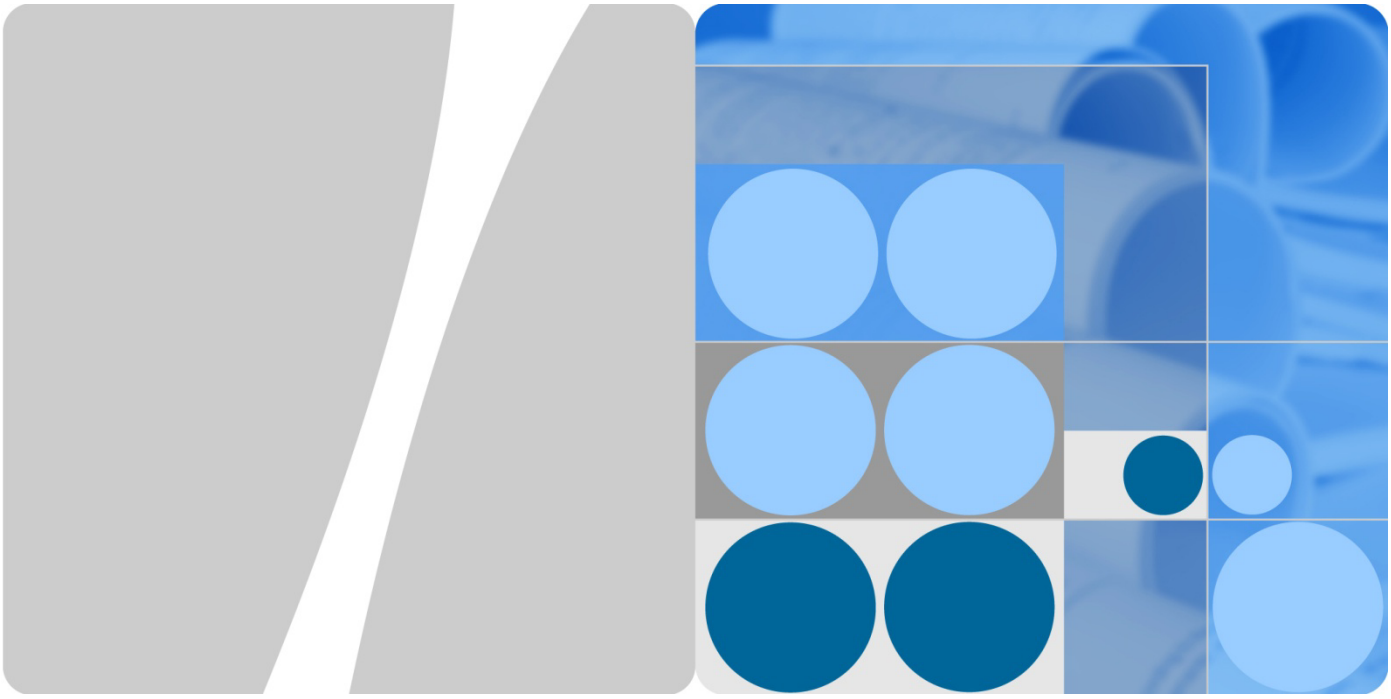

Huawei Mate 40 Pro (M40 pro) Mobile Device with EMUI 11.0 (MDFPP31/WLANCEP10) Security Target

Issue 1.0

Date 2021-11-03

HUAWEI DEVICE CO., LTD.

HUAWEI

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Device Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Device Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://www.huawei.com

PSIRT Email: PSIRT@huawei.com

Contents

Tables

1 Security Target Introduction

- This document is the Common Criteria (CC) Security Target (ST) for the Huawei Mate 40 Pro running EMUI 11.0 operating system to be evaluated as Mobile Device in exact compliance with:
- The Mobile Device Fundamentals Protection Profile Version 3.1, dated 16 June, 2017 [PP_MD_V3.1];
- The Extended Package (EP) Wireless Local Area Network {WLAN} Clients, Version 1.0, dated 8 February, 2016 [PP_WLAN_CLI_EP_V1.0];
- This section contains the Security Target (ST) and Target of Evaluation (TOE) identifications, TOE overview, and TOE description. The TOE is Huawei Mate 40 Pro mobile device with EMUI 11.0, as listed in section 1.4 “TOE Description”, provided by Huawei Device Co., Ltd.
- 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 Reference

- **ST Title –** Huawei Mate 40 Pro (M40 pro) Mobile Device with EMUI 11.0 (MDFPP31/WLANCEP10) Security Target
- **ST Version** – Issue 1.0
- **ST Date** – 2021-11-03

1.2 TOE Reference

- TOE identification: Huawei Mate 40 Pro (M40 pro) with EMUI 11.0, as listed in Table 1-1.
- TOE Developer: Huawei Device Co., Ltd.
- Evaluation Sponsor: Huawei Device Co., Ltd.

1.3 TOE Overview

- The TOE is the Huawei Mate 40 Pro mobile device running the EMUI 11.0 operating system.

1.4 TOE Description

- The Target of Evaluation (TOE) is the Huawei Mate 40 Pro mobile device running the EMUI 11.0 operating system, as detailed in Table 1-1. The EMUI 11.0 is a smartphone operating system which can run upon several models of Huawei mobile phones, and it provides the security functionalities required by [PP_MD_V3.1] with [PP_WLAN_CLI_EP_V1.0]. The TOE is intended for use as part of an enterprise mobility solution providing mobile staff with enterprise connectivity.
- The TOE provides wireless connectivity and creates a runtime environment for applications designed for the mobile Android environment. The TOE also provides telephony features (make and receive phone calls, send and receive SMS messages), networking features (connect to Wi-Fi networks, send and receive MMS messages, connect to mobile data networks).
- The software identification for the evaluated device is as follows:
- Kernel Version: 4.14
- Build Number: 11.0.0.165
- The TOE includes a Common Criteria mode (or “CC mode”) that an administrator can invoke by using a Mobile Device Management (MDM) system. When CC mode has been enabled, the TOE executes the following process:
- The TOE sets the system wide Android CC mode property to be enabled.
- The TOE executes three cryptographic module (Kernel Crypto, BoringSSL, CC engine) self-test after system boot up, and records security audit log event.
- The security audit log events are stored in internal storage to prevent from losing them after a power cycle.
- The TOE wipes all of protected data after the number of unsuccessful authentication attempts reaches to the max number.
- Users are not allowed to use the NM (Nano Memory) card in CC mode.

Table 1-1 The Detailed Description of Evaluated Devices

Device Name	Model Number	Chipset Vendor	CPU	Build Arch/ISA[1]	OS Version	Kernel Version
Mate 40 Pro	NOH-AN00	Hisilicon	Kirin 9000	ARM 64	EMUI 11.0	4.14

1.4.1 TOE Architecture

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

[1] ISA - Instruction Set Architecture, an ISA defines everything a machine language programmer needs to know in order to program a computer.

- The TOE also provides users with the ability to protect Data-At-Rest with AES encryption, including all user and mobile application data stored in the user's data partition. The TOE affords special protection to all user and application cryptographic keys stored in the TOE.
- Finally, the TOE interacts with an MDM server to allow enterprise control of the configuration and operation of the device so as to ensure adherence to enterprise-wide policies.

1.4.2 Required non-TOE Hardware/Software/Firmware

- The following non-TOE components are required:
- 802.11-2012 access point for WLAN connection
- Authentication Server for EAP-TLS mutual authentication and establishment of pairwise master key (PMK)
- Mobile data networks for network connectivity
- MDM server for administrative control of the TOE

1.4.3 Physical Boundaries

- The TOE's physical boundary is the physical perimeter of its enclosure. The TOE does not include the user applications that run on top of the operating system, but does include controls that limit application behavior.
- Included into the TOE scope is also the documentation listed in section 1.4.5 .

1.4.4 Logical Boundaries

- 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

1.4.4.1 Security audit

- The TOE generates audit records for a wide array of security relevant events concerning TOE usage and configuration. The TOE stores all audit records in the log files and sets access permission of logs making them unavailable to applications, to prevent from any malicious modifications. A storage area with 25MB on Flash is assigned for all log files. The storage mechanism is well designed and ensured to work well even after the audit capacity is reached.

1.4.4.2 Cryptographic support

- The TOE includes cryptographic modules with CAVP validated algorithms that are used for cryptographic functions including: asymmetric key generation and establishment, symmetric key generation, encryption/decryption, cryptographic hashing and keyed-hash message authentication. These functions are supported with suitable random bit generation, key derivation, salt generation, initialization vector generation,

secure key storage, and key and protected data destruction. These primitive cryptographic functions are used to implement security protocols such as TLS, and HTTPS and also to encrypt the media (including the generation and protection of data encryption keys and key encryption keys) used by the TOE. Many of these cryptographic functions must also be accessible as services to applications running on the TOE.

- The TOE provides cryptographic services via the following three cryptographic modules:
- The BoringSSL Cryptographic Module (User Space)
- The Kernel Crypto Cryptographic Module (Kernel Space)
- The CC engine Cryptographic Module (TEE and bootup)

1.4.4.3 User data protection

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

1.4.4.4 Identification and authentication

- The TOE supports a number of features related to identification and authentication. From a user perspective, except for limited functions such as making phone calls to an emergency number and receiving notifications, a password or Biometric Authentication Factor (BAF) must be correctly entered to unlock the TOE. Also, even when the TOE is unlocked the password must be re-entered to change the password. Passwords are obscured when entered so they cannot be read from the TOE's display and the frequency of entering passwords is limited and when a configured number of failures occurs, the TOE is wiped of user data. Passwords can be constructed using upper and lower cases characters, numbers, and special characters and password lengths of 16 characters are supported.
- The TOE can serve as an 802.1X supplicant. The TOE can use X.509v3 certificates to perform certificate validation on EAP-TLS, TLS, and HTTPS exchanges.

1.4.4.5 Security management

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

1.4.4.6 Protection of the TSF

- The TOE implements a number of features designed to protect itself to ensure the reliability and integrity of its security features. It protects particularly sensitive data such as cryptographic keys so that they are not accessible or exportable. It also provides its own timing mechanism to ensure that reliable time information is available. It enforces read, write, and execute memory page protections, uses address space layout randomization, and uses stack-based buffer overflow protections to minimize the potential to exploit application flaws. It is designed to protect itself from modification by applications as well as to isolate the address spaces of applications from one another to protect those applications.
- The TOE includes functions to perform self-tests and software/firmware integrity checking so that it can detect when it is failing or may be corrupt. If any self-test fails, the TOE does not go into an operational mode. It includes mechanisms (i.e., verification of the digital signature of each new image) so that the TOE itself can be updated while ensuring that the updates will not introduce malicious or other unexpected changes in the TOE. Digital signature checking also extends to verifying applications prior to their installation.

1.4.4.7 TOE access

- The TOE can be locked, obscuring its display, by a user or after a configured interval of inactivity.

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

1.4.4.8 Trusted path/channels

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

1.4.5 TOE Documentation

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

- [Guide] Common Criteria Guide for Huawei EMUI 11.0, Issue 0.4, 2021-09-03

2 Conformance Claims

- This TOE is conformant to the following CC specifications:
- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April, 2017.
  - Part 2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1 Revision 5, April, 2017.
  - Part 3 Extended
- Package Claims:
  - Exact conformance: Protection Profile for Mobile Device Fundamentals, Version 3.1, 16 June 2017 (MDFPP31)
  - Exact conformance: General Purpose Operating Systems Protection Profile/Mobile Device Fundamentals Protection Profile Extended Package (EP) Wireless Local Area Network (WLAN) Clients, Version 1.0, 08 February 2016 (WLANCEP10)
- Technical Decisions, all applicable technical decisions until 1 March 2021:

TD No.	Applied	Rationale
TD0174	Yes	Optional Ciphersuites for TLS. While the TD is applied to previous MDFPP version 3.0 and has not explicitly been referenced by MDFPP31 on the NIAP web site, it is included to remain consistency with previous changes to the MDFPP as well as with other recently issued PPs and packages.
TD0194 – WLANCEP10	Yes	Impacts required audit events
TD0244 – MDFPP31/WLANCEP10	Yes	Allows additional TLSC curves
TD0301 – MDFPP31	Yes	Impacts assurance activities and allows for assignment for FIA_BMG_EXT.1.1
TD0304 – MDFPP31	Yes	Impacts assurance activities
TD0305 – MDFPP31	Yes	Impacts assurance activities
TD0346 – MDFPP31	Yes	Removes selection from FMT_SMF_EXT.2.1

TD0347 – MDFPP31	No	Use Case 2 not selected
TD0351 – MDFPP31	Yes	Adds DEK selections to FCS_CKM_EXT.2.1
TD0366 – MDFPP31	Yes	FCS_COP.1(5) updated per TD
TD0369 – MDFPP31	Yes	LTTCKM present
TD0371 – MDFPP31	No	Use Case 2 not selected
TD0413 – MDFPP31	Yes	Any allowed PP-Module
TD0439 – WLANCEP10	Yes	FIA_X509_EXT.1/WLAN added for EAP-TLS certificate validation
TD0468 – MDFPP31	Yes	Impacts assurance activities
TD0470 –WLANCEP10	Yes	Impacts assurance activities
TD0492 – WLANCEP10	Yes	Impacts assurance activities
TD0502 – MDFPP31	Yes	FCS_CKM.1 and FCS_CKM.2(1) are updated per TD
TD0517 – WLANCEP10	Yes	WLAN Client Corrections for X509 and TLSC
TD0523 – MDFPP31	Yes	FIA_X509_EXT.1.1 updated per TD

2.1 Conformance Rationale

The ST conforms to the MDFPP31/WLANCEP10. As explained previously, the security problem definition, security objectives, and security requirements have been drawn from the PPs and the EP.

3 Security Problem Definition

The security problem definition has been taken from the **MDFPP31** and the **WLANCEP10**. It is reproduced here for the convenience of the reader.

3.1 Threats

T.EAVESDROP Network Eavesdropping (PP_MD_V3.1)

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 Network Attack (PP_MD_V3.1)

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 Physical Access (PP_MD_V3.1)

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 of this ST.

T.FLAWAPP Malicious or Flawed Application (PP_MD_V3.1)

Applications loaded onto the Mobile Device may include malicious or exploitable code. This code could be included intentionally by its developer 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, cameras, and microphones) 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 Persistent Presence (PP_MD_V3.1)

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 at least as well as by its legitimate owner.

T.TSF_FAILURE (PP_WLAN_CLI_EP_V1.0)

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 (PP_WLAN_CLI_EP_V1.0)

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 (PP_WLAN_CLI_EP_V1.0)

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 (PP_MD_V3.1)

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 (PP_MD_V3.1)

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

A.PRECAUTION (PP_MD_V3.1)

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

A.TRUSTED_ADMIN (PP_WLAN_CLI_EP_V1.0)

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

A.NO_TOE_BYPASS (PP_WLAN_CLI_EP_V1.0)

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

3.3 Organizational Security Policy

There are no OSPs for the Mobile Device.

4 Security Objectives

The security objectives for the TOE and the have been taken from the **MDFPP31** and the **WLANCEP10**. It is reproduced here for the convenience of the reader. Both the MDFPP31 and the WLANCEP10 offers additional information about the identified security objectives as well as a security objectives rationale, but that has not been reproduced here and the MDFPP31 and the WLANCEP10 should be consulted if there is interest in that material.

The PP from which the security objective originate is indicated in bracket as being either PP_MD_V3.1 or PP_WLAN_CLI_EP_V1.0 to indicate if they originate from MDFPP31 or from WLANCEP10 respectively.

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.COMMS Protected Communications (PP_MD_V3.1)

To address the network eavesdropping and network attack threats described in section 3.1, 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, 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, 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 Protected Storage (PP_MD_V3.1)

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 Mobile Device Configuration (PP_MD_V3.1)

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 Authorization and Authentication (PP_MD_V3.1)

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 Mobile Device Integrity (PP_MD_V3.1)

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 End User Privacy and Device Functionality (PP_MD_V3.1)

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 (PP_WLAN_CLI_EP_V1.0)

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 (PP_WLAN_CLI_EP_V1.0)

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 (PP_WLAN_CLI_EP_V1.0)

The TOE will provide the capability to generate audit data.

O.TOE_ADMINISTRATION (PP_WLAN_CLI_EP_V1.0)

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

O.TSF_SELF_TEST (PP_WLAN_CLI_EP_V1.0)

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 (PP_WLAN_CLI_EP_V1.0)

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 (PP_MD_V3.1)

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

OE.NOTIFY (PP_MD_V3.1)

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

OE.PRECAUTION (PP_MD_V3.1)

The Mobile User exercises precautions to reduce the risk of loss or theft of the Mobile Device.

OE.TRUSTED_ADMIN (PP_WLAN_CLI_EP_V1.0)

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

OE.NO_TOE_BYPASS (PP_WLAN_CLI_EP_V1.0)

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

5 Extended Components Definition

- All of the extended requirements in this ST have been drawn from the **MDFPP31** and the **WLANCEP10**. The MDFPP31 and the WLANCEP10 defines the following extended requirements and since they are not redefined in this ST the MDFPP31 or the WLANCEP10 should be consulted for more information in regard to those CC extensions. The extended components taken from the WLANCEP10 have been marked with "/WLAN" all others have been taken from MDFPP31.
- Extended SFRs:
- FCS_CKM_EXT.1: Extended: Cryptographic Key Support
- FCS_CKM_EXT.2: Extended: Cryptographic Key Random Generation
- FCS_CKM_EXT.3: Extended: Cryptographic Key Generation
- FCS_CKM_EXT.4: Extended: Key Destruction
- FCS_CKM_EXT.5: Extended: TSF Wipe
- FCS_CKM_EXT.6: Extended: Salt Generation
- FCS_HTTPS_EXT.1: Extended: HTTPS Protocol
- FCS_IV_EXT.1: Extended: Initialization Vector Generation
- FCS_RBG_EXT.1: Extended: Cryptographic Operation (Random Bit Generation)
- FCS_SRV_EXT.1: Extended: Cryptographic Algorithm Services
- FCS_STG_EXT.1: Extended: Cryptographic Key Storage
- FCS_STG_EXT.2: Extended: Encrypted Cryptographic Key Storage
- FCS_STG_EXT.3: Extended: Integrity of encrypted key storage
- FCS_TLSC_EXT.1: Extended: TLS Protocol
- FCS_TLSC_EXT.1/WLAN: Extended: Extensible Authentication Protocol-Transport Layer Security - WLAN
- FCS_TLSC_EXT.2/WLAN: Extended: TLS Client Protocol - WLAN
- FDP_ACF_EXT.1: Extended: Security access control
- FDP_DAR_EXT.1: Extended: Protected Data Encryption
- FDP_DAR_EXT.2: Extended: Sensitive Data Encryption
- FDP_IFC_EXT.1: Extended: Subset information flow control
- FDP_PBA_EXT.1: Extended: Storage of Critical Biometric Parameters
- FDP_STG_EXT.1: Extended: User Data Storage

- FDP_UPC_EXT.1: Extended: Inter-TSF user data transfer protection
- FIA_AFL_EXT.1: Extended: Authentication failure handling
- FIA_BLT_EXT.1: Extended: Bluetooth User Authorization
- FIA_BLT_EXT.2: Extended: Bluetooth Mutual Authentication
- FIA_BLT_EXT.3: Extended: Rejection of Duplicate Bluetooth Connections
- FIA_BLT_EXT.4: Extended: Secure Simple Pairing
- FIA_BMG_EXT.1: Extended: Accuracy of Biometric Authentication
- FIA_PAE_EXT.1: Extended: Port Access Entity Authentication
- FIA_PMG_EXT.1: Extended: Password Management
- FIA_TRT_EXT.1: Extended: Authentication Throttling
- FIA_UAU_EXT.1: Extended: Authentication for Cryptographic Operation
- FIA_UAU_EXT.2: Extended: Timing of Authentication
- FIA_X509_EXT.1: Extended: Validation of certificates
- FIA_X509_EXT.1/WLAN: Extended: X.509 Certificate Validation
- FIA_X509_EXT.2: Extended: X509 certificate authentication
- FIA_X509_EXT.2/WLAN: Extended: X.509 Certificate Authentication (EAP-TLS) - WLAN
- FIA_X509_EXT.3: Extended: Request Validation of certificates
- FMT_MOF_EXT.1: Extended: Management of security functions behavior
- FMT_SMF_EXT.1: Extended: Specification of Management Functions
- FMT_SMF_EXT.1/WLAN: Extended: Specification of Management Functions -WLAN
- FMT_SMF_EXT.2: Extended: Specification of Remediation Actions
- FPT_AEX_EXT.1: Extended: Anti-Exploitation Services (ASLR)
- FPT_AEX_EXT.2: Extended: Anti-Exploitation Services (Memory Page Permissions)
- FPT_AEX_EXT.3: Extended: Anti-Exploitation Services (Overflow Protection)
- FPT_AEX_EXT.4: Extended: Domain Isolation
- PT_JTA_EXT.1: Extended: JTAG Disablement
- FPT_KST_EXT.1: Extended: Key Storage
- FPT_KST_EXT.2: Extended: No Key Transmission
- FPT_KST_EXT.3: Extended: No Plaintext Key Export
- FPT_NOT_EXT.1: Extended: Self-Test Notification
- FPT_TST_EXT.1: Extended: TSF Cryptographic Functionality Testing
- FPT_TST_EXT.1/WLAN: Extended: TSF Cryptographic Functionality Testing - WLAN
- FPT_TST_EXT.2(1): Extended: TSF Integrity Testing

- FPT_TST_EXT.2(2): Extended: TSF Integrity Testing
- FPT_TUD_EXT.1: Extended: Trusted Update: TSF version query
- FPT_TUD_EXT.2: Extended: Trusted Update Verification
- FTA_SSL_EXT.1: Extended: TSF- and User-initiated locked state
- FTA_WSE_EXT.1: Extended: Wireless Network Access - WLAN
- FTP_ITC_EXT.1: Extended: Trusted channel Communication
- FTP_ITC_EXT.1/WLAN: Extended: Trusted Channel Communication (Wireless LAN) - WLAN
- Extended SARs:
- ALC_TSU_EXT.1: Timely Security Updates

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 the MDFPP31/WLANCEP10. The refinements and operations already performed in the MDFPP31/WLANCEP10 are not identified (e.g., highlighted) here, rather the requirements have been copied from the MDFPP31/WLANCEP10 and any residual operations have been completed herein. Of particular note, the MDFPP31/WLANCEP10 made a number of refinements and completed some of the SFR operations defined in the Common Criteria (CC) and that the MDFPP31/WLANCEP10 should be consulted to identify those changes if necessary.
- The SARs are also drawn from the MDFPP31/WLANCEP10 which includes all the SARs for EAL 1 augmented with ALC_TSU_EXT.1. However, the SARs are effectively refined since requirement-specific 'Assurance Activities' are defined in the MDFPP31/WLANCEP10 that serve to ensure corresponding evaluations will yield more practical and consistent assurance than the EAL 1 assurance requirements alone. The MDFPP31/WLANCEP10 should be consulted for the assurance activity definitions.

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.
    - Iteration: allows a component to be used more than once with varying operations. Iteration is indicated by a number in parentheses after the SFR name. For example FIA_UAU.6(1) and FIA_UAU.6(2) indicate that the ST includes two iterations of the FIA_UAU.6 requirement, (1) and (2).
    - 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~~ …").

6.1 TOE Security Functional Requirements

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

Table 6-1 TOE Security Functional Components

Requirement Class	Requirement Component

FAU: Security Audit	FAU_GEN.1: Audit Data Generation
	FAU_GEN.1/WLAN: Audit Data Generation (Wireless LAN)
	FAU_STG.1: Audit Storage Protection
	FAU_STG.4: Prevention of Audit Data Loss
FCS: Cryptographic support	FCS_CKM.1: Cryptographic key generation
	FCS_CKM.1/WLAN: Cryptographic Key Generation
	FCS_CKM.2(1): Cryptographic key establishment
	FCS_CKM.2(2): Cryptographic key distribution
	FCS_CKM.2/WLAN: Cryptographic Key Distribution
	FCS_CKM_EXT.1: Extended: Cryptographic Key Support
	FCS_CKM_EXT.2: Extended: Cryptographic Key Random Generation
	FCS_CKM_EXT.3: Extended: Cryptographic Key Generation
	FCS_CKM_EXT.4: Extended: Key Destruction
	FCS_CKM_EXT.5: Extended: TSF Wipe
	FCS_CKM_EXT.6: Extended: Salt Generation
	FCS_COP.1(1): Cryptographic operation
	FCS_COP.1(2): Cryptographic operation
	FCS_COP.1(3): Cryptographic operation
	FCS_COP.1(4): Cryptographic operation
	FCS_COP.1(5): Cryptographic operation
	FCS_HTTPS_EXT.1: Extended: HTTPS Protocol
	FCS_IV_EXT.1: Extended: Initialization Vector Generation
	FCS_RBG_EXT.1: Extended: Cryptographic Operation (Random Bit Generation)

	FCS_SRV_EXT.1: Extended: Cryptographic Algorithm Services
	FCS_STG_EXT.1: Extended: Cryptographic Key Storage
	FCS_STG_EXT.2: Extended: Encrypted Cryptographic Key Storage
	FCS_STG_EXT.3: Extended: Integrity of encrypted key storage
	FCS_TLSC_EXT.1: Extended: TLS Protocol
	FCS_TLSC_EXT.1/WLAN: Extended: Extensible Authentication Protocol-Transport Layer Security - WLAN
	FCS_TLSC_EXT.2: Extended: TLS Protocol
	FCS_TLSC_EXT.2/WLAN: Extended: TLS Client Protocol - WLAN
FDP: User data protection	FDP_ACF_EXT.1: Extended: Security access control
	FDP_DAR_EXT.1: Extended: Protected Data Encryption
	FDP_DAR_EXT.2: Extended: Sensitive Data Encryption
	FDP_IFC_EXT.1: Extended: Subset information flow control
	FDP_PBA_EXT.1 Extended: Storage of Critical Biometric Parameters
	FDP_STG_EXT.1: Extended: User Data Storage
	FDP_UPC_EXT.1: Extended: Inter-TSF user data transfer protection
FIA: Identification and authentication	FIA_AFL_EXT.1: Extended: Authentication failure handling
	FIA_BLT_EXT.1: Extended: Bluetooth User Authorization
	FIA_BLT_EXT.2: Extended: Bluetooth Mutual Authentication
	FIA_BLT_EXT.3: Extended: Rejection of Duplicate Bluetooth Connections
	FIA_BLT_EXT.4: Extended: Secure Simple Pairing
	FIA_BMG_EXT.1(1): Extended: Accuracy of Biometric Authentication
	FIA_BMG_EXT.1(2): Extended: Accuracy of Biometric Authentication
	FIA_PAE_EXT.1: Extended: Port Access Entity Authentication

	FIA_PMG_EXT.1: Extended: Password Management
	FIA_TRT_EXT.1: Extended: Authentication Throttling
	FIA_UAU.5: Multiple Authentication Mechanisms
	FIA_UAU.6(1): Re-Authentication
	FIA_UAU.6(2): Re-Authentication
	FIA_UAU.7: Protected authentication feedback
	FIA_UAU_EXT.1: Extended: Authentication for Cryptographic Operation
	FIA_UAU_EXT.2: Extended: Timing of Authentication
	FIA_X509_EXT.1: Extended: Validation of certificates
	FIA_X509_EXT.1/WLAN: Extended: X.509 Certificate Validation (EAP-TLS) - WLAN
	FIA_X509_EXT.2: Extended: X509 certificate authentication
	FIA_X509_EXT.2/WLAN: Extended: X.509 Certificate Authentication (EAP-TLS) - WLAN
	FIA_X509_EXT.3: Extended: Request Validation of certificates
FMT: Security management	FMT_MOF_EXT.1: Extended: Management of security functions behavior
	FMT_SMF_EXT.1: Extended: Specification of Management Functions
	FMT_SMF_EXT.1/WLAN: Extended: Specification of Management Functions -WLAN
	FMT_SMF_EXT.2: Extended: Specification of Remediation Actions
FPT: Protection of the TSF	FPT_AEX_EXT.1: Extended: Anti-Exploitation Services (ASLR)
	FPT_AEX_EXT.2: Extended: Anti-Exploitation Services (Memory Page Permissions)
	FPT_AEX_EXT.3: Extended: Anti-Exploitation Services (Overflow Protection)
	FPT_AEX_EXT.4: Extended: Domain Isolation

	FPT_JTA_EXT.1: Extended: JTAG Disablement
	FPT_KST_EXT.1: Extended: Key Storage
	FPT_KST_EXT.2: Extended: No Key Transmission
	FPT_KST_EXT.3: Extended: No Plaintext Key Export
	FPT_NOT_EXT.1: Extended: Self-Test Notification
	FPT_STM.1: Reliable time stamps
	FPT_TST_EXT.1: Extended: TSF Cryptographic Functionality Testing
	FPT_TST_EXT.1/WLAN: Extended: TSF Cryptographic Functionality Testing - WLAN
	FPT_TST_EXT.2(1): Extended: TSF Integrity Testing
	FPT_TST_EXT.2(2): Extended: TSF Integrity Testing
	FPT_TUD_EXT.1: Extended: Trusted Update: TSF version query
	FPT_TUD_EXT.2: Extended: Trusted Update Verification
FTA: TOE access	FTA_SSL_EXT.1: Extended: TSF- and User-initiated locked state
	FTA_WSE_EXT.1: Extended: Wireless Network Access - WLAN
FTP: Trusted path/channels	FTP_ITC_EXT.1: Extended: Trusted channel Communication
	FTP_ITC_EXT.1/WLAN: Extended: Trusted Channel Communication (Wireless LAN) - WLAN

6.1.1 Security Audit (FAU)

6.1.1.1 Audit Data Generation (FAU_GEN.1)

FAU_GEN.1.1

The TSF shall be able to generate an audit record of the following auditable events:

1. Start-up and shutdown of the audit functions

2. All auditable events for the not selected level of audit

3. All administrative actions

4. Start-up and shutdown of the Rich OS

5. Insertion or removal of removable media

6. Specifically defined auditable events in Table 6-2

7. [***Audit records reaching [90%] percentage of audit capacity***]

8. [***no additional auditable events***]

FAU_GEN.1.2

The TSF shall record within each audit record at least the following information:

1. Date and time of the

2. Type of event

3. Subject identity

4. The outcome (success or failure) of the event

5. Additional information in Table 6-2

6. [***no additional information***]

Table 6-2 Mandatory Auditable Events

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	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.1	[***None***].	No additional information.
FCS_CKM.2(*)	None.	
FCS_COP.1(*)	None.	
FCS_IV_EXT.1	None.	
FCS_SRV_EXT.1	None.	

FCS_STG_EXT.1	Import or destruction of key. [***No other events***]	Identity of key. Role and identity of requestor.
FCS_STG_EXT.2	None.	
FCS_STG_EXT.3	Failure to verify integrity of stored key.	Identity of key being verified.
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_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.
FMT_MOF_EXT.1	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.2(1)	Start-up of TOE.	No additional information.
	[***None***]	[***No additional information***]
FPT_TUD_EXT.1	None.	
FTA_SSL_EXT.1	None.	

6.1.1.2 Audit Data Generation (FAU_GEN.1/WLAN)

There are additional auditable events (listed in Table 6-3) that serve to extend the FAU_GEN.1 SFR found in both the OS PP and MDF PP. The following events should be combined with those of the OS PP or MDF PP in the context of a conforming Security Target.

Table 6-3 WLAN Auditable Events

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1/WLAN	None.	
FCS_CKM.1/WLAN	None.	
FCS_CKM.2/WLAN	None.	
FCS_CKM_EXT.4	None.	
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.
FIA_PAE_EXT.1	None.	
FIA_X509_EXT.1/WLAN (TD0439 applied)	Failure to validate X.509v3 certificate.	Reason for failure of validation.
FIA_X509_EXT.2/WLAN	None.	

FMT_SMF_EXT.1/WLAN	None.	
FPT_TST_EXT.1/WLAN	Execution of this set of TSF self-tests. [***none***].	[***No additional information***].
FTA_WSE_EXT.1	All attempts to connect to wireless networks.	Identity of wireless networks being connected to as well as success and failures (including reason for failure).
FTP_ITC_EXT.1/WLAN	All attempts to establish a trusted channel. (TD0194 applied)	Identification of the non-TOE endpoint of the channel.

Application note: TD0439: EAP-TLS Revocation Checking added an audit event to FIA_X509_EXT.1/WLAN.

Application note: TD0194: Update to Audit of FTP_ITC_EXT.1/WLAN. The PP_WLAN_CLI_EP_V1.0 contains an audit requirement for the FTP_ITC.1/WLAN requirement that is associated with "Detection of modification of channel data." This audit record is about packet corruption and not directly security related, and should be removed.

6.1.1.3 Audit Storage Protection (FAU_STG.1)

FAU_STG.1.1

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

FAU_STG.1.2

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

6.1.1.4 Prevention of Audit Data Loss (FAU_STG.4)

FAU_STG.4.1

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

6.1.2 Cryptographic support (FCS)

6.1.2.1 Cryptographic key generation (FCS_CKM.1)

FCS_CKM.1.1

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

- ***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]***].

Application note: TD0502 Cryptographic selections and updates for MDFPP.

6.1.2.2 Cryptographic Key Generation – WLAN (FCS_CKM.1/WLAN)

FCS_CKM.1.1/WLAN

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

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

FCS_CKM.2.1(1)

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

- ***RSA-based key establishment schemes that meets the following: [NIST Special Publication 800-56B, 'Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography']***
- ***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'].***

Application note: TD0502 Cryptographic selections and updates for MDFPP.

6.1.2.4 Cryptographic key establishment (while device is locked) (FCS_CKM.2(2))

FCS_CKM.2.1(2)

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

- ***Elliptic curve-based key establishment schemes that meets the following: [NIST Special Publication 800-56A, 'Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography']***

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

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

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.

6.1.2.6 Extended: Cryptographic Key Support (FCS_CKM_EXT.1)

FCS_CKM_EXT.1.1

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

FCS_CKM_EXT.1.2

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

FCS_CKM_EXT.1.3

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

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

FCS_CKM_EXT.2.1

All DEKs shall be [***from the combination of a randomly generated DEK with another DEK or salt in a way that preserves the effective entropy of each factor by concatenating the keys and using a***

KDF (as described in SP 800-108)] with entropy corresponding to the security strength of AES key sizes of [***256***] bits. (TD0351 applied)

Application note: Additional methods for DEK formation. The methods that MDFPP allows for KEK formation should be expanded to DEKs.

6.1.2.8 Extended: Cryptographic Key Generation (FCS_CKM_EXT.3)

FCS_CKM_EXT.3.1

The TSF shall use [***asymmetric KEKs of [128 bits] security strength***, ***symmetric KEKs of [256 bits] security strength corresponding to at least the security strength of the keys encrypted by the KEK***].

FCS_CKM_EXT.3.2

The TSF shall generate all KEKs using one of the following methods:

- Derive the KEK from a Password Authentication Factor using according to FCS_COP.1.1(5) 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)]***].

Application note: TD0366: Flexibility in Password Conditioning in FCS_COP.1(5). With impact on FCS_CKM_EXT.3.2.

6.1.2.9 Extended: Key Destruction (FCS_CKM_EXT.4)

FCS_CKM_EXT.4.1

The TSF shall destroy cryptographic keys in accordance with the specified cryptographic key destruction methods:

- by clearing the KEK encrypting the target key
- in accordance with the following rules
    - For volatile memory, the destruction shall be executed by a single direct overwrite [***consisting of zeroes***].
    - For non-volatile EEPROM, the destruction shall be executed by a single direct overwrite consisting of a pseudo random pattern using the TSF's RBG (as specified in FCS_RBG_EXT.1), followed a read-verify.
    - For non-volatile flash memory, that is not wear-leveled[2], the destruction shall be executed [***by a single direct overwrite consisting of zeros followed by a read-verify***].
    - For non-volatile flash memory, that is wear-leveled, the destruction shall be executed [***by a single direct overwrite consisting of zeros***].
    - For non-volatile memory other than EEPROM and flash, the destruction shall be executed by overwriting three or more times with a random pattern that is changed before each write.

FCS_CKM_EXT.4.2

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

[2] Wear-leveling phrase and next bullet added by TD0047 and selection modified by TD0057.

6.1.2.10 Extended: TSF Wipe (FCS_CKM_EXT.5)

FCS_CKM_EXT.5.1

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

FCS_CKM_EXT.5.2

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

6.1.2.11 Extended: Salt Generation (FCS_CKM_EXT.6)

FCS_CKM_EXT.6.1

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

6.1.2.12 Cryptographic operation (FCS_COP.1(1))

FCS_COP.1.1(1)

The TSF shall perform encryption/decryption in accordance with a specified cryptographic algorithm
- AES-CBC (as defined in FIPS PUB 197, and NIST SP 800-38A) mode,
- AES-CCMP (as defined in FIPS PUB 197, NIST SP 800-38C and IEEE 802.11-2012), and

[***- AES Key Wrap (KW) (as defined in NIST SP 800-38F),***

- AES-GCM (as defined in NIST SP 800-38D),

- ***AES-XTS (as defined in NIST SP 800-38E) mode***]
and cryptographic key sizes 128-bit key sizes and [***256-bit key sizes***].

6.1.2.13 Cryptographic operation (FCS_COP.1(2))

FCS_COP.1.1(2)

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(3))

FCS_COP.1.1(3)

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

6.1.2.15 Cryptographic operation (FCS_COP.1(4))

FCS_COP.1.1(4)

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

6.1.2.16 Cryptographic operation (FCS_COP.1(5))

FCS_COP.1.1(5)

The TSF shall perform conditioning in accordance with a specified cryptographic algorithm HMAC-

[***SHA-256***] using a salt, and [***PBKDF2 with [1000] iterations***] and output cryptographic key sizes [***256***] that meet the following: NIST [***SP 800-132***].

Application note: TD0366 applied: Flexibility in Password Conditioning in FCS_COP.1(5).

6.1.2.17 Extended: HTTPS Protocol (FCS_HTTPS_EXT.1)

FCS_HTTPS_EXT.1.1
The TSF shall implement the HTTPS protocol that complies with RFC 2818.

FCS_HTTPS_EXT.1.2
The TSF shall implement HTTPS using TLS (FCS_TLSC_EXT.1).

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

6.1.2.18 Extended: Initialization Vector Generation (FCS_IV_EXT.1)

FCS_IV_EXT.1.1
The TSF shall generate IVs in accordance with Table 11 in MDFPP31: References and IV Requirements for NIST-approved Cipher Modes.

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

FCS_RBG_EXT.1.1
The TSF shall perform all deterministic random bit generation services in accordance with NIST Special Publication 800-90A using [***CTR_DRBG (AES)***].

FCS_RBG_EXT.1.2
The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [***TSF-hardware-based noise source***] with a minimum of [***256 bits***] of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

FCS_RBG_EXT.1.3
The TSF shall be capable of providing output of the RBG to applications running on the TSF that request random bits.

6.1.2.20 Extended: Cryptographic Algorithm Services (FCS_SRV_EXT.1)

FCS_SRV_EXT.1.1
The TSF shall provide a mechanism for applications to request the TSF to perform the following cryptographic operations:

- All mandatory and [***selected algorithms***] in FCS_CKM.2(2)
- The following algorithms in FCS_COP.1(1): AES-CBC, [***AES-GCM***]
- All mandatory and selected algorithms in FCS_COP.1(3)
- All mandatory and selected algorithms in FCS_COP.1(2)
- All mandatory and selected algorithms in FCS_COP.1(4)

[

- ***All mandatory and [selected algorithms] in FCS_CKM.1,***
- ***The selected algorithms in FCS_COP.1(5)***

].

6.1.2.21 Extended: Cryptographic Key Storage (FCS_STG_EXT.1)

FCS_STG_EXT.1.1

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

FCS_STG_EXT.1.2

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

FCS_STG_EXT.1.3

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

FCS_STG_EXT.1.4

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

FCS_STG_EXT.1.5

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

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

FCS_STG_EXT.2.1

The TSF shall encrypt all DEKs, KEKs, [***WPA2 (PSKs), Bluetooth keys***] and [***all software-based key storage***] by KEKs that are [

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

].

FCS_STG_EXT.2.2

DEKs, KEKs, [***WPA2 (EAP-TLS private keys and PSK), Bluetooth keys***] and [***all software-based key storage***] shall be encrypted using one of the following methods: [***using AES in the [GCM mode]***].

Application note: TD0369: Long-term trusted channel key material. Changes to the PP application notes.

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

FCS_STG_EXT.3.1

The TSF shall protect the integrity of any encrypted DEKs and KEKs and [***[WPA2 (EAP-TLS private keys and PSKs), Bluetooth keys]***] by [***[GCM] cipher mode for encryption according to FCS_STG_EXT.2***]. (TD0369 applied)

FCS_STG_EXT.3.2

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

Application note: TD0369: Long-term trusted channel key material. Changes to the PP application notes.

6.1.2.24 Extended: TLS Protocol (FCS_TLSC_EXT.1)

FCS_TLSC_EXT.1.1

The TSF shall implement TLS 1.2 (RFC 5246) supporting the following ciphersuites:

- Optional Ciphersuites**:** [

- ***TLS_ECDHE_RSA_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_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***

]

FCS_TLSC_EXT.1.2

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

FCS_TLSC_EXT.1.3

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

FCS_TLSC_EXT.1.4

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

Application note: TD0174: The states “TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC5246” and “TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC5246” ciphersuites have been moved from the mandatory ciphersuites to the optional one.

6.1.2.25 Extended: EAP-TLS – WLAN (FCS_TLSC_EXT.1/WLAN)

FCS_TLSC_EXT.1.1/WLAN

The TSF shall implement [***TLS 1.2 (RFC 5246)***] in support of the EAP-TLS protocol as specified in RFC 5216 supporting the following ciphersuites: (TD0492 applied)

[

- ***TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***
- ***TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289***
- ***TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***

]

FCS_TLSC_EXT.1.2/WLAN

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

FCS_TLSC_EXT.1.3/WLAN

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

FCS_TLSC_EXT.1.4/WLAN

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

FCS_TLSC_EXT.1.5/WLAN

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

Application note: TD0492: TLS-EAP Ciphers and TLS versions for WLAN Client. FCS_TLSC_EXT.1.6/WLAN is deleted.

Application note: TD0517: WLAN Client Corrections for X509 and TLSC.

6.1.2.26 Extended: TLS Protocol (FCS_TLSC_EXT.2)

FCS_TLSC_EXT.2.1

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

Application note: TD0244: FCS_TLSC_EXT - TLS Client Curves Allowed. FCS_TLSC_EXT.2 in MD PP v3.1 limits the curves that a client may propose. This also affects APP PP v1.2, OS PP v4.1, Base Virtualization PP v1.0, and WLAN Client EP v1.0.

6.1.2.27 Extended: TLS Protocol - WLAN (FCS_TLSC_EXT.2/WLAN)

FCS_TLSC_EXT.2.1/WLAN

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

6.1.3 User data protection (FDP)

6.1.3.1 Extended: Security access control (FDP_ACF_EXT.1)

FDP_ACF_EXT.1.1

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

FDP_ACF_EXT.1.2

The TSF shall provide an access control policy that prevents [***application***] from accessing [***private***] data stored by other [***application***]. Exceptions may only be explicitly authorized for such sharing by [***a common application developer***].

6.1.3.2 Extended: Protected Data Encryption (FDP_DAR_EXT.1)

FDP_DAR_EXT.1.1

Encryption shall cover all protected data.

FDP_DAR_EXT.1.2

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

6.1.3.3 Extended: Sensitive Data Encryption (FDP_DAR_EXT.2)

FDP_DAR_EXT.2.1

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

FDP_DAR_EXT.2.2

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

FDP_DAR_EXT.2.3

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

FDP_DAR_EXT.2.4

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

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

FDP_IFC_EXT.1.1

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

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

FDP_PBA_EXT.1.1

The TSF shall protect the authentication template [***by storing it in the secure storage area which can only be accessed in TEE***].

6.1.3.6 Extended: User Data Storage (FDP_STG_EXT.1)

FDP_STG_EXT.1.1

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

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

FDP_UPC_EXT.1.1

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

FDP_UPC_EXT.1.2

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

6.1.4 Identification and authentication (FIA)

6.1.4.1 Authentication failure handling (FIA_AFL_EXT.1)

FIA_AFL_EXT.1.1

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

FIA_AFL_EXT.1.2

The TSF shall detect when a configurable positive integer within [***1- (2^{31}-1)***] of [***non-unique***] unsuccessful authentication attempts occur related to last successful authentication for each authentication mechanism.

FIA_AFL_EXT.1.3

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

FIA_AFL_EXT.1.4

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

FIA_AFL_EXT.1.5

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

FIA_AFL_EXT.1.6

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

6.1.4.2 Extended: Bluetooth User Authorization (FIA_BLT_EXT.1)

FIA_BLT_EXT.1.1

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

6.1.4.3 Extended: Bluetooth Mutual Authentication (FIA_BLT_EXT.2)

FIA_BLT_EXT.2.1

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

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

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. (TD0468 applied)

6.1.4.5 Extended: Secure Simple Pairing (FIA_BLT_EXT.4)

FIA_BLT_EXT.4.1

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

Furthermore, Secure Simple Pairing shall be used during the pairing process if the remote device also supports it.

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

FIA_BMG_EXT.1.1(1)

The one-attempt BAF False Accept Rate (FAR) for [**fingerprint**] shall not exceed [**1:50,000**] with a one-attempt BAF False Reject Rate (FRR) not to exceed 1 in [**50**]. (TD0301 applied)

FIA_BMG_EXT.1.2(1)

The overall System Authentication False Accept Rate (SAFAR) shall be no greater than 1 in [**2,500**] within a 1% margin.

Application note: TD0301: Updates to Administrator Management and Biometric Authentication. For FIA_BMG_EXT.1.1, vendors should be allowed to assign their particular FAR as opposed to being forced to select one from the list.

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

FIA_BMG_EXT.1.1(2)

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 [**14**]. (TD0301 applied)

FIA_BMG_EXT.1.2(2)

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

Application note: TD0301: Updates to Administrator Management and Biometric Authentication. For FIA_BMG_EXT.1.1, vendors should be allowed to assign their particular FAR as opposed to being forced to select one from the list.

6.1.4.8 Extended: PAE Authentication (FIA_PAE_EXT.1)

FIA_PAE_EXT.1.1

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

6.1.4.9 Extended: Password Management (FIA_PMG_EXT.1)

FIA_PMG_EXT.1.1

The TSF shall support the following for the Password Authentication Factor:

1. Passwords shall be able to be composed of any combination of [***upper and lower case letters***], numbers, and special characters: [***"!", "@", "#", "$", "%", "^", "&", "*", "(", ")"***];

2. Password length up to [**16**] characters shall be supported.

6.1.4.10 Extended: Authentication Throttling (FIA_TRT_EXT.1)

FIA_TRT_EXT.1.1

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

6.1.4.11 Multiple Authentication Mechanisms (FIA_UAU.5)

FIA_UAU.5.1

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

FIA_UAU.5.2

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

- **Passwords**
  - **Can be used at any time**
- **Biometric**
  - **Can only be used**
    - **when there is an enrolled biometric,**
    - **when the user enables the allow biometrics for unlock feature,**
    - **the non-critical biometric failed limit has not been reached, and**
    - **at a normal lock screen (not a crypto-lock screen[3])**

].

[3] Crypto-lock screen: the first lock screen which only accepts the password authentication method after the mobile device power on or reset. Accordingly, after users type in the correct password and unlock the device successfully, if the device locks again, the device will present a normal lock screen which accepts multiple authentication methods.

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

FIA_UAU.6.1(1)

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

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

FIA_UAU.6.1(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)

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)

FIA_UAU_EXT.1.1

The TSF shall require the user to present the Password Authentication Factor prior to decryption of protected data and encrypted DEKs, KEKs and [***no other keys***] at startup.

6.1.4.16 Extended: Timing of Authentication (FIA_UAU_EXT.2)

FIA_UAU_EXT.2.1

The TSF shall allow [***[make an emergency call, receive an incoming phone calls, take screen shots (automatically named and stored internally by the TOE), turn the TOE off, choose the keyboard input method, use the flashlight, use a calculator app, use a recorder app, use the world clock, stopwatch and timer functions in the clock app, browse/pin/remove/like the magazine pictures in the lock-screen, use camera to take photos]***] on behalf of the user to be performed before the user is authenticated.

FIA_UAU_EXT.2.2

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

6.1.4.17 Extended: Validation of certificates (FIA_X509_EXT.1)

FIA_X509_EXT.1.1

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

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a certificate in the Trust Anchor Database.
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates, 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 [***CRL as specified in RFC 5759***].
- The TSF shall validate the extendedKeyUsage field according to the following rules:
    - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.

- Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
- Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) in the extendedKeyUsage field. [conditional]
- Client certificates presented for TLS shall have Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the EKU field.

FIA_X509_EXT.1.2

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

Application note: TD0523: Updates to Certification Revocation.

6.1.4.18 Extended: X.509 Certificate Validation (FIA_X509_EXT.1/WLAN)

FIA_X509_EXT.1.1/WLAN

The TSF shall validate certificates for EAP-TLS in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation
- The certificate path must terminate with a certificate in the Trust Anchor Database
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates.
- The TSF shall validate the extendedKeyUsage field according to the following rules:
  - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field
  - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the extendedKeyUsage field.

FIA_X509_EXT.1.2/WLAN

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

Application note: TD0439: EAP-TLS Revocation Checking.

6.1.4.19 Extended: X509 certificate authentication (FIA_X509_EXT.2)

FIA_X509_EXT.2.1

The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [***TLS, HTTPS***], and [***no additional uses***].

FIA_X509_EXT.2.2

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

6.1.4.20 Extended: X509 certificate authentication (EAP-TLS) - WLAN (FIA_X509_EXT.2/WLAN)

FIA_X509_EXT.2.1/WLAN

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

Application note: TD0517: WLAN Client Corrections for X509 and TLSC. FIA_X509_EXT.2.2/WLAN is deleted.

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

FIA_X509_EXT.3.1
The TSF shall provide a certificate validation service to applications.

FIA_X509_EXT.3.2
The TSF shall respond to the requesting application with the success or failure of the validation.

6.1.5 Security management (FMT)

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

FMT_MOF_EXT.1.1
The TSF shall restrict the ability to perform the functions in column 3 of Table 6-4 to the user.

FMT_MOF_EXT.1.2
The TSF shall restrict the ability to perform the functions in column 5 of Table 6-4 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)

FMT_SMF_EXT.1.1
The TSF shall be capable of performing the functions in column 2 of Table 6-4 Security Management Functions.

Table 6-4 Security Management Functions

Management Function **Status Markers:** **M – Mandatory**	**FMT_SMF_EXT.1**	**Users (exclusive)**	**Administrator**	**MDM Policy**
1. configure password policy: a. minimum password length b. minimum password complexity c. maximum password lifetime The administrator can configure the required password characteristics (minimum length, complexity, and lifetime) using the Android MDM APIs. There are distinct settings for the passwords used to unlock the device. Length: an integer value of characters (0 = no minimum) Complexity: Unspecified, Something, Numeric, Alphabetic, Alphanumeric, Complex. Lifetime: an integer value of days (0 = no maximum)	M		M	M

2. configure session locking policy: a. screen-lock enabled/disabled b. screen lock timeout c. number of authentication failures The administrator can configure the session locking policy using the Android MDM APIs. There are distinct settings for the device. The user can also adjust each of the session locking policies for the device; however, if set by the administrator, the user can only set a more strict policy (e.g., setting the device to allow fewer authentication failures than configured by the administrator). Screen lock timeout: an integer number of minutes before the TOE locks (0 = no lock timeout) Authentication failures: an integer number	M		M	M
3. enable/disable the VPN protection: a. across device [***d. no other method***] Both users and administrator (using the TOE's MDM APIs) can configure a third party VPN client and then enable the VPN client to protect traffic.	M		M	M
4. enable/disable [**NFC[6], Bluetooth, Wi-Fi, and cellular radios**] The administrator can disable the radios using the TOE's MDM APIs. Once disabled, a user cannot enable the radio. The administrator cannot fully disable/restrict cellular voice capabilities. The TOE's radios operate at frequencies of 2.4 GHz (NFC/Bluetooth), 2.4/5 GHz (Wi-Fi), and 110M, 2545~2655 MHz (4G/LTE/5G).	M			M
5. enable/disable [**camera, microphone**]: a. across device [***d. no other method***] An administrator may configure the TOE (through an MDM agent utilizing the TOE's MDM APIs) to turn off the camera and or microphones. If the administrator has disabled either the camera or the microphones, then the user cannot use those capture devices.	M			M

6. transition to the locked state Both users and administrators (using the TOE's MDM APIs) can transition the TOE into a locked state.	M		M	
7. TSF wipe of protected data Both users and administrators (using the TOE's MDM APIs) can force the TOE to perform a full wipe (factory reset) of data.	M		M	
8. configure application installation policy by: [***a. restricting the sources of applications*** ***b. specifying a set of allowed applications based on [application name] (an application whitelist)*** ***c. denying installation of applications***] The administrator using the TOE's MDM APIs can configure the TOE so that applications cannot be installed and can also block the use of the Google Market Place. There are distinct settings for disabling the installation of applications.	M		M	M
9. import keys/secrets into the secure key storage Both users and administrators (using the TOE's MDM APIs) can import secret keys into the secure key storage.	M		M	
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		M	
11. import X.509v3 certificates into the Trust Anchor Database Both users and administrators (using the TOE's MDM APIs) can import X.509v3 certificates into the Trust Anchor Database.	M		M	
12. Remove imported X.509v3 certificates and [***default X.509v3 certificates***] in the Trust Anchor Database. Both users and administrators (using the TOE's MDM APIs) can remove imported 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		M	
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	M	M		

agent application) on the TOE prior to enrollment.				
14. remove applications Both users and administrators (using the TOE's MDM APIs) can uninstall user and administrator installed applications on the TOE.	M		M	
15. update system software Users can check for updates and cause the device to update if an update is available. An administrator can use system APIs to query the version of the TOE and query the installed applications. When an update is available, user will get a prompt in the settings UI.	M		M	
16. install applications Both users and administrators (using the TOE's MDM APIs) can install applications.	M		M	
17. remove Enterprise applications Both users and administrators (using the TOE's MDM APIs) can uninstall user and administrator installed applications on the TOE.	M		M	
18. configure the Bluetooth trusted channel: a. disable/enable the Discoverable mode (for BR/EDR) b. change the Bluetooth device name [***k. no other Bluetooth configuration,]*** TOE users can enable Bluetooth discoverable mode for a short period of time and can also change the device name which is used for the Bluetooth name.	M		M	M
19. enable/disable display notification in the locked state of: [***f. all notifications***] TOE users can configure the TOE to allow or disallow notifications while in a locked state.	M		M	
20. enable data-at rest protection The TOE meets this management function in the sense that the TOE always encrypts data-at-rest. This device behavior matches the administrator enabling data-at-rest protection and then preventing the user from altering it.	M			
21. enable removable media's data-at-rest protection Though the external card is supported in TOE, it's not allowed to	M	M		

use in CC mode.				
22. enable/disable location services: a. across device [***d. no other method***] The administrator (using the TOE's MDM APIs) can disable location services. Unless disabled by the administrator, TOE users can enable and disable location services.	M		M	M
23. enable/disable the use of [***Biometric Authentication Factor***] The TOE supports disabling fingerprint, face authentication for the TOE's normal device lock screen.	M		M	M
FMT_SMF_EXT.1(1)/WLAN				
48. configure security policy for each wireless network: a. [***specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s)***] b. security type c. authentication protocol d. client credentials to be used for authentication;	**M**		**M**	
49. specify wireless networks (SSIDs) to which the TSF may connect; (TD0470 applied)	M		M	

6.1.5.3 Extended: Specification of Management Functions – WLAN (FMT_SMF_EXT.1/WLAN)

FMT_SMF_EXT.1.1/WLAN

The TSF shall be capable of performing the functions **in rows 48 and 49 of Table 6-4**.

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

FMT_SMF_EXT.2.1

The TSF shall offer [***wipe of protected data, wipe of sensitive data, remove Enterprise applications, [remove MDM policies]***] upon unenrollment and [***no other triggers***]. (TD0346 applied)

Application note: TD0346: Revision of FMT_SMF_EXT.2 in MDF PP. Remove "alert the administrator" from the selection.

6.1.6 Protection of the TSF (FPT)

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

FPT_AEX_EXT.1.1

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

FPT_AEX_EXT.1.2

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

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

FPT_AEX_EXT.2.1

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

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

FPT_AEX_EXT.3.1

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

6.1.6.4 Extended: Domain Isolation (FPT_AEX_EXT.4)

FPT_AEX_EXT.4.1

The TSF shall protect itself from modification by untrusted subjects.

FPT_AEX_EXT.4.2

The TSF shall enforce isolation of address space between applications.

6.1.6.5 Extended: JTAG Disablement (FPT_JTA_EXT.1)

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)

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)

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)

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)

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)

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)

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 -WLAN (FPT_TST_EXT.1/WLAN)

FPT_TST_EXT.1.1/WLAN

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

FPT_TST_EXT.1.2/WLAN

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

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

FPT_TST_EXT.2.1(1)

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

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

FPT_TST_EXT.2.1(2)

The TSF shall verify the integrity of [***[the system partition]***], stored in mutable media prior to its execution through the use of [***an immutable hardware hash of an asymmetric key***].

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

FPT_TUD_EXT.1.1

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

FPT_TUD_EXT.1.2

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

FPT_TUD_EXT.1.3

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

6.1.6.16 Extended: Trusted Update Verification (FPT_TUD_EXT.2)

FPT_TUD_EXT.2.1

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

FPT_TUD_EXT.2.2

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

FPT_TUD_EXT.2.3

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

FPT_TUD_EXT.2.4

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

6.1.7 TOE access (FTA)

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

FTA_SSL_EXT.1.1

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

FTA_SSL_EXT.1.2

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

FTA_SSL_EXT.1.3

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

a) Clearing or overwriting display devices, obscuring the previous contents;

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

6.1.7.2 Extended: Wireless Network Access (FTA_WSE_EXT.1)

FTA_WSE_EXT.1.1

The TSF shall be able to attempt connections only to wireless networks specified as acceptable networks as configured by the administrator in FMT_SMF_EXT.1.1/WLAN. (TD0470 applied)

6.1.8 Trusted path/channels (FTP)

6.1.8.1 Extended: Trusted channel Communication (FTP_ITC_EXT.1)

FTP_ITC_EXT.1.1

The TSF shall use 802.11-2012, 802.1X, and EAP-TLS and [***TLS, HTTPS***] protocol to provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels, provides assured identification of its end points, protects channel data from disclosure, and detects modification of the channel data.

FTP_ITC_EXT.1.2

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

FTP_ITC_EXT.1.3

The TSF shall initiate communication via the trusted channel for wireless access point connections, administrative communication, configured enterprise connections, and [***no other connections***].

6.1.8.2 Extended: Trusted channel Communication - WLAN (FTP_ITC_EXT.1/WLAN)

FTP_ITC_EXT.1.1/WLAN

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

FTP_ITC_EXT.1.2/WLAN

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

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 6-5 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 - conformance
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.1.1c
The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.

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

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

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

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

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

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.1.1c
The operational user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

AGD_OPE.1.2c
The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.

AGD_OPE.1.3c

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

AGD_OPE.1.4c

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

AGD_OPE.1.5c

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

AGD_OPE.1.6c

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

AGD_OPE.1.7c

The operational user guidance shall be clear and reasonable.

AGD_OPE.1.1e

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

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.1.1c

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

AGD_PRE.1.2c

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

AGD_PRE.1.1e

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

AGD_PRE.1.2e

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

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.1.1c

The TOE shall be labelled with its unique reference.

ALC_CMC.1.1e

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

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.1.1c

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

ALC_CMS.1.2c

The configuration list shall uniquely identify the configuration items.

ALC_CMS.1.1e

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

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.1.1c

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

ALC_TSU_EXT.1.2c

The description shall express the time window as the length of time, in days, between public disclosure of a vulnerability and the public availability of security updates to the TOE.

ALC_TSU_EXT.1.3c

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

ALC_TSU_EXT.1.4c

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

ALC_TSU_EXT.1.1e

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

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.1.1c

The TOE shall be suitable for testing.

ATE_IND.1.1e

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

ATE_IND.1.2e

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

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.1.1c

The TOE shall be suitable for testing.

AVA_VAN.1.1e

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

AVA_VAN.1.2e

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

AVA_VAN.1.3e

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

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

FAU_GEN.1&FAU_GEN.1/WLAN: The following table enumerates the events that the TOE audits.

Table 7-1 Audit Event

Requirement	Audit Event	Content
FAU_GEN.1	Start-up and shutdown of the audit functions	
FAU_GEN.1	All administrative actions	
FAU_GEN.1	Start-up and shutdown of the OS and kernel	
FAU_GEN.1	Insertion or removal of removable media	

FAU_GEN.1	Reaching the configured audit log critical size limit	
FCS_STG_EXT.1	Import or destruction of key.	Identity of key. Role and identity of requestor.
FCS_STG_EXT.3	Failure to verify integrity of stored key.	Identity of key being verified.
FDP_DAR_EXT.2	Failure to encrypt/decrypt data.	No additional information.
FDP_STG_EXT.1	Addition or removal of certificate from Trust Anchor Database.	Subject name of certificate.
FIA_X509_EXT.1	Failure to validate X.509v3 certificate.	Reason for failure of validation.
FPT_TST_EXT.1	Initiation of self-test.	
FPT_TST_EXT.1	Failure of self-test.	
FPT_TST_EXT.2(1)	Start-up of TOE.	No additional information.

Table 7-2 WLAN Audit Event

Requirement	Audit Event	Content
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.
FIA_X509_EXT.1/WLAN	Failure to validate X.509v3 certificate.	Reason for failure of validation.
FPT_TST_EXT.1/WLAN	Execution of this set of TSF self-tests.	No additional information.

FTA_WSE_EXT.1	All attempts to connect to wireless networks.	Identity of wireless networks being connected to as well as success and failures (including reason for failure).
FTP_ITC_EXT.1/WLAN	All attempts to establish a trusted channel.	Identification of the non-TOE endpoint of the channel.

FAU_STG.1: The TOE stores its audit records in the /data/system/admin/auditlog/audit.log.n (n=0,1,2,3,4) and protects these logs by making them non-writable (file permissions) to any application other than the TSF itself (more precisely the **systemserver** process).

FAU_STG.4: The TOE maintains a set of five audit log files on Flash, and the TOE rotates logs. Every log file is named as audit.log.n (n=0~4). If audit.log.n file size exceeds 5MB, a new log file named as audit.log.m (m=n+1) is created. If n = 4, the oldest log file(audit.log.0) is discarded and overwritten in order to make room for new audit events. The five audit logs provide the TOE with 25MB of audit storage.

7.2 Cryptographic support

The TOE implements cryptographic algorithms in accordance with the following NIST standards. The programs running in the user space all rely on the BoringSSL cryptographic module to fulfill all of the cryptographic operations encryption/decryption/signing/verify etc. The BoringSSL cryptographic module provides the following algorithms.

Table 7-3 BoringSSL Cryptographic Algorithms

Algorithm	NIST Standard	SFR Reference
AES 128/256 CBC, CCM, GCM	FIPS PUB 197, NIST SP 800-38A/C/D	FCS_COP.1(1)
AES KW 128/256	NIST SP800-38F	FCS_COP.1(1)
CVL ECCDH	NIST SP 800-56A	FCS_CKM.2(1)
DRBG AES-256-CTR	NIST SP 800-90A	FCS_RBG_EXT.1(1)
ECDSA SIG(gen)/SIG(ver)/Key(gen)	FIPS Pub 186-4	FCS_CKM.1 FCS_COP.1(3)
HMAC SHA-1/256/384	FIPS Pub 198-1 & 180-4	FCS_COP.1(4)
RSA SIG(gen)/SIG(ver)/Key(gen)	FIPS Pub 186-4	FCS_CKM.1 FCS_COP.1(3)
SHS SHA-1/256/384/512	FIPS Pub 180-4	FCS_COP.1(2)
RSA-based Key Exchange	NIST SP800-56B	FCS_CKM.2(1)

The CC engine cryptographic module can only be used at initial boot time and in TEE environment. During the initial startup stage, because the software-based cryptographic module (such as BoringSSL, Kernel Crypto) are not available in BIOS and bootloader boot stage, CC engine will help BIOS and bootloader to finish the digital signature verification of the software/firmware in the boot time. CC engine also plays the role of cryptographic module in TEE and becomes a part of the secure world.

Table 7-4 CC engine Cryptographic Algorithms

Algorithm	NIST Standard	SFR Reference
RSA SIG(ver)	FIPS Pub 186-4	FCS_COP.1(3)
SHA-256	FIPS Pub 180-4 (SHA256)	FCS_COP.1(2)
DRBG AES-256-CTR	FIPS SP 800-90A	FCS_RBG_EXT.1(1)
KBKDF	NIST SP 800-108	FCS_CKM_EXT.3
AES-CBC-256	NIST SP 800-38A	FCS_COP.1(1)
ECDH	FIPS SP 186-4	FCS_CKM.2(2)

Programs running in the kernel space rely on the Kernel Crypto cryptographic module to fulfill the cryptographic operations.

Table 7-5 Kernel Crypto Cryptographic Algorithms

Algorithm	NIST Standard	SFR Reference
AES 256 CBC	FIPS PUB 197, NIST SP 800-38A	FCS_COP.1(1)
AES 256 GCM	NIST SP 800-38D	FCS_COP.1(1)

The TOE's application processor includes a source of hardware entropy that the TOE distributes throughout TrustZone[4]. And the android space use a software based entropy. The TOE's RBGs make use of that entropy when seeding/instantiating themselves.

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

- FCS_CKM.1: The TOE provides asymmetric key generation for RSA, ECDSA and ECDH. The TOE generates RSA and ECDH/ECDSA (including P-256 and P-384) keys in its BoringSSL software library. The TOE generates keys for files encryption by ECDH key agreement protocol with P-256 curve in the CC engine Crypto module. The TOE supports generating keys with a security strength of 112-bits and larger, thus supports 2048-bit RSA keys, and 256-bit/384-bit ECDH/ECDSA keys.
- FCS_CKM.1/WLAN: The TOE adheres to IEEE 802.11-2012 and IEEE 802.11ac-2014 for key generation. The TOE's wpa_supplicant provides the PRF384 for WPA2 derivation of 128-bit AES Temporal Key (using the

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

HMAC implementation provided by BoringSSL) and employs its BoringSSL AES-256 DRBG when generating random values used in the EAP-TLS and 802.11i (802.1X) 4-way handshake. The TOE supports the AES-128 CCMP encryption mode. To implement the WPA2 security protocol, there is a Wi-Fi module integrated in the TOE's chipset. The module has passed the Wi-Fi Alliance certification. For NOH-AN00, the Certification ID is WFA93651. The detail certification information can be found on the website: https://www.wi-fi.org/certification/programs by the Certification ID.

- FCS_CKM.2(1): The TOE supports RSA (800-56B) and ECDHE (ECC 800-56A) methods in TLS key establishment/exchange (the sole secure channel the TOE provides). The user and administrator need take no special configuration of the TOE as the TOE automatically generates the keys needed for negotiated TLS ciphersuite. Because the TOE only acts as a TLS client, the TOE only performs 800-56B encryption (specifically the encryption of the Pre-Master Secret using the Server's RSA public key) when participating in TLS_RSA_* based TLS handshakes. Thus, the TOE does not perform 800-56B decryption. However, the TOE's TLS client correctly handles other cryptographic errors (for example, invalid checksums, incorrect certificate types, corrupted certificates) by sending a TLS fatal alert.

- FCS_CKM.2(2): The TOE performs cryptographic key establishment in accordance with Elliptic curve-based key establishment schemes that meets the NIST Special Publication 800-56A standard. When a new user account is created, the TSF will generate a pair of device-wide key for the sensitive data storage purpose. While the device is locked, once the TOE receives sensitive data, the TSF will generate a pair of data key, and use the private part of the data key and the public part of the device-wide key to establish a key factor with the ECC key establishment schemes (as above mentioned). At the same time, a KEK (ClassKEK) is generated in the TEE. The key factor and the ClassKEK are put into the UFSC (Universal Flash Storage Controller) hardware and used to derive a file encryption key (a DEK) which is used to encrypt the received sensitive data. Eventually, the data is encrypted by the DEK with AES-XTS algorithm in UFSC hardware. After the file is encrypted, the private part of the data key, the key factor are abandoned and cleared from memory. While the device turns to unlocked state after the user typed in the correct password, the TSF decrypts the private part of the device-wide key with user password factor, and the ClassKEK in TEE. The private part of the device-wide key can also be released by the biometric authentication factor. While the application reads the encrypted file, the TSF uses the private part of the device-wide key and the public part of the data key pair to establish (recover) the key factor. The key factor and the ClassKEK are used to derive the file encryption key. Eventually, the file is decrypted. Now application can read the file as normal. In this process, the CC engine module is used.

- FCS_CKM.2/WLAN: The TOE adheres to RFC 3394, SP 800-38F, and 802.11-2012 standards and unwraps the GTK (sent encrypted with the WPA2 KEK using AES Key Wrap in an EAPOL-Key frame). The TOE, upon receiving an EAPOL frame, will subject the frame to a number of checks (frame length, EAPOL version, frame payload size, EAPOL-Key type, key data length, EAPOL-Key CCMP descriptor version, and replay counter) to ensure a proper EAPOL message and then decrypt the GTK using the KEK, thus ensuring that it does not expose the Group Temporal Key (GTK).

- FCS_CKM_EXT.1: The TOE supports a Root Encryption Key (REK) stored in a series of 256-bit fuses within the main (application) processor. The TOE generates the REK/fuse value during manufacturing using the AES-256 CTR DRBG provided by the CC Engine that is seeded by a hardware entropy source. The REK is never exported or updated after manufacturing. Requests for encryption or decryption chaining to the REK are only accessible through the Trusted Execution Environment, or TEE (TrustZone). The TEE does not allow direct access to the REK but provides services including encryption/decryption of other keys using the REK and derivation of keys from the REK through a KDF function.

- FCS_CKM_EXT.2: The File-Based Encryption (FBE) mechanism is used to protect all of protected data files in the TOE. Each File is encrypted by a unique DEK. Further, the DEKs are derived from a ClassKEK. The ClassKEK are encrypted by the KEKs (for the appropriate function) which are protected by the user factor (passsord/PIN) or the device factor (REK) or both. These ClassKEKs can only be decrypted/encrypted in TEE environment. This design guarantee the protected data file can only be decrypted by the RIGHT person in the RIGHT device.

  Note in addition to DEKs and KEKs, there are other keys that exist in the TOE:

  - WPA2 keys: EAPOL KCK, KEK, and TK (which are 128 bit keys)

- EAP-TLS session keys (128/256 bit AES keys)
- HTTPS/TLS session keys (128/256 bit AES keys)
- Bluetooth BR/EDR link keys and LE long term keys (128 bit AES keys)
- WPA2 long term key(PSK)
- mobile application keys

For those keys that are randomly generated, they are generated using the BoringSSL AES-256 CTR_DRBG meeting FCS_RBG_EXT.1.

- FCS_CKM_EXT.3: The TOE derives all password based KEKs by combining a user's password with an RBG generated salt value using PBKDF2 (in accordance with FCS_COP.1(5)). As described above, a device-wide key pair and a data key pair are used in ECC key establishment scheme to establish a KEK for protecting sensitive data received in the locked state. These key pairs are generated in accordance with FCS_CKM.1.
- FCS_CKM_EXT.4: The TOE destroys cryptographic keys when they are no longer in use by the system. The exceptions to this are public keys (that protect the boot chain and software updates) and the REK, which are never cleared. REK never leaves the processor. KEKs and DEKs stored in RAM during use are destroyed by a zero overwrite. Keys stored in Flash (i.e. eMMC) are destroyed by cryptographic erasure through a reset factory setting operation. In this operation, the user data partition is reformatted. Once reset factory setting executed, all keys (of course including the KEKs and DEKs used for FBE) of the TOE are considered cryptographically erased.
- FCS_CKM_EXT.5: The TOE provides a TOE Wipe function to erase all encrypted DEKs and KEKs that are stored in the Flash by the reset factory setting operation. In this operation, the TOE will reformat the partition. Upon completion of reformatting the Flash partition holding user data, the TOE will perform a power-cycle.
- FCS_CKM_EXT.6: The Huawei's Mobile Device utilizes Salt values in the following locations:

  1. Salts used to derive password based KEKs for FBE
  2. TLS/HTTPS (c_random, pre-master secret, ECDHE_* private value)
  3. WPA2 nonces

  All of these values come from the BoringSSL AES-256 CTR_DRBG meeting FCS_RBG_EXT.1.
- FCS_COP.1(1): The TOE performs encryption and decryption using AES in accordance with the standards, key sizes and modes referenced in the tables above. AES-XTS algorithm is implemented in the UFS controller hardware and not in the crypto modules like the other cryptographic algorithms.
- FCS_COP.1(2): The TOE performs cryptographic hashing using SHA algorithms in accordance with the standards and message digest sizes referenced in the table 7-3, 7-4, 7-5 above.
- FCS_COP.1(3): The TOE performs cryptographic signature generation and verification using either an RSA or an ECDSA scheme that is in accordance with standards referenced in the table 7-3, 7-4, 7-5 above.
- FCS_COP.1(4): The TOE performs keyed-hash message authentication using the HMAC-SHA-1, HMAC-SHA-256, and HMAC-SHA-384 algorithms, meeting the standards referenced in the tables above. These include key sizes and message digest sizes of 160, 256 and 384.
- FCS_COP.1(5): The TOE conditions a user's password per SP 800-132 using PBKDF2 using HMAC-SHA-256 with 1000 iterations, to combine a 128-bit salt with the user's password to obtain 256-bit KEKs which are used to protect the FBE (File Based Encryption) keys. The TOE utilizes several mechanisms to increase the computational complexity of the protections afforded by keys derived from passwords. Firstly, the TOE performs 1000 HMAC-SHA-256 iterations during PBKDF2 key derivation. Secondly, the TOE enforces a maximum number of incorrect login attempts prior to wiping all user data.

  The TOE's maximum incorrect login attempts (less than 2^{31}-1), password length (between 4 and 16 characters) and password complexity rules prevent exhaustive online attacks because the number of combinations of passwords that an adversary has to attempt is much higher than the maximum incorrect login attempts allowed before a device wipe is triggered.

- FCS_HTTPS_EXT.1: The TOE supports the HTTPS protocol (compliant with RFC 2818) using TLS as described by FCS_TLSC_EXT.1 so that (mobile and system) applications executing on the TOE can act as HTTPS clients and securely connect to external servers using HTTPS. Huawei has also modified TLS APIs to ensure that certificate checking performed by the TOE in accordance with FIA_X509_EXT.1, FIA_X509_EXT.2 and FIA_X509_EXT.3.

- FCS_IV_EXT.1: The initialization vectors (IVs) used by the TOE for AES-CBC and AES-GCM encryption are unpredictable 128 bits random numbers which are generated by the BoringSSL AES-256 CTR in accordance with FCS_RBG_EXT.1. The IVs that are used to support the encryption of the DEKs for FBE (File Based Encryption) use output from the kernel urandom( ) function.

- FCS_RBG_EXT.1: The TOE's application processor provides a true random bit generator (Non-deterministic RBG (NRBG)) which accumulates entropy from hardware noise sources. The TOE, as shown in the tables above, also has two DRBGs: 1) DRBG provided by the CC Engine cryptographic module; 2) DRBG provided by the BoringSSL cryptographic module. All of these DRBGs are AES-256 CTR_DRBG in accordance with NIST SP800-90A.The kernel crypto module doesn't have its own DRBG, but just use the DRBG provided by the CC engine cryptographic module.

  The TOE initializes the DRBG in the CC Engine with the random bits provided by the application processor's NRBG, ensuring at least 256-bits of entropy. The CC engine seeds the DRBG in the BoringSSL. The TOE provides output of the BoringSSL DRBG to mobile applications that request random bits.

- FCS_SRV_EXT.1: The TOE provides applications access to the cryptographic operations including encryption (AES-CBC, AES-GCM), hashing (SHA), signing and verification (RSA, ECDSA), key hashing (HMAC), password-based key-derivation functions (PBKDF2 HMAC-SHA-256), 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 kernel.

- FCS_STG_EXT.1: The TOE provides the user, the administrator and mobile applications the ability to import and use asymmetric public and private keys into the TOE's software-based Secure Key Storage. Additionally, the user and administrator can request the TOE to destroy the keys stored in the Secure Key Storage. While normally mobile applications cannot use or destroy the keys of another application, applications that share a common application developer (and are thus signed by the same developer key) may do so. In other words applications with a common developer may use and destroy each other's keys located within the Secure Key Storage.

- FCS_STG_EXT.2: All DEKs and KEKs stored in non-volatile memory are encrypted using AES-GCM with 256-bit keys. A KEK (which is ultimately protected by REK) is used to protect asymmetric key pairs. The private keys are encrypted with the KEK using AES-256 in GCM mode.

  Long-term trusted channel key material (LTTCKM), i.e. BlueTooth and Wi-Fi keys, are encrypted using AES-GCM encryption.

  Wi-Fi keys are protected through the use of 802.11-2012 KCK (Key Confirmation Key) and Key Encryption Key (KEK) keys. These keys unwrap the WPA2 GTK (Group Temporal Key) sent by an access point. Persistent Wi-Fi keys such as user certificates and corresponding private keys are protected by a KEK derived from the REK.

- FCS_STG_EXT.3: See the description of FCS_STG_EXT.2 above.

- FCS_TLSC_EXT.1/2: The TOE provides mobile applications (through its Android API) the use of TLS version 1.2 including support for the selected ciphersuites in the selections in section 6.1.2.24. The TOE supports only Subject Alternative Name (SAN) (DNS and IP address) as reference identifiers. It does not accept reference identifiers in the Common Name (CN). The TOE supports client (mutual) authentication. The TOE inherently (without requiring any configuration) supports the evaluated elliptic curves (P-256 and P-384); neither the user nor the administrator need configure anything in order for the TOE to support these curves. If the server sends an ECDHE key exchange message in the TLS connection using a non-supported ECDHE curve (for example, P-192), the TOE disconnects after receiving the server's Key Exchange handshake message. The TOE supports the use of wildcards in X.509 reference identifiers (SAN only), but doesn't support certificate pinning.

- FCS_TLSC_EXT.1/WLAN and FCS_TLSC_EXT.2/WLAN: The TSF supports TLS versions 1.2 with client (mutual) authentication and supports the following ciphersuites for use with EAP-TLS:
  - ***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, by design, supports the evaluated elliptic curves (P-256, P-384) and requires/allows no configuration of the supported curves.

7.3 User data protection

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

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

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

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

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

  When a new application first launch after installation, the TOE will list all of the permissions requested by the application and notice the user to make a choice. The user can select all of or a part of the permissions to grant.

  Multiple applications which is developed by the same developer can share data without sandbox restrictions if they use the same signature and Shared user ID.

- FDP_DAR_EXT.1: The TOE provides AES-256 XTS encryption of all data (which includes both user data and TSF data) stored on the TOE through the File Classification Encryption Mechanism (which provide asymmetric and symmetric scheme to protect protected data or sensitive data) on the device. The File Classification Encryption Mechanism generates a unique file encryption key for each individual file, encrypts it with AES-256-XTS and stores it on the internal storage. The TOE also has a read-only file system in which the TOE's system executables, libraries, and their configuration data reside.
- FDP_DAR_EXT.2: The TOE provides mobile applications the ability to mark data (which may be a key) as sensitive or not. Based on the work environment lock-state, sensitive data is protected by asymmetric key scheme if it was received in locked state, otherwise it is protected by symmetric key scheme.

  An application can determine whether sensitive data should remain encrypted in this manner or if it should be decrypted and re-encrypted (such as by a symmetric key for better performance). Applications can use this to securely receive data while the work environment is locked (such as an email application). By default, the attachment of e-mails is marked as the sensitive data, and other user data are marked as the protected data by system. Applications can specify files as sensitive data by filename or directory name.
- FDP_IFC_EXT.1: The TOE provides mobile applications with an MDM API to ensure IP routes are configured to direct IP traffic through the VPN. The TOE provide an VPN mode as Always-on VPN， if it's been set, all network traffic must be transported through this VPN tunnel, no matter Wi-Fi or 3/4G network.
- FDP_PBA_EXT.1: The TOE requires the user to enter their password to enroll, re-enroll or un-enroll any biometric templates. When the user attempts biometric authentication to the TOE, the biometric sensor takes an image of the presented biometric for comparison to the enrolled templates. The captured image is compared to all the stored templates on the device to determine if there is a match. The complete biometric authentication process is handled inside the TEE (including image capture, all processing and match determination). The image is provided to the biometric service to check the enrolled templates for a match to the captured image.
- FDP_STG_EXT.1: The TOE's Trusted Anchor Database consists of the built-in certs and any additional user or admin/MDM loaded certificates. The built-in certs are (individually stored in device's read-only system image in the /system/etc/security/cacerts directory, and the user can individually disable them through Android's user interface "*Settings->Security & privacy->More settings->Encryption and credentials->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.
- FDP_UPC_EXT.1: The TOE provides APIs allowing non-TSF applications (mobile applications) the ability to establish a secure channel using TLS, HTTPS, and Bluetooth BR/EDR/LE. Mobile applications can use the following Android APIs for TLS, HTTPS, and Bluetooth respectively:

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

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

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

7.4 Identification and authentication

The Identification and authentication function is designed to fulfill the following security functional requirements:

- FIA_AFL_EXT.1: The TOE maintains, for each user, the number of failed logins since the last successful login. If the biometric authentication failed attempt reaches the maximum number (5 times), the TOE will refuse the biometric authentication for at least 30 seconds, or only accept the password authentication login. If the password authentication failed login attempt reaches the maximum number (default 2^{31}-1 times, can be configured as from 1 to 2^{31}-1), the TOE performs a full wipe of all protected data (and in fact, wipes all user data). Turning the device power off does not affect the count of failed login attempts maintained by the TOE, because this count is persistent stored in the non-volatile storage.

- FIA_BLT_EXT.1: The TOE requires explicit user authorization before it will pair with a remote Bluetooth device. The user can make the TOE visible to other Bluetooth enabled devices and can attempt, via explicit user action, to pair with a visible device. Furthermore, the user must explicitly accept pairing attempts from other devices.

- FIA_BLT_EXT.2: The TOE requires mutual authentication between itself and the peer before data transfers over the link. When transferring data with another device, the TOE requires that the user must confirm an authorization popup displayed allowing data transfer or confirm the user passkey displayed numeric passcode matches between the two devices. Before the user confirm connection request, the TOE won't establish RFCOMM and L2CAP data transmit link with another Bluetooth device.

- FIA_BLT_EXT.3: The TOE tracks active connections and actively ignores connection attempts from Bluetooth device addresses for which the TOE already has an active connection. When the TOE authenticates with the Bluetooth peer each other, they negotiate and have a Link key agreement which will be used in the following data communication. If another Bluetooth device which has a same Bluetooth address attempt to connect to the TOE, the communication link can't be established due to the peer without the correct Link key.

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

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

- FIA_PMG_EXT.1: The TOE allows password for accounts to be composed of upper or lower case letters, numbers, and special characters including !, @, #, $, %, ^, &, *, (and). The TOE defaults to requiring passwords to have a minimum of four characters but no more than sixteen. However, an MDM application can change these defaults.

- FIA_TRT_EXT.1: The TOE doesn't allow users to authenticate through external ports. A user must authenticate through the standard User Interface (using the TOE touchscreen). The TOE limits the number of authentication attempts through the UI to no more than five attempts within 30 seconds (irrespective of what keyboard the operator uses). Thus if the current [the n^{th}] and prior four authentication attempts have failed, and the n-4^{th} attempt was less than 30 second ago, the TOE will prevent any further authentication attempts until 30 seconds has elapsed. Note as well that the TOE will wipe itself when it reaches the maximum number of unsuccessful authentication attempts (as described in FIA_AFL_EXT.1 above).

- FIA_UAU.5: The TOE, allows the user to authenticate to the device's Android lock screens using a password, fingerprint, or face, but does not support hybrid authentication (password + other authentication factor).

- FIA_UAU.6(1)/FIA_UAU.6(2): The TOE requires the user to enter their password or supply their biometric in order to unlock the TOE. Additionally the TOE requires the user to confirm their current password when accessing the "*Settings->Security & privacy ->Lock screen password ->Set lock screen password*" menu in the TOE's user interface. The TOE can disable Smart Lock through management controls. Only after entering their current user password can the user then elect to change their password. Before changing face/fingerprint template, the user must type in the correct password to re-authenticate as well.

- FIA_UAU.7: The TOE allows the user to enter the user's password from the lock screen or from the crypto-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.

- FIA_UAU_EXT.1: The TOE requires that a user enter their password in order for the TOE to derive KEK_1 which, together with the REK, is used to decrypt other KEKs and DEKs; thus the TOE cannot decrypt the protected data stored in the user data partition before it has the user's password.
- FIA_UAU_EXT.2: The TOE will allow a user to do the following things before successfully authenticating the user: make an emergency call, receive an incoming phone calls, take screen shots (automatically named and stored internally by the TOE), turn the TOE off, choose the keyboard input method, use a flashlight, use a calculator app, use a voice recorder app, use camera to take photos. Beyond those actions, a user cannot perform any other actions other than observing notifications displayed on the lock screen until after successfully authenticating. The TOE allows a user to configure, on a per application basis, whether notifications will be displayed.
- FIA_X509_EXT.1: The TOE checks the validity of all imported CA certificates by checking for the presence of the basicConstraints extension and that the CA flag is set to TRUE as the TOE imports the certificate (per RFC 5280). Additionally the TOE verifies the extendedKeyUsage Server Authentication purpose during WPA2/EAP-TLS negotiation. The TOE's certificate validation algorithm examines each certificate in the path (starting with the peer's certificate) and first checks for validity of that certificate (e.g., has the certificate expired or it not yet valid, whether the certificate contains the appropriate X.509 extensions, whether the keyUsage extension filed is CA signing purpose, , 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, or whether the extendedKeyUsage field is set as client authentication purpose for a client certificate), then verifies each certificate in the chain (applying the same rules as above, but also ensuring that the Issuer of each certificate matches the Subject in the next rung 'up' in the chain and that the chain ends in a self-signed certificate present in either the TOE's trusted anchor database or matches a specified Root CA), and finally the TOE performs revocation checking for all certificates in the chain using a Certificate Revocation List (per RFC 5759).
- FIA_X509_EXT.1/WLAN: During WPA2/EAP-TLS negotiation, the WLAN client in the TOE performs certificate validation and certificate path validation for EAP-TLS certificates as described above. Additionally, it verifies that, in the ExtendedKeyUsage field, a server certificate contains the Server Authentication purpose and a client certificate contains the Client Authentication purpose. However, the WLAN client does not perform revocation checking for the EAP-TLS certificates itself but rather relies on the underlying platform in the TOE to do that.
- FIA_X509_EXT.2/FIA_X509_EXT.2/WLAN: The TOE uses X.509v3 certificates as part of protocol processing for EAP-TLS, TLS and HTTPS. The TOE comes with a built-in set of default Trusted Credentials (Android's set of trusted CA certificates). And while the user cannot remove any of the built-in default CA certificates, the user can disable any of those certificates through the user interface so that certificates issued by disabled CA's cannot validate successfully. In addition, a user can import a new trusted CA certificate into the Keystore or an administrator can install a new certificate through an MDM.

  The TOE does not establish TLS connections itself (beyond EAP-TLS used for WPA2 Wi-Fi connections), but provides a series of APIs that mobile applications can use to check the validity of a peer certificate. The mobile application, after correctly using the specified APIs, can be assured as to the validity of the peer certificate and will not establish the trusted connection if the peer certificate cannot be verified (including validity, certification path, and revocation through CRL). Before verifying the peer certificate revocation status, the mobile application should enable the PKIXRevocationChecker Java class[5] to check the revocation status of certificates with CRLs. By default, CRL is not selected as the mechanism for checking the revocation status. The mobile application should set the revocation options as PREFER_CRLS to switch to CRL. During the process of certificate verification, the TOE cannot establish a connection with the server acting as the CRL Distribution Point (CDP), the TOE will deem the server's certificate as invalid and not establish a TLS connection with the server.
- FIA_X509_EXT.3: The TOE's Android operating system provides applications the java.security.cert.CertPathValidator Java API Class of methods for validating certificates and certification paths (certificate chains establishing a trust chain from a certificate to a trust anchor). This class is also recommended to be used by third-party Android developers for certificate validation. However, TrustedCertificateStore must

[5] About Java class PKIXRevocationChecker, more programing guide and detail can be found in the following website: https://developer.android.com/reference/java/security/cert/PKIXRevocationChecker.html

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

- FIA_BMG_EXT.1(1) and (2):

The TOE provides fingerprint biometric authentication. The user must always have a password, even when a biometric authentication method is enabled. Biometric authentication attempts are limited though there are differences based on the devices. The user can switch between mechanisms at any time though the counters will not reset between modality switches. All fingerprint attempts are enumerated in one counter with a maximum of 20 total attempts before no more attempts are allowed.

SAFAR

$$fingerprint = 1 - (1 - 2{*}10^{-5})^{20} = 4 * 10^{-4}$$

The TOE provides face biometric authentication. The user must always have a password, even when a biometric authentication method is enabled. Biometric authentication attempts are limited though there are differences based on the devices. The user can switch between mechanisms at any time though the counters will not reset between modality switches. All face attempts are enumerated in one counter with a maximum of 20 total attempts before no more attempts are allowed.

SAFAR

$$face = 1 - (1 - 1\text{x}10^{-5})^{20} = 2 * 10^{-4}$$

In order to collect FRR and FAR data, the vendor testers conduct the fingerprint biometric authentication in the normal temperature, the low temperature, the wet hand, and the highlight environments. And testers conduct the spoofing test with the fake 2D and 3D hands more than 1, 000 times.

The vendor testers also conduct the face biometric authentication in several kinds of light environment, including:

(1) Night, day, cloudy day, indoor light

(2) Light from the inverted direction and the lateral direction

(3) Face to camera as look up and look down angle

(4) Spoofing test with hard masks and silicon masks

The test activities in each environment are executed at least 10 times for every device model. Especially, the spoofing test cases are tested more than 100 times in each spoofing way.

And a data base which contains more than 20, 000 faces and fingers sample is used to test the biometric authentication to get the FFR and the FAR data too.

The False Accept Rat (FAR) and False Reject Rate (FRR) testing result are:

- fingerprint:
  - FAR:
    0.00002
  - FRR: 0.02
- face:
  - FAR:
    0.00001
  - FRR: 0.07142

All of the test activities are carried out in offline environment.

7.5 Security management

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

- FMT_MOF_EXT.1: The TOE provides the management functions described in Table 6-4 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.
- FMT_SMF_EXT.1: The TOE provides all management functions indicated as mandatory (“M”) by Table 6-4.
- FMT_SMF_EXT.1/WLAN: The TOE provides the management functions described in Table 6-4 Rows 48-49. The table includes annotations describing the roles that have access to each service and how to access the service. The TOE can be configured to enforce administrative configured restrictions by rejecting user configuration (through the UI) when attempted.
- FMT_SMF_EXT.2: The TOE when unenrolled from an MDM wipes all protected data.

7.6 Protection of the TSF

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

- FPT_AEX_EXT.1: The Linux kernel of the TOE's Android operating system provides address space layout randomization utilizing the get_random_int(void) kernel random function to provide eight unpredictable bits to the base address of any user-space memory mapping. The random function, though not cryptographic, ensures that one cannot predict the value of the bits.
- FPT_AEX_EXT.2: The TOE's Android 10.0 operating system utilizes the Linux kernel 4.14, whose memory management unit (MMU) enforces read, write, and execute permissions on all pages of virtual memory. The Android operating system (as of Android 2.3) sets the ARM No eXecute (XN) bit on memory pages and the TOE’s ARM v8 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 D4.2 of the ARM v8 Architecture Reference Manual contains additional details about the MMU of ARM-based processors:

  https://developer.arm.com/docs/ddi0487/latest/arm-architecture-reference-manual-armv8-for-armv8-a-architecture-profile
- 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 -fno-stack-protector compile option to enable stack overflow protection and android takes advantage of ARM v8 eXecute-Never to make the stack and heap non-executable. The vendor applies these protections to all TSF executable binaries and libraries (refer to 8 , TSF for a complete list).
- FPT_AEX_EXT.4: The TOE protects itself from modification by untrusted subjects using a variety of methods. The first protection employed by the mobile device is a Secure Boot process that uses cryptographic signatures to ensure the authenticity and integrity of the bootloader and kernels using data fused into the device processor. To minimize the attack face, the auxiliary boot mode is not supported in the TOE. The TOE protects its Device Key (REK) by generating and securely storing the Device Key within hardware and making it accessible only to Android TrustZone applications. The REK is used to protect all other keys in the key hierarchy. The TOE ensures that all TrustZone applications are cryptographically signed and that the signature is verified when the TrustZone application is invoked.

  Additionally, the TOE's Android operating system provides “sandboxing” that ensures that each third-party mobile application executes with the file permissions of a unique Linux User ID, in a different virtual memory space. This ensures that applications cannot access each other's memory space or files and cannot access the

memory space or files of other applications (notwithstanding access between applications with a common application developer).

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

- FPT_KST_EXT.1: The TOE does not store any plaintext key material in its internal Flash. This ensures that irrespective of how the TOE powers down (e.g., a user commands the TOE to power down, the TOE reboots itself, or battery depletes or is removed), all keys in internal Flash 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. As the TOE encrypts all keys stored in Flash, upon boot-up, the TOE must first decrypt any keys in order to utilize them.

  Though the biometric data won't be used as the key material for key derivation purpose, the biometric data still is under a strong protection. The TOE collects the original biometric data (face or fingerprint) through the sensors and transmit it into the TEE environment through the secure channel (REE cannot access). In TEE, biometrics algorithm is used to extract the biometric information which will be used as the BAF (Biometric Authentication Factor). In biometric information registering scenario, the biometric data is encrypted by AES algorithm and saved to RBMB (Replay Protected Memory Block). In biometric authenticating scenario, the extracted biometric data will be compared with the BAF to determine whether the biometric authentication pass or not. All of the plain text of the biometric data is handled in TEE, and encrypted before it is stored into the persistent storage.

- FPT_KST_EXT.2: The TOE itself (i.e., the mobile device) comprises cryptographic modules that utilize cryptographic libraries including BoringSSL, Kernel Crypto, and the following system-level executables that utilize KEKs: FBE (file based encryption), wpa_supplicant, and the Secure Key Store. The TOE ensures that plaintext key material does not leave this cryptographic module by only allowing the system-level executables access to the plaintext KEK values that protect all other keys in the TOE. The TSF software (the system-level executables) protects the plaintext KEKs and any plaintext DEK values in memory both by not providing any access to these values and by clearing them when no longer needed (in compliance with FCS_CKM_EXT.4).

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

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

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

- FPT_TST_EXT.1: The TOE utilizes a custom built service to do the power-up self-tests. When the services receives a BOOT_COMPLETE message, the service will start up and perform the self-test for the crypto modules (including BoringSSL, Kernel Crypto and hardware crypto module CC engine). If the self-test failed, the TOE presents a system dialog to user, and then goes into recovery mode once the user confirms the message. The self-test includes all of the cryptographic algorithm known answer tests shown in table.

- FPT_TST_EXT.1/WLAN: The TOE platform performs the previously mentioned self-tests to ensure the integrity of the WLAN client (wpa_supplicant) in addition to the cryptographic libraries used by the clients. The TOE checks if the hash of file block coincides with the executable file of WLAN client before launching it. If not, it means the integrity of the WLAN has been breached. In this situation, the TOE denies to launch the WLAN client, and records an integrity breaching event for the next time version updating. The system will carry out a whole-package version updating when any integrity breaching events occurred.

Table 7-6 Table 7-6 Power-up Cryptographic Algorithm Known Answer Tests

Algorithm	Implemented in	Description
AES encryption/decryption	BoringSSL	Comparison of known answer to calculated valued
ECDH key agreement	BoringSSL	Comparison of known answer to calculated valued
DRBG random bit generation	BoringSSL	Comparison of known answer to calculated valued
HMAC-SHA	BoringSSL	Comparison of known answer to calculated valued
RSA sign/verify/key generation	BoringSSL	Comparison of known answer to calculated valued
ECDSA sign/verify/key generation	BoringSSL	Comparison of known answer to calculated valued
SHA hashing	BoringSSL	Comparison of known answer to calculated valued
AES encryption/decryption	Kernel Crypto	Comparison of known answer to calculated valued
ECDH key agreement	Kernel Crypto	Comparison of known answer to calculated valued
AES encryption/decryption	CC engine	Comparison of known answer to calculated valued
SHA hashing	CC engine	Comparison of known answer to calculated valued
RSA verify	CC engine	Comparison of known answer to calculated valued
DRBG random bit generation	CC engine	Comparison of known answer to calculated valued

- FPT_TST_EXT.2(1)/FPT_TST_EXT.2(2): The TOE ensures a secure boot process in which the TOE verifies the digital signature of the Xloader, Fastboot[6], and Android OS, using the Huawei root public key (RSA key) whose hash (SHA-256) resides in the processor's internal fuses. The processor first uses the hash to verify the integrity of the Huawei root public key that is stored in the Flash. It then verifies the Xloader image signature by using this root public key. The Fastboot image includes an image signature and an RSA public key certificate (in a certificate format defined by Huawei) that was used to sign the image. The Xloader verifies the certificate, which must be signed with the same Huawei root signing key whose hash is in the processor's internal fuses. Once the certificate has been verified, the Xloader verifies the Fastboot image signature. If the signature is valid. The Xloader loads Fastboot. Fastboot in turn verifies the boot.img that contains the OS Linux Kernel and the TrustZone image, prior to starting the TrustZone image, which eventually starts the OS Linux Kernel.

  The integrity of the system partition is protected by Android's DM-verity mechanism (details available at https://source.android.com/security/verifiedboot/dm-verity). The TOE verifies the digital signature of the DM-verity hash tree, by using the same Huawei root public key.

- FPT_TUD_EXT.1: The TOE's user interface provides a method to query the current version of the TOE software/firmware (Android version, baseband version, kernel version, and build number) and hardware (model

[6] The 'Fastboot' is based upon Google's Android Fastboot with Huawei enhancements. And 'Xloader' is a Huawei proprietary boot software, it is prior to the Fastboot in the boot sequence.

number). Additionally, the TOE provides users the ability to review the currently installed apps (including 3rd party 'built-in' applications) and their version.

- FPT_TUD_EXT.2: The TOE verifies all updates to the TOE software using a public key chaining ultimately to the Huawei root public key. As described above, the hash (SHA-256) of the Huawei root public key is in e-fuses within the application processor. The Android OS on the TOE requires that all applications bear a valid signature before Android will install the application. Because the baseband processor is a part of application processor, the baseband processor software and firmware are packed into the TOE software pack, and can't be updated individually. Naturally, the baseband processor software and firmware are protected by the same integrity protection mechanism as other TOE software.
- ALC_TSU_EXT.1: To make the security issues dealt and the software updates released in time, Huawei take some effort in the operation, process and organization assurance.

  a. Assurance organization: Huawei has a dedicated organization PSIRT which is responsible for receiving, handling and disclosing security vulnerabilities in Huawei mobile products and solutions. The Huawei PSIRT joined FIRST in 2010, and follows ISO/IEC 29147:2018 to address security breaches in Huawei mobile products. This website (https://www.huawei.com/en/psirt) gives more detail about the Huawei PSIRT.

  b. Vulnerabilities reporting: Huawei encourages security researchers, industry organizations, government agencies and suppliers to report security vulnerabilities in Huawei products to Huawei PSIRT. The potential security vulnerabilities can be submitted to Huawei PSIRT via email (PSIRT@huawei.com).

  c. Security vulnerability response process: Huawei has established a vulnerabilities response mechanism and process, include the security problems receiving, problem verification, solution development, security vulnerabilities disclosure, and problem feedback. See (https://www.huawei.com/cn/psirt/vul-response-process). If the vulnerabilities come from AOSP, Huawei will report it to the android maintainer (https://source.android.com/setup/contribute/report-bugs), track the vulnerability patch releasing. As an android OEM, Huawei often synchronize the vulnerabilities information from Google.

     According to the severity of the vulnerabilities, Huawei security development team provides the patch and upgrade package to solve the problem as soon as possible. When a new security update is available, the mobile phone will receive a notification. Once the mobile phone receives the update information of patch or upgrade package, it will promote the user. If the user confirms the update, the phone will download the update and install it immediately. Besides, users can also use "software update" management menu to manually check if there is any software update available.

  d. Security vulnerabilities announcement: if the countermeasure for the security vulnerabilities has been found, the vulnerabilities will be public on the website: https://www.huawei.com/en/psirt/all-bulletins. The software update method for this security vulnerabilities can be found in the "Obtaining Fixed Software" chapter in the security vulnerabilities announcement.

     Periodically security update: Huawei introduces a monthly and quarterly security patch update schedule for the device models which are on sale or in maintenance lifecycle stage. The monthly and quarterly security patch update include patches for the Android OS, and patches for Huawei devices. The exact release time of each security patch may vary depending on the region and the device model. Every region or country has its own bulletin web pages. The supporting models and monthly security update information can be found in these web pages. For example, the global web page is:

     https://consumer.huawei.com/en/support/bulletin/

7.7 TOE access

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

- FTA_SSL_EXT.1: The TOE can become locked immediately after a User initiates an explicit lock action (i.e., pressing the power button) or after a configurable period of inactivity. As part of the transition to a locked state, the TOE displays a lock screen that hides prior contents of the display. The TOE's lock screen displays email

notifications, calendar appointments, user configured widgets, text message notifications, the time, date, call notifications, battery life, signal strength, and carrier network. In order for the user to perform any actions using these notifications, the user must authenticate. Only actions explicitly identified by FIA_UAU_EXT.2.1 (see section 6.1.4.16) are permitted prior to user login. During power up, the TOE presents the user with an initial power-up login screen (also known as the crypto-lock screen), where the user can only make an emergency call or enter the user's password in order to allow the TOE to derive the key needed to be able to access the data partition. After successfully authenticating at the power-up login screen, the TOE (when subsequently locked) presents the user with the normal lock screen.

- FTA_WSE_EXT.1: The TOE allows an administrator to specify (through the use of an MDM) a list of wireless networks (SSIDs) to which the user may direct the TOE to connect to. When not enrolled with an MDM, the TOE allows the user to control to which wireless networks the TOE should connect, but does not provide an explicit list of such networks, rather the 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/channels function is designed to fulfill the following security functional requirements:

- FTP_ITC_EXT.1 & FTP_ITC_EXT.1/WLAN: The TOE provides secured (encrypted and mutually authenticated) communication channels between itself and other trusted IT products through the use of 802.11-2012, 802.1X, and EAP-TLS, TLS, and HTTPS. The TOE permits itself and applications to initiate communication via the trusted channel, and the TOE initiates communication via the trusted channel for connection to a wireless access point. The TOE provides access to TLS via published APIs which are accessible to any application that needs an encrypted end-to-end trusted channel.

8 TSF Inventory

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

Table 8-1 TSF name and path

Name	Path	Security Function
dalvikvm32	**system/bin**	**VM**
dalvikvm64	**system/bin**	**VM**
keystore	**system/bin**	**Key Store**
libkeystore_binder.so	**system/lib** **system/lib64**	**Key Store**
libsoftkeymaster.so	**system/lib**	**Key Store**
keystore.kirin980.so	**system/lib/hw** **system/lib64/hw**	**Key Store**
vold	**system/bin**	**DAR**
wpa_supplicant	**/vendor/bin/hw**	**DAR**
libkeyutils.so	**system/lib**	**DAR**
libHewdkmgr.so	**system/lib64**	**DAR**
libkeyutils.so	**system/lib64**	**DAR**
libcrypto.so	**system/lib** **system/lib64**	**Crypto**
libssl.so	**system/lib** **system/lib64**	**SSL/TLS**

teecd	**/vendor/bin/**	**trust zone daemon**
libteec.so	**system/lib**	**trust zone client API**
libHwCustMdppSelftest.so	**system/lib64**	**Self-test**
libbspatchhwouc.so	**system/lib**	**OTA**