_CMC.1.1d

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

ALC_CMC.1.2c

The TOE shall be labeled with a unique reference.

ALC_CMC.1.3e

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

6.2.3.2 TOE CM coverage (ALC_CMS.1)

ALC_CMS.1.1d

The developer shall provide a configuration list for the TOE.

ALC_CMS.1.2c

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

ALC_CMS.1.3c

The configuration list shall uniquely identify the configuration items.

ALC_CMS.1.4e

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

6.2.3.3 Timely Security Updates (ALC_TSU_EXT.1)

ALC_TSU_EXT.1.1d

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

ALC_TSU_EXT.1.2c

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

ALC_TSU_EXT.1.3c

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.1.4c

The description shall include the mechanisms publicly available for reporting security issues pertaining to the TOE.

ALC_TSU_EXT.1.5c

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.1.6e

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

6.2.4 Tests (ATE)

6.2.4.1 Independent testing - conformance (ATE_IND.1)

ATE_IND.1.1d

The developer shall provide the TOE for testing.

ATE_IND.1.2c

The TOE shall be suitable for testing.

ATE_IND.1.3e

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

ATE_IND.1.4e

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

6.2.5 Vulnerability assessment (AVA)

6.2.5.1 Vulnerability survey (AVA_VAN.1)

AVA_VAN.1.1d

The developer shall provide the TOE for testing.

AVA_VAN.1.2c

The TOE shall be suitable for testing.

AVA_VAN.1.3e

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

AVA_VAN.1.4e

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

AVA_VAN.1.5e

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.

7 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

7.1 Security Audit

MDFPP32: FAU_GEN.1:

The TOE uses different forms of logs to meet all the required management logging events specified in Table 1 of the MDFPP32 and all mandatory auditable events specified by the MODBT10 and WLANCEP10:

1. Security Logs
2. Logcat Logs

Each of the above logging methods are described below.

- Security Logs: A table that depicts the list of all auditable events (for MDFPP32 and all mandatory auditable events specified by the MODBT10 and WLANCEP10) can be found here: https://developer.android.com/reference/android/app/admin/SecurityLog. Additionally, the following link provides the additional information that can be grabbed when an MDM requests a copy of the logs: https://developer.android.com/reference/android/app/admin/SecurityLog.SecurityEvent. Each log contains a keyword or phrase describing the event, the date and time of the event, and further event- specific values that provide success, failure, and other information relevant to the event. These logs can be read by an administrator via an MDM agent.
- Logcat Logs: Similar to Security Logs, Logcat Logs contain date, time, and further even-specific values within the logs. In addition, Logcat Logs provide a value that maps to a user ID to identify which user caused the event that generated the log. Finally, Logcat Logs are descriptive and do not require the administrator to know the template of the log to understand its values. Logcat Logs cannot be exported but can be viewed by an administrator via an MDM agent.

Both types of logs, when full, wrap around and overwrite the oldest log (as the start of the buffer).

Each log entry is formatted as: "<Keyword> (<Date><Timestamp>): <message>", where "Keyword" indicates what type of the log it is, "<Date><Timestamp>" indicates the exact time the entry is recorded, and "<message>" shows the content of the log entry.

The following table enumerates the events that the TOE audits.

Table 7: Audit Event

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	None.	
FAU_GEN.1/WLAN	None.	

FAU_STG.1	None.	
FAU_STG.4	None.	
FCS_CKM_EXT.1	[***None***].	No additional information.
FCS_CKM_EXT.2	None.	
FCS_CKM_EXT.3	None.	
FCS_CKM_EXT.4	None.	
FCS_CKM_EXT.5	[***None***].	No additional information.
FCS_CKM_EXT.6	None.	
FCS_CKM_EXT.8	None.	
FCS_CKM.1	[***None***].	No additional information.
FCS_CKM.1/WLAN	None.	
FCS_CKM.2/UNLOCKED	None.	
FCS_CKM.2/LOCKED	None.	
FCS_CKM.2/WLAN	None.	
FCS_COP.1/ENCRYPT	None.	
FCS_COP.1/HASH	None.	
FCS_COP.1/SIGN	None.	
FCS_COP.1/KEYHMAC	None.	
FCS_COP.1/CONDITION	None.	
FCS_IV_EXT.1	None.	
FCS_SRV_EXT.1	None.	
FCS_STG_EXT.1	Import or destruction of key.	Identity of key. Role and identity of requestor.
	[***No other events***]	
FCS_STG_EXT.2	None.	
FCS_STG_EXT.3	Failure to verify integrity of stored key.	Identity of key being verified.

FCS_TLSC_EXT.1/WLAN	Failure to establish an EAP-TLS session.	Reason for failure.
	Establishment/termination of an EAP-TLS session.	Non-TOE endpoint of connection.
FDP_DAR_EXT.1	[***None***].	No additional information.
FDP_DAR_EXT.2	Failure to encrypt/decrypt data.	No additional information
FDP_IFC_EXT.1	None.	
FDP_STG_EXT.1	Addition or removal of certificate from Trust Anchor Database.	Subject name of certificate.
FIA_BLT_EXT.1	Failed user authorization of Bluetooth device.	User authorization decision (e.g., user rejected connection, incorrect pin entry).
	Failed user authorization for local Bluetooth Service.	Bluetooth address and name of device. Bluetooth profile. Identity of local service with [***service ID***].
FIA_BLT_EXT.2	Initiation of Bluetooth connection.	Bluetooth address and name of device.
	Failure of Bluetooth connection.	Reason for failure.
FIA_BLT_EXT.4	None.	
FIA_BLT_EXT.6	None.	
FIA_BLT_EXT.7	None.	
FIA_PAE_EXT.1	None.	
FIA_PMG_EXT.1	None.	
FIA_TRT_EXT.1	None.	
FIA_UAU_EXT.1	None.	
FIA_UAU.5	None.	
FIA_UAU.7	None.	
FIA_X509_EXT.1	Failure to validate X.509v3 certificate.	Reason for failure of validation.
FIA_X509_EXT.2/WLAN	None.	

FMT_MOF_EXT.1	None.	
FMT_SMF_EXT.1/WLAN	None.	
FPT_AEX_EXT.1	None.	
FPT_AEX_EXT.2	None.	
FPT_AEX_EXT.3	None.	
FPT_JTA_EXT.1	None.	
FPT_KST_EXT.1	None.	
FPT_KST_EXT.2	None.	
FPT_KST_EXT.3	None.	
FPT_NOT_EXT.1	[***None***].	[***No additional information***].
FPT_STM.1	None.	
FPT_TST_EXT.1	Initiation of self-test.	
	Failure of self-test.	[***None***]
FPT_TST_EXT.1/WLAN **(note: can be performed by TOE or TOE platform)**	Execution of this set of TSF self-tests. [***none***].	[***No additional information***] (Done as part of FPT_TST_EXT.1)
FPT_TST_EXT.2/PREKERNEL	Start-up of TOE.	No additional information.
	[***None***]	[***No additional information***]
FPT_TUD_EXT.1	None.	
FTA_SSL_EXT.1	None.	
FTA_WSE_EXT.1	All attempts to connect to access points.	Identity of access point being connected to as well as success and failures (including reason for failure).
FTP_BLT_EXT.1	None.	
FTP_BLT_EXT.2	None.	
FTP_BLT_EXT.3/BR	None.	
FTP_BLT_EXT.3/LE **(if claimed)**	None.	

FTP_ITC_EXT.1/WLAN	All attempts to establish a trusted channel. (TD0194 applied)	Identification of the non-TOE endpoint of the channel.

MDFPP32: FAU_STG.1: For security logs, the TOE stores all audit records in memory, making it only accessible to the logd daemon, and only applications that are set with the "device owners" permission by MDM can call the MDM API to retrieve a copy of the logs. Additionally, only new logs can be added. There is no designated method allowing for the deletion or modification of logs already present in memory, but reading the security logs clears the buffer at the time of the read.

The TOE stores Logcat Logs in memory and only allows access by an administrator via an MDM Agent. The TOE prevents deleted of these logs by any method other than USB debugging (and enabling USB Debugging takes the phone out of the evaluated configuration).

MDFPP32: FAU_STG.4: The security logs and logcat logs are stored in memory in a circular log buffer of 4096KB/256KB, respectively. Logcat logs alone have a configurable size, able to be set by an MDM API. There is no limit to the size that the Logcat log buffer can be configured to and it is limited to the size of the system's memory. Each log system retains its own circular buffer. Once either the log is full, it begins overwriting the oldest message in its respective buffer and continues overwriting the oldest message with each new auditable event. These logs persist until they are either overwritten or the device is restarted.

7.2 Cryptographic Support

The TOE implements cryptographic algorithms in accordance with the following NIST standards.

Table 8: Supported Cryptographic Algorithms

Algorithm	NIST Standard	SFR Reference
AES CBC, CCMP, KW, KWP, GCM, CCM, XTS	FIPS 197, SP 800-38A/C/D/E/F	FCS_COP.1/ENCRYPT
SHA-1, SHA-256, SHA384, SHA512	FIPS 180-4	FCS_COP.1/HASH
RSA, ECDSA	FIPS SP 186-4	FCS_COP.1/SIGN FCS_CKM.1
HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	FIPS 198-1 & 180-4	FCS_COP.1/KEYHMAC
RSA, ECDSA	SP800-56A/B	FCS_CKM.2/UNLOCKED FCS_CKM.2/LOCKED
DRBG	FIPS SP 800-90A	FCS_RBG_EXT.1

The Cryptographic support function in the TOE is designed to fulfill the following security functional requirements:

MDFPP32: FCS_CKM.1: The TOE provides generation of asymmetric keys including:

Table 9: Asymmetric Key Generation

Algorithm	Key Sizes / Curves	Usage

RSA, FIPS 186-4	2048/3072/4096	API / Application & Sensitive Data Protection (DAR.2)
ECDSA, FIPS 186-4	P-256/384	API / Application
ECDHE (not domain parameters)	P-256/384	TLS Key exchange (WPA2 with EAP-TLS & HTTPS)

All the cryptographic algorithms that provided by the AP have NIST CAVP certificates, which are listed in the tables of FCS_COP.1, other algorithms provided by BoringSSL have also been tested with the NIST ACVTS system.

TOE provides key generation APIs to mobile applications to allow them to generate RSA/ECDSA key pairs. The TOE generates only ECDH key pairs (as BoringSSL does not support DH/DHE cipher suites) and does not generate domain parameters (curves) for use in TLS Key Exchange.

The TOE will provide a library for application developers to use for Sensitive Data Protection (SDP). This library (class) generates asymmetric RSA keys for use to encrypt and decrypt data that comes to the device while in a locked state. Any data received for a specified application (that opts into SDP via this library and protected by BE mechanism), is encrypted using the public key and stored until the device is unlocked. The public key stays in memory no matter the state of the device (locked or unlocked). However, when the device is locked, the private key is evicted from memory and unavailable for use until the device is unlocked. Upon unlock, the private key is re-decrypted and used to decrypt data received and encrypted while locked.

WLANCEP10: 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 PRF384 and PRF704 for WPA2 derivation of 128-bit and 256-bit AES Temporal Keys (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 and AES-256 GCMP encryption modes. 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.

MDFPP32: FCS_CKM.2/UNLOCKED: The TOE performs key establishment as part of EAP-TLS and TLS session establishment. Table 9: Asymmetric Key Generation Asymmetric Key Generation enumerates the TOE'S supported key establishment implementations (RSA/ECDH for TLS/EAP-TLS). The TOE acts as a TLS client, the TOE only performs 800-56B encryption when participating in TLS_RSA_* based TLS handshakes. Thus, the TOE does not perform 800-56B decryption.

MDFPP32: FCS_CKM.2/LOCKED: The TOE provides an SDP library for applications that uses a hybrid crypto scheme based on 2048-bit RSA based key establishment. Applications can utilize this library to implement SDP that encrypts incoming data received while the phone is locked in a manner compliant with this requirement.

WLANCEP10: FCS_CKM.2/WLAN: The TOE adheres to RFC 3394 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).

MDFPP32: FCS_CKM_EXT.1: The TOE includes a Root Encryption Key (REK) stored in a 256-bit fuse bank within the application processor. The TOE generates the REK/fuse value during manufacturing using its hardware DRBG. The application processor protects the REK by preventing any direct observation of the value and prohibiting any ability to modify or update the value. The application processor loads the fuse value into an internal hardware crypto register and the Trusted Execution Environment (TEE) provides trusted applications the ability to derive KEKs from the REK (using an SP 800-108 KDF to combine the REK with a salt). Additionally, the when the REK is loaded, the fuses for the REK become locked, preventing any further changing or loading of the REK value. The TEE does not allow trusted applications to use the REK for encryption or decryption, only the ability to derive a KEK from the REK. The TOE includes a TEE application that calls into the TEE in order to derive a KEK from the 256-bit REK/fuse value and then only permits use of the derived KEK for encryption and decryption as part of the TOE key hierarchy.

MDFPP32: FCS_CKM_EXT.2: The TOE utilizes its approved RBGs to generate DEKs. When generating AES keys for itself (for example, the TOE'S sensitive data encryption keys or for the Secure Key Storage), TEE will call qsee_prng_getdata() API to generate a 256-bit AES key. The TOE utilizes the RAND_bytes() API call from its BoringSSL AES-256 CTR_DRBG to generate a 256-bit AES key. The TOE also utilizes that same DRBG when servicing API requests from mobile applications wishing to generate AES keys (either 128 or 256-bit).

When generating keys, DRBG is fed in with 384 bits length of entropy input, based on the entropy analysis, this 384 bits stream contains more than 256-bits entropy which is the maximum length of the generated keys, which could ensure that the TOE generates DEKs with sufficient entropy, the generated key cannot be recovered with less work than a full exhaustive search of the key space.

MDFPP32: FCS_CKM_EXT.3: The TOE takes the user-entered password and conditions/stretches this value before combining the factor with other KEK. The TOE generates all non-derived KEKs using the RAND_bytes() API call from its BoringSSL AES-256 CTR_DRBG to ensure a full 112/256-bits of strength for asymmetric/symmetric keys, respectively. And the TOE combines KEKs by encrypting one KEK with the other so as to preserve entropy.

MDFPP32: FCS_CKM_EXT.4: The TOE clears sensitive cryptographic material (plaintext keys, authentication data, other security parameters) from memory when no longer needed or when transitioning to the device's locked state (in the case of the Sensitive Data Protection keys). Public keys (such as the one used for Sensitive Data Protection) can remain in memory when the phone is locked, but all crypto-related private keys are evicted from memory upon device lock. No plaintext cryptographic material resides in the TOE'S Flash as the TOE encrypts all keys stored in Flash. When performing a full wipe of protected data, the TOE cryptographically erases the protected data by clearing the Data-At-Rest DEK. Because the TOE'S keystore resides within the user data partition, the TOE effectively cryptographically erases those keys when clearing the Data-At-Rest DEK. In turn, the TOE clears the Data-At-Rest DEK and Secure Key Storage KEK through a secure direct overwrite (BLKSECDISCARD ioctl) of the wear-leveled Flash memory containing the key followed by a read-verify. Document "Key Hierarchy Figure" further explains how each type of plaintext key material is generated, stored and cleared.

MDFPP32: FCS_CKM_EXT.5: The TOE stores all protected data in encrypted form within the user data partition (either protected data or sensitive data). Upon request, the TOE cryptographically erases the Data-At-Rest DEK protecting the user data partition and the Sensitive Data Protection KEKs protecting sensitive data files in the user data partition by overwrite the DEK, KEK, and then clears those keys from memory, reformats the partition, and then reboots. The TOE's clearing of the keys follows the requirements of FCS_CKM_EXT.4. Document "Key Hierarchy Figure" further explains how each type of plaintext key material is generated, stored and cleared.

MDFPP32: FCS_CKM_EXT.6: The TOE generates salt nonces (which are just salt values used in WPA2) using its /dev/random

Table 10: Salt Generation

Salt value and size	Used RBG	Storage
User password salt (128-bit)	BoringSSL's AES-256 CTR_DRBG	Flash filesystem
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)	BoringSSL's AES-256 CTR_DRBG	N/A (ephemeral)
WPA2 4-way handshake supplicant nonce (SNonce)	BoringSSL's AES-256 CTR_DRBG	N/A (ephemeral)

MODBT10: FCS_CKM_EXT.8: The TSF generates public/private ECDH key pairs for Bluetooth every paring to protect the data that are exchanged between TOE and the paired device.

MDFPP32: FCS_COP.1: The TOE implements cryptographic algorithms in accordance with the following NIST standards and has received the following CAVP algorithm certificates.

The TOE's BoringSSL library (ae2bb641735447496bed334c495e4868b981fe32) provides the following cryptographic algorithms:

Table 11: Cryptographic Algorithms Provided by BoringSSL

SFR	Algorithm	Standard
FCS_CKM.1 (Key Gen)	RSA/ECDSA	FIPS186-4
FCS_CKM.2 (Key Establishment)	ECDSA-based Key exchange, RSA-based Key exchange	SP800-56A SP800-56B
FCS_COP.1/ENCRYPT (AES)	AES CBC, CCMP, KW, KWP, GCM, CCM, XTS	FIPS 197, SP800-38A/C/D/E/F
FCS_COP.1/HASH (Hash)	SHA-1, SHA-256/384/512	FIPS 180-4
FCS_COP.1/SIGN (Sign/Verify)	RSA/ECDSA Signature generation and verification	FIPS 186-4
FCS_COP.1/KEYHMAC (Keyed Hash)	HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384. HMAC-SHA-512	FIPS198-1
FCS_RBG_EXT.1 (Random)	CTR_DRBG Hash_DRBG	SP800-90A

The TOE's Wi-Fi chipset (WCN6856) provides an AES-CCMP implementation, and the TOE's application processor provides the following cryptographic algorithms:

Table 12: Cryptographic Algorithms Provided by Application Processor

SFR	Module	Algorithm	Standard	Certificates
FCS_COP.1/ ENCRYPT (AES)	Qualcomm(R) Crypto Engine Core	AES 128/256 CBC	FIPS 197, SP 800-38A	A2045
FCS_COP.1/ ENCRYPT (AES)	Qualcomm (R) Inline Crypto Engine (UFS) Encryption (ENCRYPT) Qualcomm (R) Inline Crypto Engine (UFS) Decryption (DECRYPT)	AES 128/256 XTS	FIPS 197, SP 800-38E	A2116 A2117
FCS_COP.1/ HASH (Hash)	Qualcomm(R) Crypto Engine Core	SHA 1/256 Hashing	FIPS 180-4	A2045
FCS_COP.1/ KEYHMAC (Keyed Hash)	Qualcomm(R) Crypto Engine Core	HMAC-SHA-1/256	FIPS 198-1, FIPS 180-4	A2045

FCS_RBG_EXT.1 (Random) (DRBG)	Qualcomm(R) Pseudo Random Number Generator (DRBG)	DRBG Bit Generation	SP 800-90A (Hash-256)	A2064 A2065
FCS_CKM_EXT.3	Qualcomm(R) Trusted Execution Environment Kernel Software Crypto APIs (KERNEL)	KBKDF	SP 800-108	A2122

The TOE's BoringSSL library supports the TOE's cryptographic Android Runtime (ART) methods (through Android's conscrypt JNI provider) afforded to mobile applications and also supports Android user-space processes and daemons (e.g., wpa_supplicant). The TOE's Application Processor provides hardware accelerated cryptography utilized in Data-At-Rest (DAR) encryption of the user data partition.

The TOE stretches the user's password to create a password derived key. The TOE stretching function uses a series of steps to increase the memory required for key derivation (thus thwarting GPU-acceleration, off-line brute force, and precomputed dictionary attacks) and ensure proper conditioning and stretching of the user's password.

The TOE conditions the user's password using two iterations of PBKDFv2 with HMAC-SHA-256 in addition to some ROMix operations in an algorithm named scrypt. Scrypt consists of one iteration of PBKDFv2, followed by a series of ROMix operations, and finished with a final iteration of PBKDFv2. The ROMix operations increase the memory required for key derivation, thus thwarting GPU-acceleration (which can greatly decrease the time needed to brute force PBKDFv2 alone).

The following scrypt diagram shows how the password and salt are used with PBKDF v2 and ROMix to fulfil the requirements for password conditioning.

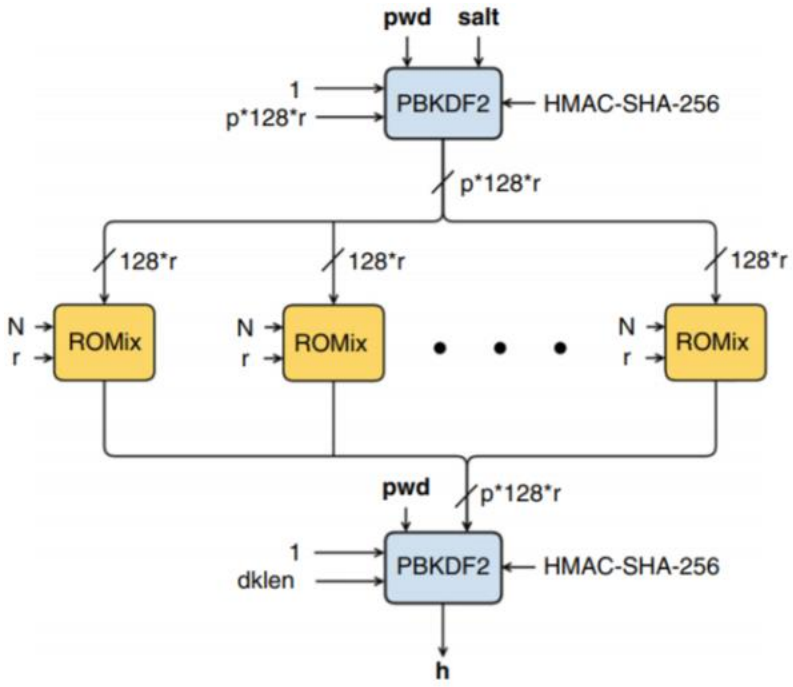

Figure 1: Password conditioning diagram

The resulting derived key from this operation is used to decrypt the FBE DEK (document "Key Hierarchy Figure" further explains on how this key is used in FBE) and to decrypt the Software based protected key.

The TOE uses HMAC as part of the TLS ciphersuites and makes HMAC functionality available to mobile applications. For TLS, the TOE uses HMAC using SHA-1 (with a 160-bit key) to generate a 160-bit MAC, SHA-256 (with a 256-bit key) to generate a 256-bit MAC, SHA-384 (with a 384-bit key) to generate a 384-bit MAC. For mobile applications, the TOE

provides all of the previous HMACs as well as SHA-512 (with a 512-bit key) to generate a 512-bit MAC. FIPS 198-1 & 180-4 dictate the block size used, and they specify block sizes/output MAC lengths of 512/160, 512/256, 1024/384, and 1024/512-bits for HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 respectively.

The TOE uses SHA together with algorithms in FCS_COP.1/SIGN for digital signature generation and verification, SHA-1 is used to generate 160-bit message digest, SHA-256, SHA-384 and SHA512 are used to generate 256-bit, 384-bit and 512-bit message digests respectively. The TOE also uses SHA with algorithms in FCS_COP.1/KEYHMAC and FCS_COP.1/CONDITION for the hash value generation.

MDFPP32: FCS_HTTPS_EXT.1: The TOE supports the HTTPS protocol (compliant with RFC 2818) so that (mobile and system) applications executing on the TOE can act as HTTPS clients and 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. The TOE does not establish the connection if the peer certificate is deemed invalid, and notify to the application that making the HTTPS connection.

MDFPP32: FCS_IV_EXT.1: The TOE generates IVs by reading from /dev/random for use with all keys, which is compliance with the requirements of Table 13: References and IV Requirements for NIST-approved Cipher Modes of MDFPP32.

MDFPP32: FCS_RBG_EXT.1: The TOE provides two RBGs including:

1. A SHA-256 Hash_DRBG provided in the hardware of the Application Processor.
2. An AES-256 CTR_DRBG provided by BoringSSL.

The AES-256 CTR_DRBG that comes with BoringSSL is the only RBG present in the ColorOS and available for other user applications that running upon the OS. As such, the TOE provides mobile applications access (through an Android Java API) to random data drawn from its AES-256 CTR_DRBG.

The TOE seeds the Hash_DRBG of Application Processor with its hardware noise source to ensure at least 256-bits of entropy. The TOE then uses the output of Hash_DRBG to continuously fill the Linux Kernel Random Number Generator (LKRNG) input pool, then the TOE seeds its BoringSSL AES-256 CTR_DRBG using 384-bits of data from /dev/random, which get data from the LKRNG input pool, thus ensuring at least 256-bits of entropy can be got for the generated random numbers. The TOE uses its BoringSSL DRBG for all random generation including keys, IVs and salts.

MDFPP32: FCS_SRV_EXT.1: The TOE provides applications access to the cryptographic operations including encryption (AES), hashing (SHA), signing and verification (RSA & ECDSA), key hashing (HMAC), generation of asymmetric keys for key establishment (RSA and ECDH), and generation of asymmetric keys for signature generation and verification (RSA, ECDSA). The TOE provides access through the Android operating system's Java API, through the native BoringSSL API, and through the application processor module (user and kernel) APIs.

MDFPP32: FCS_STG_EXT.1: The TOE provides the user, administrator, and mobile applications the ability to import and use asymmetric public and private keys into the TOE's software-based Secure Key Storage. Certificates are stored in files using UID-based permissions and an API virtualizes the access. Additionally, the user and administrator can request the TOE to destroy the keys stored in the Secure Key Storage.

While normally mobile applications cannot use or destroy the keys of another application, applications that share a common application developer (and are thus signed by the same developer key) may do so. In other words, applications with a common developer (and which explicitly declare a shared UUID in their application manifest) may use and destroy each other's keys located within the Secure Key Storage.

The TOE also provides additional protections on keys beyond including key attestation, to allow enterprises and application developers the ability to ensure which keys have been generated securely within the phone.

Document "Key Hierarchy Figure" further explains how each type of plaintext key material is generated, stored and cleared.

MDFPP32: FCS_STG_EXT.2: The TOE employs a key hierarchy that protects all DEKs and KEKs, see document "Key Hierarchy Figure" for more information.

Long-term Trusted channel Key Material (LTTCKM, i.e., Bluetooth and WiFi keys) are encrypted using AES-256-GCM encryption within their respective configuration files.

All keys are 256-bits in size. All keys are generated using the TOE'S BoringSSL AES-256 CTR_DRBG or application processor SHA-256 Hash_DRBG. By utilizing only 256-bit KEKs, the TOE ensures that all keys are encrypted by an equal or larger sized key.

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. The TOE protects persistent Wi-Fi keys (user certificates and private keys) by storing them in the Android Key Store.

MDFPP32: FCS_STG_EXT.3: The TOE protects the integrity of all DEKs and KEKs (other than LTTCKM keys) stored in Flash by using authenticated encryption/decryption methods (GCM). The TOE protects the Wi-Fi LTTCKM keys using an AES-GCM-256 key protected by the TOE's secure key storage (KeyStore). The TOE protects the BT LTTCKM keys through an AES-CBC-256 key protected by TEE, and then a HMAC-SHA-256 key protected by TEE.

PKGTLS11: FCS_TLS_EXT.1: The TOE is implemented as the TLS client.

PKGTLS11: FCS_TLSC_EXT.1/2: The TOE provides mobile applications (through its Android API) the use of TLS version 1.1 and 1.2 together with the ciphersuites defined in Section 6.1.2.25 to all the applications that running on the ColorOS, and the TOE requires no configuration other than using the appropriate library APIs as described in the Admin Guidance. The TOE also supports to import certificate for the applications use.

When an application uses the APIs provided in the Admin Guide to attempt to establish a trusted channel connection based on TLS or HTTPS, the TOE supports only Subject Alternative Name (SAN) (DNS and IP address) as reference identifiers (the TOE does not accept reference identifiers in the Common Name[CN]). The TOE supports client (mutual) authentication using X.509v3 certificates, and there is no any other factors beyond configuration that are necessary in order for the client to engage in mutual authentication. The TOE in its evaluated configuration and, by design, supports elliptic curves for TLS (P-256 & P-384) and has a fixed set of supported curves (thus the admin cannot and need not configure any curves).

No additional configuration is needed to restrict allow the device to use the supported cipher suites, as only the claimed cipher suites are supported in the aforementioned library as each of the aforementioned ciphersuites are supported on the TOE by default or through the use of the TLS library.

While the TOE supports the use of wildcards in X.509 reference identifiers (SAN and CN), the TOE does not support certificate pinning. If the TOE cannot determine the revocation status of a peer certificate, the TOE rejects the certificate and rejects the connection.

WLANCEP10: FCS_TLSC_EXT.1/2/WLAN: The TSF supports TLS versions 1.0, 1.1, and 1.2 and also supports the selected ciphersuites utilizing SHA-1, SHA-256, and SHA-384 (see the selections in section 6 for FCS_TLSC_EXT.1/WLAN) for use with EAP-TLS as part of WPA2. The TOE in its evaluated configuration and, by design, supports only evaluated elliptic curves (P-256 & P-384 and no others) and has a fixed set of supported curves (thus the admin cannot and need not configure any curves).

Below is the list of ciphersuites that supported by the TOE:

- 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

The TOE allows the user to load and utilize authentication certificates for EAP-TLS used with WPA. The Android UI (Settings->Security->Credential storage: Install from device storage) allows the user to import an RSA or ECDSA certificate and designate its use as WiFi.

The TOE supports Elliptic Curves Extension of NIST curves (secp256r1, secp384r1) in the Client Hello handshake message by default.

PKGTLS11: FCS_TLSC_EXT.4: The TOE support secure renegotiation in accordance with RFC 5746. Rehandsharking SSL engine URL:

https://developer.android.com/reference/javax/net/ssl/SSLEngine

PKGTLS11: FCS_TLSC_EXT.5: The TOE supports Elliptic Curves Extension of NIST curves (secp256r1, secp384r1) in the Client Hello handshake message by default.

7.3 User Data Protection

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

MDFPP32: FDP_ACF_EXT.1: 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). 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 request (or declare) this permission would have its vibration requests ignored.
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 cannot automatically grant it to the requesting application. For example, any dangerous permissions requested by an application will be displayed to the user and require confirmation before proceeding or some other approach can be taken to avoid the user automatically allowing the use of such facilities.
   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 running of the application. The TOE prompts the user to review the application's requested permissions (by displaying a description of each permission group, into which individual permissions map, that an application requested access to). If the user approves, then the application is allowed to continue running. If the user disapproves, the devices continues to run, but cannot use the services protected by the denied permissions. Thereafter, the mobile device grants that application during execution access to the set of permissions declared in its Manifest file.
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.
   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 (2nd installed app) that requests this permission and that has been signed with the same developer key used to sign the application (1st installed app) declaring the permission (in the case of the example, the Android Framework itself).
4. Signature|System - 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 into a system image which need to share specific features explicitly because they are being built together.
   An example of a Signature|System 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 into a system image which need to share specific features explicitly because they are being built together.
2. system - Old synonym for 'privileged'.
3. development - this permission can also (optionally) be granted to development applications (e.g., to allow additional location reporting during beta testing).
4. appop - this permission is closely associated with an app op for controlling access.
5. pre23 - this permission can be automatically granted to apps that target API levels below API level 23 (Marshmallow/6.0).
6. installer - this permission can be automatically granted to system apps that install packages.
7. verifier - this permission can be automatically granted to system apps that verify packages.
8. preinstalled - this permission can be automatically granted to any application pre-installed on the system image (not just privileged apps) (the TOE does not prompt the user to approve the permission).

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

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

The Android 12.0 (Level 31) API (https://developer.android.com/about/versions/12/features) provides a description of the services available to mobile applications.

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

Table 13: Function Categories

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	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

Applications with a common developer have the ability to allow sharing of data between their applications. A common application developer can sign their generated APK with a common certificate or key and set the permissions of their

application to allow data sharing. When the different applications' signatures match and the proper permissions are enabled, information can then be shared as needed.

The TOE supports Enterprise profiles to provide additional separation between application and application data belonging to the Enterprise profile. Applications installed into the Enterprise versus Personal profiles cannot access each other's secure data, applications, and can have separate device administrators/managers. This functionality is built into the device by default and does not require an application download. The Enterprise administrative app (an MDM agent application installed into the Enterprise Profile) may enable cross-profile contacts search, in which case, the device owner can search the address book of the enterprise profile. Please see the Admin Guide for additional details regarding how to set up and use Enterprise profiles. Ultimately, the enterprise profile is under control of the personal profile. The personal profile can decide to remove the enterprise profile, thus deleting all information and applications stored within the enterprise profile. However, despite the "control" of the personal profile, the personal profile cannot dictate the enterprise profile to share applications or data with the personal profile; the enterprise profile MDM must allow for sharing of contacts before any information can be shared.

MDFPP32: FDP_ACF_EXT.2: The TOE allows an administrator to allow sharing of the enterprise profile address book with the normal profile. Each application group (profile) has its own calendar as well as keychain (keychain is the collection of user [not application] keys, and only the user can grant the user's applications access to use a given key in the user's keychain), thus the personal and work profiles do not share calendar appointments nor keys.

MDFPP32: FDP_DAR_EXT.1: The TOE provides Data-At-Rest AES-256 XTS hardware encryption (also known as FBE, file-based encryption) for all data stored on the TOE in the user data partition (which includes both user data and TSF data). FBE provide the Credential Encrypted (CE) storage locations available to applications, which is the default storage location and only available after the user has unlocked the device.

The TOE also has TSF data relating to key storage for TSF keys not stored in the system's Android Key Store. The TOE separately encrypts those TSF keys and data. Additionally, the TOE includes a read-only filesystem in which the TOE's system executables, libraries, and their configuration data reside. For its Data-At-Rest encryption of the data partition on the internal Flash (where the TOE stores all user data and all application data), the TOE uses an AES-256 bit DEK with XTS feedback mode to encrypt each file in the data partition using dedicated application processor hardware.

MDFPP32: FDP_DAR_EXT.2: The TOE provides a Java library for Sensitive Data Protection (SDP) that application developers can use to opt-in for sensitive data protection. When developer opt-in for SDP, all data that is received on the device destined for that application is treated as sensitive. This library provides two mechanisms, Authenticated Encryption (AE) and Brief Authenticated Encryption (BE), using the symmetric and asymmetric key scheme respectively. When the AE mechanism is used, the library calls into the TOE to generate an AES key that acts as a master KEK for the SDP encryption process. The AE mechanism could encrypt or decrypt data only when the device is unlocked. When the BE mechanism is used, the library calls into the TOE to generate an RSA key that acts as a master KEK for the SDP encryption process. When an application that uses BE mechanism receives incoming data while the device is locked, an AES symmetric DEK is generated to encrypt that data. The public key from the master RSA KEK above is then used to encrypt the AES DEK. Once the device is unlocked, the RSA KEK private key is decrypted and can be used to decrypt the AES DEK for each piece of information that was stored while the device was locked. The TOE then takes that decrypted data and could re-encrypts it with AE mechanism.

MDFPP32: FDP_IFC_EXT.1: The TOE supports the installation of VPN Client applications, which ensures all traffics other than traffic necessary to establish the VPN connection go through the VPN tunnel. The TOE routes all packets through the kernel's IPsec interface (ipsec0) when the VPN is active. When the kernel routes these data packets, it will determine whether to protect, bypass or discard according to the policy configured by the user.

There is no difference in the routing of IP traffic when using any supported baseband protocols (e.g. Wi-Fi or, LTE). The only exception to all traffic being routed to the VPN is in the instance of ICMP echo requests. The TOE uses ICMP echo responses on the local subnet to facilitate network troubleshooting and categorizes it as a part of ARP. As such, if an ICMP echo request is issued on the subnet the TOE is part of, it will respond with an ICMP echo response, but no other instances of traffic will be routed outside of the VPN.

MDFPP32: FDP_PBA_EXT.1: The TOE requires the user to enter their password to enroll, re-enroll or un-enroll any biometric templates. When the user attempts biometric authentication to the TOE, the biometric sensor takes an image of the presented biometric for comparison to the enrolled templates. The captured image is used to generate the features points and then compared to all the stored templates on the device to determine if there is a match. The complete biometric

authentication process is handled inside the TEE (including image capture, all processing and match determination). The image is provided to the biometric service to check the enrolled templates for a match to the captured image.

Password authentication is required prior to managing the authentication templates, include viewing or modifying the template information, enrolling a new template, unenrolling the existing template, trying to enable or disable the biometric authentication mechanism.

MDFPP32: FDP_STG_EXT.1: The TOE'S Trusted Anchor Database consists of the built-in certs and any additional user or admin/MDM loaded certificates. The built-in certs are individually stored in the device's read-only system image in the /system/etc/security/cacerts directory, and the user can individually disable certs through Android's user interface [Settings->Security-> Trusted Credentials]. Because the built-in CA certificates reside on the read-only system partition, the TOE places a copy of any disabled built-in certificate into the /data/misc/user/X/cacerts- removed/ directory, where 'X' represents the user's number (which starts at 0).

The TOE stores added CA certificates in the corresponding /data/misc/user/X/cacerts-added/ directory and also stores a copy of the CA certificate in the user's Secure Key Storage (residing in the /data/misc/keystore/user_X/ directory). The TOE uses Linux file permissions that prevent any mobile application or entity other than the TSF from modifying these files. Only applications registered as an administrator (such as an MDM Agent Application) have the ability to access these files, staying in accordance to the permissions established in FMT_SMF_EXT.1 and FMT_MOF_EXT.1.

MDFPP32: FDP_UPC_EXT.1/APPS: The TOE provides APIs allowing non-TSF applications (mobile applications) the ability to establish a secure channel using TLS, HTTPS, and Bluetooth BR/EDR and LE. Mobile applications can use the following Android APIs for TLS, HTTPS, and Bluetooth respectively:

1. javax.net.ssl.SSLContex
   http://developer.android.com/reference/javax/net/ssl/SSLContext.html
2. javax.net.ssl.HttpsURLConnection
   http://developer.android.com/reference/javax/net/ssl/HttpsURLConnection.html
3. android.bluetooth
   http://developer.android.com/reference/android/bluetooth/package-summary.html

7.4 Identification and Authentication

The Identification and authentication functions are designed to fulfill the following security functional requirements:

MDFPP32: FIA_AFL_EXT.1: The TOE maintains in persistent storage, for each user, the number of failed password logins since the last successful login, and upon reaching the maximum number of incorrect logins, the TOE performs a full wipe of all protected data (and in fact, wipes all user data). An administrator can adjust the number of failed logins for the cryptlock screen from the default of 10 failed logins to a value between 0 (deactivate wiping) and 50 through an MDM. The TOE validates passwords by providing them to Android's Gatekeeper (which runs in the Trusted Execution Environment). If the presented password fails to validate, the TOE increments the incorrect password counter before displaying a visual error to the user. Android's Gatekeeper keeps this password counter in persistent secure storage and increments the counter before validating the password. Upon successful validation of the password, this counter is reset back to zero. By storing the counter persistently, and by incrementing the counter prior to validating it, the TOE ensures a correct tally of failed attempts even if it loses power.

The TOE allows the user to unlock the device using his or her fingerprint or face. The TOE (through a separate counter) allows users up to 5 attempts to unlock the device via fingerprint/face before temporarily disabling fingerprint/face authentication for 30 seconds. While the TOE has temporarily disabled the fingerprint sensor/ face camera, the user can input their password to unlock the phone. After a total of 4 failed rounds of attempted fingerprint/face authentications (20 total unlock attempts), the TOE completely disables the fingerprint sensor/face camera. Once the TOE has disabled the fingerprint or face unlock entirely, it remains disabled until the user enters their password to unlock the device. Note that restarting the phone at any point disables the fingerprint sensor and face cameras automatically until the user enters a correct password and unlocks the phone, and therefore TOE restart disruptions are not applicable for biometric authentication mechanisms.

MODBT10: 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).

The TOE requires this authorization (via manual input) for mobile application use of the Bluetooth trusted channel and in situations where temporary (non-bonded) connections are formed.

MODBT10: FIA_BLT_EXT.2: The TOE prevents data transfer of any type until Bluetooth pairing has completed, there is no RFCOMM nor L2CAP data transfer can occur before pairing. Additionally, the TOE supports OBEX (OBject Exchange) through L2CAP (Logical Link Control and Adaptation Protocol).

MODBT10: FIA_BLT_EXT.3: The TOE rejects duplicate Bluetooth connections by only allowing a single session per paired device. This ensures that when the TOE receives a duplicate session attempt while the TOE already has an active session with that device, then the TOE ignores the duplicate session.

MODBT10: FIA_BLT_EXT.4: The TOE'S Bluetooth host and controller supports Bluetooth Secure Simple Pairing and the TOE utilizes this pairing method when the remote host also supports it.

Secure Simple Paring follows these phases:

1. generate ECDH key and exchange public key through ECDH procedure
2. negotiate authentication protocol
3. authenticate peer device; 4)derive BR/EDR or LE shared link key

MODBT10: FIA_BLT_EXT.6: The TOE requires explicit user authorization before granting trusted remote devices access to services associated with the OPP and MAP Bluetooth profiles. Additionally, the TOE requires explicit user authorization before granting untrusted remote devices access to services associated with all available profiles.

OPP will pair after the end user agrees to the pairing. After the pairing is completed, the TOE will pop a notification when trusted remote device transmits a file to the TOE and the user needs to agree again. Every time the TOE gets a "receive file" action, it needs user's consent.

MAP will pair after the end user agrees to the pairing. After the pairing is completed, the TOE will pop a notification that prompts to access MAP service and the user needs to agree again. Once it is agreed, the trust remote device can access MAP every time without user's further consent.

MODBT10: FIA_BLT_EXT.7: The TOE requires explicit user authorization before granting untrusted remote devices access to services associated with all available profile.

MDFPP32: FIA_BMG_EXT.1/Fingerprint: The TOE's fingerprint sensor provides a FAR of 1:100,000 with an FRR of 3%, which meets the requirements for FIA_BMG_EXT.1/Fingerprint. Calculations of FAR/FRR/SAFAR can be found in the Section 1 of document "Appendix of OPPO Find X5 Pro on ColorOS 12.1 Security Target - FAR/FRR Calculation".

Users have up to 5 attempts to unlock the phone using fingerprint before the fingerprint unlock method is disabled for 30 seconds. After the 4th unsuccessful round of unlock attempts (a total of 20 fingerprint attempts), the fingerprint sensor is disabled entirely and the user is prompted for their password. The fingerprint unlock remains disabled until the user enters their password.

Since the user can attempt to unlock the phone a total of 20 times before the fingerprint is disabled, the SAFAR of the fingerprint authentication factor is 1:5,000.

MDFPP32: FIA_BMG_EXT.1/Face: The TOE's face camera provides a FAR of 1:100,000 with an FRR of 3%, which meets the requirements for FIA_BMG_EXT.1/Face. Calculations of FAR/FRR/SAFAR can be found in the Section 2 of document "Appendix of OPPO Find X5 Pro on ColorOS 12.1 Security Target - FAR/FRR Calculation".

Users have up to 5 attempts to unlock the phone using their face before the Face Unlock method is disabled for 30 seconds. After the 4th unsuccessful round of unlock attempts (a total of 20 face recognition attempts), the face camera is disabled entirely and the user is prompted for their password. The Face unlock remains disabled until the user enters their password.

Since the user can attempt to unlock the phone a total of 20 times before Face Unlock is disabled, the SAFAR of the phone is 1:5000.

WLANCEP10: 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).

MDFPP32: FIA_PMG_EXT.1: The TOE authenticates the user through a password consisting of basic Latin characters (upper and lower case, numbers, and the special characters noted in FIA_PMG_EXT.1). The TOE defaults to requiring passwords to have a minimum of four characters but no more than sixteen, contain at least one letter; however, an MDM application can change these defaults. The Smart Lock feature is not allowed in the evaluated config as this feature circumvents the requirements for FIA_PMG_EXT.1 and many others.

MDFPP32: FIA_TRT_EXT.1: The TOE limits the number of authentication attempts through the UI to no more than 5 attempts within 30 seconds. 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).

MDFPP32: FIA_UAU.5: The TOE allows the user to authenticate using either a password or biometric methods (fingerprint sensor, face camera). Upon boot, the first unlock screen presented requires the user to enter their password to unlock the device. The fingerprint sensor and face camera is enabled only after the user enters their password for the first time.

Upon device lock during normal use of the device, the user has the ability to unlock the phone by entering their password, by using a fingerprint authentication, or by using a face authentication. The TOE verifies user's password by sending hash of the password to the TEE. FIA_PMG_EXT.1 describes the password authentication process and its security measures.

Some security related user settings (e.g. changing the password, modifying, deleting, or adding stored fingerprint or face templates, SmartLock settings, etc.) and actions (e.g. factory reset) require the user to enter their password before modifying these settings or executing these actions. In these instances, biometric authentication is not accepted to permit the referenced functions.

The TOE's evaluated configuration disallows other authentication mechanisms, such as PIN, or Smart Lock mechanisms (on-body detection, trusted places, trusted devices, trusted voice).

MDFPP32: 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 -> Fingerprint, face & password -> Set Lock screen password / Add fingerprint / Add Face" menu in the TOE's user interface. The TOE can disable Smart Lock through management controls. Only after entering their current user password the user then can elect to change their password.

MDFPP32: FIA_UAU.7: The TOE allows the user to enter the user's password from the lock screen. The TOE will, by default, display the most recently entered character of the password briefly or until the user enters the next character in the password, at which point the TOE obscures the character by replacing the character with a dot symbol.

Further, the TOE provides no feedback other than whether the fingerprint or face unlock attempt succeeded or failed.

MDFPP32: FIA_UAU_EXT.1: As described before, the TOE's key hierarchy requires the user's password in order to derive the KEK_* keys in order to decrypt other KEKs and DEKs. Thus, until it has the user's password, the TOE cannot decrypt the DEK utilized for Data-At-Rest encryption, and thus cannot decrypt the user's protected data.

MDFPP32: FIA_UAU_EXT.2: The TOE allows a user to perform the actions assigned in FIA_UAU_EXT.2.1 without first successfully authenticating.

Actions that may access internal Flash storage (e.g. take screen shots, take pictures) are automatically done by the TOE, user could not change the storage location or rename them.

When configured, the user can also launch Breeno Assistant to initiate some features of the phone. However, If the actions require accessing to the user's data (e.g. contacts for calls or messages), the phone requires the user to manually unlock the phone before the action can be completed.

Beyond those actions, a user cannot perform any other actions other than observing notifications displayed on the lock screen until after successfully authenticating. Additionally, the TOE provides the user the ability to hide the contents of notifications once a password (or any other locking authentication method) is enabled.

MDFPP32: FIA_X509_EXT.1: The TOE checks the validity of all imported CA certificates by checking for the presence of the basicConstraints extension and that the CA flag is set to TRUE as the TOE imports the certificate into 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). Additionally, the TOE verifies the extendedKeyUsage Server Authentication purpose during WPA2/EAP- TLS negotiation. The TOE'S certificate validation algorithm examines each certificate in the path (starting with the peer's certificate) and first checks for validity of that certificate (e.g., has the certificate expired; or if not yet valid, whether the certificate contains the appropriate X.509 extensions [e.g., the CA flag in the basic constraints extension for a CA certificate, or that a server certificate contains the Server Authentication purpose in the ExtendedKeyUsagefield]), then verifies each certificate in the chain (applying the same rules as above, but also ensuring that the Issuer of each certificate matches the Subject in the next rung "up" in the chain and that the chain ends in a self-signed certificate present in either the TOE'S trusted anchor database or matches a specified Root CA), and finally the TOE performs revocation checking for all certificates in the chain.

MDFPP32: FIA_X509_EXT.2: The TOE uses X.509v3 certificates during EAP-TLS, TLS, and HTTPS. The TOE comes with a built-in set of default Trusted Credentials (Android's set of trusted CA certificates plus OPPO's additional set of trusted CA certificates), and while the user cannot remove any of the built-in default CA certificates, the user can disable any of those certificates through the user interface so that certificates issued by disabled CA's cannot validate successfully. In addition, a user and an administrator/MDM can import a new trusted CA certificate into the Trust Anchor Database (the TOE stores the new CA certificate in the Security Key Store).

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

The user or administrator explicitly specifies the trusted CA that the TOE will use for EAP-TLS authentication of the server's certificate. For mobile applications, the application developer will specify whether the TOE should use the Android system Trusted CAs, use application-specified trusted CAs, or a combination of the two. In this way, the TOE always knows which trusted CAs to use.

WLANCEP10: FIA_X509_EXT.2/WLAN: The TOE uses X.509v3 certificates during EAP-TLS. The TOE comes with a built-in set of default Trusted Credentials (Android's set of trusted CA certificates), and while the user cannot remove any of the built-in default CA certificates, the user can disable any of those certificates through the user interface so that certificates issued by disabled CA's cannot validate successfully. In addition, a user and an administrator/MDM can import a new trusted CA certificate into the Trust Anchor Database (the TOE stores the new CA certificate in the Security Key Store).

The user or administrator explicitly specifies the trusted CA that the TOE will use for EAP-TLS authentication of the server's certificate. For mobile applications, the application developer will specify whether the TOE should use the Android system Trusted CAs, use application-specified trusted CAs, or a combination of the two. In this way, the TOE always knows which trusted CAs to use.

The TOE, when acting as a WPA2 supplicant uses X.509 certificates for EAP-TLS authentication. Because the TOE may not have network connectivity to a revocation server prior to being admitted to the WPA2 network and because the TOE cannot determine the IP address or hostname of the authentication server (the Wi-Fi access point proxies the supplicant's authentication request to the server), the TOE will accept the certificate of the server.

MDFPP32: FIA_X509_EXT.3: The TOE's ColorOS 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.

7.5 Security Management

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

MDFPP32: FMT_MOF_EXT.1: The TOE provides the management functions described in Table 4 in **section** 6.1.5.2. The table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative function (Add User / Add Managed Profile) and 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).

MDFPP32: FMT_SMF_EXT.1: The TOE provides all management functions indicated as mandatory ("M") by Table 4. 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. Once the phone is enrolled into the enterprise environment, the services "Add User" and "Add Managed Profile" can only be performed by the administrator. 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).

MODBT10: FMT_SMF_EXT.1/BT: The administrator can disable the radios using TOE's MDM APIs. Once disabled, a user cannot enable the radio. The TOE's Bluetooth operates at frequencies of 2.4 GHz. Bluetooth BR/EDR and Bluetooth LE use same power control for turning on/off. Bluetooth BR/EDR supports mode 2 and mode 4 level 4. Bluetooth LE supports mode 1 level 4 and mode 2. The TOE provides Bluetooth management functionalities, such as scanning for devices, connecting with devices, and managing data transfer between devices. TOE users can disable/enable Bluetooth discoverable mode and Advertising and can also change the device name which is used for the Bluetooth name.

The administrator (using the TOE's MDM APIs) can enable/disable Bluetooth tethering methods.

WLANCEP10: FMT_SMF_EXT.1/WLAN: The TOE provides the management functions described in 6.1.5.2 section WLANCEP10: FMT_SMF_EXT.1.1/WLAN. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted.

MDFPP32: FMT_SMF_EXT.2: The TOE offers MDM agents the ability to wipe protected data (including sensitive data), remove Enterprise applications, and remove all device stored Enterprise resource data upon un-enrollment and factory reset. The TOE offers MDM agents the ability to wipe protected data (effectively wiping the device) at any time. Similarly, the TOE also offers the ability to remove Enterprise applications and a full wipe of managed profile data of the TOE'S Enterprise data/applications at any time.

7.6 Protection of the TSF

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

MDFPP32: FPT_AEX_EXT.1: The Linux kernel of the TOE'S Android operating system provides address space layout randomization utilizing the get_random_long(void) kernel random function to provide eight unpredictable bits to the base address of any user-space memory mapping. The random function, though not cryptographic, ensures that one cannot predict the value of the bits.

MDFPP32: FPT_AEX_EXT.2: The TOE utilizes 5.10 Linux kernels (https://source.android.com/devices/architecture/kernel/modular-kernels#core-kernel-requirements), 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.

The Android operating system (as of Android 2.3) sets the ARM No eXecute (XN) bit on memory pages and the TOE'S ARMv8 Application Processor's Memory Management Unit (MMU) circuitry enforces the XN bits. From Android's documentation (https://source.android.com/devices/tech/security/index.html), Android 2.3 forward supports "Hardware-based No eXecute (NX) to prevent code execution on the stack and heap". Section D.5 of the ARMv8 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

MDFPP32: FPT_AEX_EXT.3: The TOE's Android operating system provides explicit mechanisms to prevent stack buffer overruns in addition to taking advantage of hardware-based No eXecute to prevent code execution on the stack and heap. Specifically, the vendor builds the TOE (Android and support libraries) using gcc-fstack-protector compile option to enable

stack overflow protection and Android takes advantage of hardware-based eXecute-Never to make the stack and heap non-executable. The vendor applies these protections to all TSF executable binaries and libraries.

MDFPP32: FPT_AEX_EXT.4: The TOE protects itself from modification by untrusted subjects using a variety of methods. The first protection employed by the TOE is a Secure Boot process that uses cryptographic signatures to ensure the authenticity and integrity of the bootloader and kernels using data fused into the device processor.

The TOE protects its REK by limiting access to only trusted applications within the TEE (Trusted Execution Environment). The TOE key manager includes a TEE module which utilizes the REK 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.

Additionally, the TOE'S Android operating system provides 'sandboxing' that ensures that each third-party mobile application executes with the file permissions of a unique Linux user ID, in a different virtual memory space. This ensures that applications cannot access each other's memory space or files and cannot access the memory space or files of other applications (notwithstanding access between applications with a common application developer).

The TOE, in its evaluated configuration has its bootloader in the locked state. This prevents a user from installing a new software image via another method than Google's proscribed OTA methods. The TOE allows an operator to download and install an OTA update through the system settings (Settings->Software Update) while the phone is running. The TOE will verify the digital signature of the new OTA before applying the new firmware.

For the install of the ColorOS build through fashboot interface, the user must apply "Unlock permission" on ColorOS, and unlock the device's bootloader via installing Unlock APK, "sideload" the correct build, reboot the phone back to the fastboot interface, re-lock the bootloader, and finally start the phone normally. For both the locking and unlocking of the bootloader, the device is factory reset as part of the process. This prevents an attacker from modifying or switching the image running on the device to allow access to sensitive data. After this first install of the official build, further updates can be done via normal OTA updates.

USSD and MMI code are not able to modify user or TSF data from the dialer at the TOE's locked state.

MDFPP32: FPT_JTA_EXT.1: The TOE'S prevents access to its processor's JTAG interface by requiring use of a signing key to authenticate prior to gaining JTAG access. Only a JTAG image with the accompanying device serial number (which is different for each mobile device) that has been signed by OPPO's private key can be used to access a device's JTAG interface. The OPPO private key corresponds to the OPPO ECDSA P-384 public key (a SHA-384 hash of which is fused into the TOE'S application processor).

JTAG pads are located on the printed circuit board, which is covered by the rear panel and the battery, they are inaccessible without breaking the rear panel. JTAG pads include TCK, TMS, TDI, TDO, TRST_N and SRST_N.

MDFPP32: FPT_KST_EXT.1: The TOE does not store any plaintext key material in its internal Flash; the TOE encrypts all keys before storing them. This ensures that irrespective of how the TOE powers down (e.g., a user commands the TOE to power down, the TOE reboots itself, or battery depletes or is removed), all keys stored in the internal Flash are wrapped with a KEK. Please refer to section 7.2 of the TSS for further information (including the KEK used) regarding the encryption of keys stored in the internal Flash. Note as well that the TOE does not use the user's fingerprint/face template to encrypt / protect key material. As the TOE encrypts all keys stored in Flash, and encrypts all the fingerprint templates in the TEE of AP, upon boot-up, the TOE must first decrypt any keys in order to utilize them.

MDFPP32: FPT_KST_EXT.2: The TOE itself (i.e., the mobile device) comprises a cryptographic module that utilizes cryptographic libraries including BoringSSL, application processor cryptography (which leverages AP hardware), and the following system-level executables that utilize KEKs: vold, wpa_supplicant, and the Android Key Store.

1. vold and QCT's application processor hardware provides Data-At-Rest encryption of the user data partition in Flash
2. wpa_supplicant provides 802.11-2014/WPA2 services
3. Android Key Store application provides key generation, storage, deletion services to mobile applications and to user through the UI

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 that the TOE does not use the user's biometric fingerprint/face to encrypt/protect key material (and instead only relies upon the user's password).

MDFPP32: FPT_KST_EXT.3: The TOE does not provide any way to export plaintext DEKs or KEKs (including all keys stored in the Android KeyStore) as the TOE chains or directly encrypts all KEKs to the REK.

Furthermore, the components of the device are designed to prevent transmission of key material outside the device. Each internal system component requiring access to a plaintext key (for example the Wi-Fi driver) must have the necessary precursor(s), whether that be a password from the user or file access to key in Flash (for example the encrypted AES key used for encryption of the Flash data partition). With those appropriate precursors, the internal system-level component may call directly to the system-level library to obtain the plaintext key value. The system library in turn requests decryption from a component executing inside the trusted execution environment (TEE) and then directly returns the plaintext key value (assuming that it can successfully decrypt the requested key, as confirmed by the CCM/GCM verification) to the calling system component. That system component will then utilize that key (in the example, the kernel which holds the key in order to encrypt and decrypt reads and writes to the encrypted user data partition files in Flash). In this way, only the internal system components responsible for a given activity have access to the plaintext key needed for the activity, and that component receives the plaintext key value directly from the system library.

For a user application do not have any access to any system-level components and only have access to keys that the application has imported into the Android KeyStore. Upon requesting access to a key, the mobile application receives the plaintext key value back from the system library through the Android API. Mobile applications do not have access to the memory space of any other user's application, so it is not possible for a malicious application to intercept the plaintext key value to then log or transmit the value off the device.

MDFPP32: FPT_NOT_EXT.1: When the TOE encounters a critical failure (either a self-test failure or TOE software integrity verification failure), a failure is message is displayed to the screen, the TOE attempts to reboot. If the failure persists between boots, 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.

MDFPP32: FPT_STM.1: The TOE requires time for the Package Manager (which installs and verifies APK signatures and certificates), image verifier, wpa_supplicant, and Android Key Store applications. These TOE components obtain time from the TOE using system API calls [e.g., time() or gettimeofday()]. An application (unless a system application is residing in /system/priv-app or signed by the vendor) cannot modify the system time as mobile applications need the Android 'SET_TIME' permission to do so. Likewise, only a process with root privileges can directly modify the system time using system-level APIs. The TOE uses the Cellular Carrier time (obtained through the Carrier's network time server) as a trusted source; however, the user can also manually set the time through the TOE'S user interface. Further, this stored time is used both for the time/date tags in audit logs and is used to track inactivity timeouts that force the TOE into a locked state.

MDFPP32: FPT_TST_EXT.1: The TOE automatically performs known answer power on self-tests (POST) on its cryptographic algorithms to ensure that they are functioning correctly. Each component providing cryptography (application processor, and BoringSSL) performs known answer tests on it cryptographic algorithms to ensure it is working correctly. Should any of the tests fail, the TOE displays an error message stating "Boot Failure" and halts the boot process, and forces a reboot of the device.

Table 14: Power-up Cryptographic Algorithm Known Answer Tests

Algorithm	Implemented in	Description
AES encryption/decryption	BoringSSL	Comparison of known answer to calculated value
SHA hashing	BoringSSL	Comparison of known answer to calculated value
RSA signature generation and verification	BoringSSL	Comparison of known answer to calculated value

ECDSA signature generation and verification	BoringSSL	Comparison of known answer to calculated value
HMAC-SHA	BoringSSL	Comparison of known answer to calculated value
DRBG random bit generation	BoringSSL	Comparison of known answer to calculated value
AES encryption/decryption	Application Processor	Comparison of known answer to calculated value
SHA hashing	Application Processor	Comparison of known answer to calculated value
HMAC-SHA	Application Processor	Comparison of known answer to calculated value

WLANCEP10: FPT_TST_EXT.1/WLAN: The TOE automatically performs known answer power on self-tests (POST) on its cryptographic algorithms to ensure that they are functioning correctly. Each component providing cryptography (application processor, and BoringSSL) performs known answer tests on their cryptographic algorithms to ensure they are working correctly. Should any of the tests fail, the TOE displays an error message stating "Boot Failure" and halts the boot process, and forces a reboot of the device.

MDFPP32: FPT_TST_EXT.2/PREKERNEL: The TOE ensures a secure boot process no matter in normal boot mode or auxiliary boot mode, i.e., fast boot. In both boot modes, 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 it loads. The TOE performs checking of the entire /system partition through use of Android's dm_verity mechanism (and while the TOE will still operate, it will log any blocks/executables that have been modified).

MDFPP32: FPT_TUD_EXT.1: The TOE'S user interface provides a method to query the current version of the TOE software/firmware (ColorOS version, baseband version, kernel version, build number, and software version) and hardware (model and version). Additionally, the TOE provides users the ability to review the currently installed apps (including 3rd party 'built-in' applications) and their version.

MDFPP32: FPT_TUD_EXT.2: The TOE verifies all OTA (Over The Air) updates to the TOE software (which includes baseband processor updates) using a public key chaining ultimately to the Root Public Key, a hardware protected key whose SHA-256 hash resides inside the application processor. Should this verification fail, the software update will fail and the update will not be installed.

The application processor 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 Root Public Key, which cannot be reprogrammed after the "write-enable" fuse has been blown).

The ColorOS of the TOE requires that all applications shall be signed with a valid signature before installing the application.

Additionally, ColorOS allows updates through OPPO App Market and Google Play updates, including both APK and APEX files. Both file types use Android APK signature format and the TOE verifies the accompanying signature prior to installing the file (additionally, ColorOS ensures that updates to existing files use the same signing certificate).

MDFPP32: FPT_TUD_EXT.3: The ColorOS on the TOE requires that all applications bear a valid signature before Android will install the application.

Additionally, ColorOS allows updates through Google Play or OPPO Play updates, including both APK and APEX files. Both file types use Android APK signature format and the TOE verifies the accompanying signature prior to installing the file (additionally, Android ensures that updates to existing files use the same signing certificate).

MDFPP32: ALC_TSU_EXT.1: To make timely security updates to the TOE, the following procedures are in place:

a. Security vulnerabilities reporting:

OPPO supports a Security Response Center for ColorOS outlined here: https://security.oppo.com/en/. This allows developers or users to search for, file, and vote on vulnerabilities that need to be fixed. This helps to ensure that all vulnerabilities that affect large numbers of people get pushed up in priority to be fixed. The user could login to the website from computer-based web browser, or directly establish a trusted channel web connection to securely file the vulnerability by following the set-up steps to establish a secure HTTPS/TLS/EAP-TLS connection from the TOE.

b. Security vulnerability response process:

OPPO creates updates and patches to resolve reported issues as quickly as possible. The delivery time for resolving an issue depends on the severity, normally from several days to 1 month for the critical or high-risk vulnerabilities, or up to 3 months for medium or low-risk vulnerabilities. The updates or patches are tested before releasing to ensure they will not adversely impact on other functions of the product. Once the testing is finished, OPPO rolls out the updates and patches, then user could query the updates of the TOE via OTA as addressed in FPT_TUD_EXT.1, and update the TOE by following the [CC_GUIDE].

c. Security updates announcement:

All the vulnerabilities are announced on website: https://security.oppo.com/en/notice.

OPPO commits to pushing out monthly security updates for the ColorOS operating system (including the Java layer and kernel, not including applications). Monthly security updates have historically been supported on OPPO products for 2 years after release. These systematic updates are designed to address the highest security problems as quickly as possible and allows OPPO to ensure their mobile phone products remain as safe as possible and any issues are addressed promptly.

7.7 TOE Access

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

MDFPP32: FTA_SSL_EXT.1: The TOE transitions to its locked state either immediately after a User initiates a lock by pressing the power button (if configured) or after a (also configurable) period of inactivity, and as part of that transition, the TOE will display a lock screen to obscure the previous contents and play a "lock sound" to indicate the phone's transition; however, the TOE'S lock screen still displays email notifications, calendar appointments, user configured widgets, text message notifications, the time, date, call notifications, battery life, signal strength, and carrier network. But without authenticating first, a user cannot perform any related actions based upon these notifications (they cannot respond to emails, calendar appointments, or text messages) other than the actions assigned in FIA_UAU_EXT.2.1 (see selections in section 6). Note that during power up, the TOE presents the user with an unlock screen. While at this screen, users can access some basic device functionality (e.g., making an emergency call) and basic system data is decrypted. Once the user enters their password, the user data partition is then decrypted and the full functionality of the phone is unlocked. After this initial screen, upon (re)locking the phone, the user is presented with an "unlock for all features and data" unlock screen. This screen puts the phone in the same state as the aforementioned lock screen, encrypting user data and locking any functionality that requires data that is decrypted by the user's password. While locked, the actions described in FIA_UAU_EXT.2.1 are available for the user to utilize.

WLANCEP10: FTA_WSE_EXT.1: The TOE allows an administrator to specify (through the use of an MDM) a list of wireless networks (SSIDs) to which the user may direct the TOE to connect to, the security type, authentication protocol, and the client credentials to be used for authentication. 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.

7.8 Trusted Path/Channels

The Trusted path/channel's function is designed to fulfill the following security functional requirements:

MODBT10: FTP_BLT_EXT.1: The TOE enables Bluetooth encryption by default, and enforces the use of encryption when transmitting data over the Bluetooth trusted channel for BR/EDR and LE. The TOE uses key pairs generated per requirement FCS_CKM_EXT.8 for Bluetooth encryption.

The TOE provides APIs allowing non-TSF applications (mobile applications) the ability to establish a secure channel using Bluetooth DR/EDR and LE.

Bluetooth (android.bluetooth):

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

MODBT10: FTP_BLT_EXT.2: The TOE terminates the connection if the remote device stops encryption while connected to the TOE.

MODBT10: FTP_BLT_EXT.3/BR, FTP_BLT_EXT.3/LE: The TOE sets the minimum encryption key size to 128 bits for Bluetooth BR/EDR and LE, and encryption key size is not able to be negotiated nor configured to smaller than 128 bits.

MDFPP32: FTP_ITC_EXT.1: The TOE provides secured (encrypted and mutually authenticated) communication channels between itself and other trusted IT products through the use of IEEE 802.11-2012, 802.1X, and EAP-TLS and TLS, HTTPS. The TOE permits itself and applications to initiate communicate via the trusted channel, and the TOE initiates communications via the WPA2 (IEEE 802.11-2012, 802.1X with EAP-TLS) trusted channel for connection to a wireless access point. The TOE provides mobile applications and MDM agents access to HTTPS and TLS via published APIs, thus facilitating administrative communication and configured enterprise connections. These APIs are accessible to any application that needs an encrypted end-to-end trusted channel. The TOE also provides the OTA via HTTPS and TLS channel.

WLANCEP10: 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 IEEE 802.11-2012, 802.1X, and EAP-TLS. The TOE initiates communications via the WPA2 (IEEE 802.11-2012, 802.1X with EAP-TLS) trusted channel for connection to a wireless access point.

8 TSF Inventory

Below is a list of user-mode TSF binaries and libraries that are used to provide the security functionality of the TOE. Each of the below are built with the "-fstack-protector" complier option to protect overflow stack attack.

Table 15: TSF name and path

Name	Path	Security Function
keystore2	/system/bin	KeyStore
gatekeeperd	/system/bin	Key Management
qseecomd	/vendor/bin	DAR
time_daemon	/vendor/bin	Time
vold	/system/bin	DAR
adbd	/system/bin	Security System Settings / Recovery
libcrypto.so	/system/lib	Crypto
libcrypto.so	/system/lib64	Crypto
libkeystore_crypto.so	/system/lib	KeyStore
libkeystore_crypto.so	/system/lib64	KeyStore
libkeyutils.so	/system/lib64	DAR
libssl.so	/system/lib	SSL/TLS
libssl.so	/system/lib64	SSL/TLS
update_engine_sideload	/system/bin/	Recovery / Initial Image Load
recovery	/system/bin	Recovery
mke2fs	/system/bin	Recovery
charger	/system/bin	Recovery
init	/system/bin	Recovery
libQSEEComAPI.so	system/vendor/lib, system/vendor/lib64	TrustZone Daemon
com.coloros.ocs.opencapabilityservice	/my_stock/app/OpenCapabilityService	DAR
wpa_supplicant	/vendor/bin/hw	WLAN

racoon	system/bin	VPN