KATIM

# KATIM X3M on KATIM OS 13

# SECURITY TARGET LITE

Version 1.0

25/09/2025

PUBLIC

## Table of Contents

# 1 Security Target Introduction

This section outlines the identification of the Security Target (ST) and Target of Evaluation (TOE), including the ST conventions, conformance claims, and the structure of the document. The TOE is KATIM X3M running KATIM OS 13, provided by KATIM LLC, and it is evaluated as a Mobile Device.

The Security Target contains the following additional sections:

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

**Table 1. Acronyms and Terminology**

| | |
|---|---|
| API | Application Programming Interface |
| CC | Common Criteria |
| MDM | Mobile Device Management |
| OS | Operating System |
| PP | Protection Profile |
| SAR | Security Assurance Requirement |
| SFR | Security Functional Requirement |
| ST | Security Target |
| TEE | Trusted Execution Environment |
| TOE | Target of Evaluation |
| UI | User Interface |

### Typographic Conventions

The following conventions have been applied in this document:

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

## 1.1 Security Target Reference

**Table 2. Security Target Reference**

| | |
|---|---|
| ST-Lite Title | KATIM X3M on KATIM OS 13 – Security Target Lite |
| ST-Lite Version | 1.0 |
| ST-Lite Date | September 25, 2025 |

## 1.2 TOE Reference

Table 3. TOE Reference

| | |
|---|---|
| TOE Identification | KATIM X3M on KATIM OS 13 |
| TOE Developer | KATIM LLC |
| Evaluation Sponsor | KATIM LLC |

## 1.3 TOE Overview

The Target of Evaluation (TOE) is KATIM X3M mobile device running KATIM OS 13.

Table 4. Operating System (OS) Information

| OS version | Kernel Version | Security Patch Level |
|---|---|---|
| KATIM OS 13 | 5.15 | May 5, 2025. |

Table 5. Device information

| Product | Hardware Model # | SoC | Kernel |
|---|---|---|---|
| KATIM X3M | DP103 | Qualcomm Snapdragon® 8 gen 2 (QCM8550) | 5.15 |

The Target of Evaluation (TOE) provides standard telecommunication functionalities, such as initiating and receiving voice calls, and sending and receiving SMS messages, in addition to comprehensive network connectivity capabilities, supporting both 802.11ax Wi-Fi and multi-generation cellular technologies (3G WCDMA, 4G TDD & FDD LTE, 5G NR Sub 6 GHz NSA/SA).

Furthermore, the TOE facilitates secure network access by supporting client certificate-based authentication for connecting to WPA2/WPA3 networks via 802.1ax/EAP-TLS protocols, and also establishes connectivity with cellular base stations for mobile data communication.

Additionally, the TOE exposes an Application Programming Interface (API) that integrates with the KATIM OS framework, enabling OS services for mobile applications, and also providing mobile device management (MDM) capability for MDM agents to perform device management operations via the KATIM OS MDM API.

## 1.4 TOE Description

TOE is designed as a mobile product for enterprise use. To ensure compliance with the evaluated security configuration, the following settings must be enforced via the KATIM OS Management API:

1. Enforce lock screen password
2. Disable Smart Lock
3. Encryption for Wi-Fi and Bluetooth credentials must be activated using the NIAP mode DPM API
4. Installation of applications from unknown sources must be prohibited
5. Security logging must be enabled
6. Administrator-installed, connectivity-enabled applications must use NIAPSEC library for Sensitive Data Protection, Hostname Checking, Revocation Checking, and TLS Cipher-suite restriction

Adherence to these settings ensures that the device meets the Protection Profile for Mobile Device Fundamentals (PP_MDF_V3.3) requirements. Refer to the Administrative Guide on how to configure the above settings.

### 1.4.1 TOE Architecture

The Target of Evaluation (TOE) offers a comprehensive Application Programming Interface (API) that enables mobile applications to interact with the device’s system resources. It allows users to either approve or deny applications based on the API permissions requested during installation, or dynamically manage these permissions at runtime.

To ensure data security, the TOE implements Advanced Encryption Standard (AES) for Data-At-Rest (DAR) protection, which encompasses both user data and application-specific data stored within the user’s partition. The TOE employs a hierarchical key management system that utilizes a Root Encryption Key (REK) combined with the user's password to safeguard all cryptographic keys utilized within the system.

Additionally, the TOE is equipped with a discrete hardware secure element, to provide an isolated hardware anchor for cryptographic key storage (Keymaster) and to enforce password authentication with authentication throttling.

The TOE is also compatible with Mobile Device Management (MDM) solutions (not included within the scope of this evaluation), enabling enterprises to control the device configuration and enforce organization-wide security policies. MDM can be used to impose restrictions on device features (e.g. disabling the camera), enforce security settings (e.g., maximum number of login attempts), and retrieve audit logs for compliance purposes. An MDM system typically comprises two components: the MDM Agent installed on the device with administrative privileges, and the MDM Server, which issues policy enforcement commands to the agent.

> **Note:**
>
> The MDM system, including both the Agent and Server, is external to the TOE and thus not considered within the scope of this evaluation.

The TOE's architecture is organized into multiple execution levels, progressing from the hardware layer, through boot loader, to the TEE, Linux kernel and finally the KATIM OS user space. This layered design provides a structured API interface that allows applications to leverage the underlying cryptographic functions and security mechanisms provided by the TOE.

#### 1.4.1.1 Physical Boundaries

The physical boundary of the Target of Evaluation (TOE) is defined by the outer perimeter of its hardware enclosure. The TOE runs the KATIM operating system (OS), utilizing Qualcomm Snapdragon® 8 gen 2 processor. While the TOE includes mechanisms to regulate application behavior, it does not encompass the user applications that run on top of the KATIM OS.

The MDM agent - which comes pre-installed by KATIM - can be used to restrict the device functionality according to organizational policies.

For network communication, the TOE connects to 802.11-2012 wireless access points and mobile data networks. Through these network connections, the TOE can interact with MDM servers, enabling administrative control and policy enforcement over the device. Guidance documentation listed in Section 1.4.2 is also included in the TOE scope.

#### 1.4.1.2 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

**Security Audit**

The TOE implements two primary logging mechanisms, SecurityLog and Logcat, which store logs in circular memory buffers within KATIM OS. These logs can be accessed and retrieved by an MDM agent for further processing, such as storing the log data in non-volatile memory or transmitting it to an MDM server. These logging functionalities are designed to meet the requirements specified by FAU_GEN.1 in the PP_MDF_V3.3 Protection Profile. For detailed information and specifics, refer to the Security Audit in section 6.1.

**Cryptographic Support**

The TOE incorporates several cryptographic libraries, to support a wide array of cryptographic operations, including asymmetric key generation and key agreement, symmetric key generation, encryption and decryption, cryptographic hashing, and keyed-hash message authentication. These cryptographic operations are backed by compliant random bit generation, key derivation, salt generation, initialization vector generation, secure key storage, and secure destruction of keys and protected data.

These foundational cryptographic functions are utilized to implement security protocols such as TLS, EAP-TLS, and HTTPS, as well as to provide encryption for data-at-rest, including the generation and protection of data and key encryption keys. Additionally, many of these cryptographic services are made available to applications running on the TOE, enabling application developers to ensure their applications adhere to the security requirements of the PP_MDF_V3.3 Protection Profile.

**User Data Protection**

The TOE regulates access to system services for hosted applications, ensuring the protection of critical resources such as the Trust Anchor Database. In addition, the TOE employs encryption to safeguard user and other sensitive data, ensuring the data remains secure even in the event of device loss. The root encryption key derivation process depends on both the SoC of the TOE as well as the independent secure element to prevent single points of failure.

**Identification and Authentication**

The TOE includes several identification and authentication features. From the user's perspective, a Password Authentication Factor is required to unlock the device, with the exception of FCC-mandated functions, such as emergency calls, or non-sensitive actions like selecting a keyboard input method or taking screenshots. Even after the device is unlocked, the TOE mandates re-entering the password to modify the password itself. To enhance security, passwords are masked during input and are restricted by a configurable limit on failed attempts, after which the device will wipe its contents to safeguard data. Passwords can be composed of upper and lower-case letters, numbers, and special characters, with support for up to 16-character passwords.

The TOE also functions as an 802.1X supplicant, supporting the use and validation of X.509v3 certificates for secure exchanges via EAP-TLS, TLS, and HTTPS protocols.

**Security Management**

The TOE offers all necessary interfaces for managing the security functions outlined in this Security Target, along with additional functionalities commonly found in mobile devices. While many features are accessible to the end users, certain security and management functions are restricted to administrators who manage the device through a MDM solution once the TOE is enrolled. Upon unenrollment from the MDM, the TOE will automatically remove all Enterprise applications and clear MDM policies, ensuring the device returns to its original state.

**Protection of the TFS**

The TOE incorporates a range of mechanisms to safeguard its reliability and the integrity of its security features. It ensures that sensitive data, such as cryptographic keys, remain protected by preventing their access or export via the application processor’s hardware. Specifically, the TOE blocks all read access to the Root Encryption Key (REK) and confines all keys derived from the REK into the Trusted Execution Environment (TEE). Application software is only permitted to use REK-derived keys by reference and can only receive the results of operations involving these keys.

Additionally, the TOE provides a built-in timing mechanism to ensure accurate and reliable time data, which is essential for functions like audit log accountability. It enforces protections for memory pages (read, write, execute) and leverages security measures such as address space layout randomization and stack-based buffer overflow protections to mitigate the risk of exploiting application vulnerabilities. The TOE also safeguards itself from unauthorized modifications and isolates application memory spaces from each other to protect against unauthorized access by other applications.

The TOE includes self-test and integrity-checking functions for both software and firmware, enabling it to detect any failures or potential corruption. If a self-test fails, the TOE will not enter an operational state. The TOE also verifies the digital signatures of all new software or firmware images before installation, ensuring that updates do not introduce malicious or unintended changes. This digital signature verification process also applies to applications, requiring that all applications must have a valid signature before installation.

**TOE Access**

The TOE can be locked either manually by the user or automatically after a configured period of inactivity, both causing the display of the device to be obscured. Additionally, the TOE has the capability to display an administrator-defined advisory message banner upon the first unlock following a reboot, which can be configured using the MDM API of the TOE. The TOE is also capable of attempting to connect to wireless networks according to the configured settings.

**Trusted Path/Channels**

The TOE supports secure communication channels with trusted network devices through the use of IEEE 802.11-2012, 802.1X, EAP-TLS, TLS, and HTTPS protocols.

### 1.4.2 TOE Documentation

| Document | version | Unique identifier (sha256) |
|---|---|---|
| **End User Guide** | 1.1 | a33ba6d55b2d332b38d364dbdc1cc9b0601083cd1fb45683776a8f10db2f488a |
| **Administrative Guide** | 1.3 | 09599bf86a6872e3108fc0b8037798d87205738919f44ece05bdbd1bfbc2b35d |

## 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 2022, Revision 1, November 2022
  - CC Part2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 2022, Revision 1, November 2022
  - CC Part3 Extended
- PP-Configuration for Mobile Device Fundamentals, Bluetooth, and WLAN Clients, Version 1.0, 11 October 2022 (CFG_MDF-BT-WLANC_V1.0)
  - The PP-Configuration includes the following components:
    - Base-PP: Protection Profile for Mobile Device Fundamentals, Version 3.3, 12 September 2022 (PP_MDF_V3.3)
    - PP-Module: PP-Module for Bluetooth, Version 1.0, 15 April 2021 (MOD_BT_V1.0)
    - PP-Module: PP-Module for WLAN Clients, Version 1.0, 31 March 2022 (MOD_WLANC_V1.0)
- Technical Decisions (TD) as of December 19, 2024:

**Table 6. Technical Decisions**

| Module | TD ID | TD Description | Applied | Rationale |
|---|---|---|---|---|
| **PP_MDF_V3.3** | TD0724 | Format corrections for FAU_GEN.1 | Yes | |
| | TD0704 | Part 3 (Extended) in CC Conformance Claims | Yes | |
| | TD0689 | RFC Update in FIA_X509_EXT.1 | Yes | |
| | TD0677 | Correction to Symbol in FCS_RBG_EXT.1 Test EA | Yes | |
| **MOD_BT_V1.0** | TD0707 | Formatting corrections | Yes | |
| | TD0685 | BT missing multiple SFR-to- Obj mappings | Yes | |
| | TD0671 | PP-Module updated to allow for new PP and PP-Module Versions | Yes | |
| | TD0650 | Conformance claim sections updated to allow for | No | Conformance with MOD_VPNC_V2.3<br>not claimed |

| | | | | |
|---|---|---|---|---|
| | | MOD_VPNC_V2. 3 and 2.4 | | |
| | TD0645 | Bluetooth audit details | Yes | |
| | TD0640 | Handling BT devices that do not support encryption | Yes | |
| | TD0600 | Conformance claim sections updated to allow for MOD_VPNC_V2.3 | No | Conformance with MOD_VPNC_V2.3<br>not claimed |
| **MOD_WLANC_ V1.0** | TD0837 | Updates to allow-lists | Yes | |
| | TD0797 | Addition of FCS_WPA_EXT to ECD | Yes | |
| | TD0710 | WPA version restrictions | Yes | |
| | TD0703 | Removal of FIA_X509_EXT.2/ WLAN evaluation activities for revocation checking | Yes | |
| | TD0667 | Move Set Wireless Freq Band to Optional/Objective | Yes | |
| **PKG_TLS_V1.1** | TD0779 | Updated Session Resumption Support | No | 'TLS as a server' not selected<br>in FCS_TLS_EXT. 1.1 |
| | TD0770 | TLSS.2 connection with no client cert | No | 'TLS as a server' not selected<br>in FCS_TLS_EXT. 1.1 |
| | TD0726 | Corrections to (D)TLSS SFRs | No | 'TLS as a server' and 'DTLS as a server' not selected<br>in FCS_TLS_EXT. 1.1 |
| | TD0739 | PKG_TLS_V1.1 has 2 different publication dates | Yes | |
| | TD0513 | CA Certificate loading | Yes | |

| | | | | |
|---|---|---|---|---|
| | TD0499 | Testing with pinned certificates | Yes | Pinned certificates not supported |
| | TD0469 | Modification of test activity<br>for FCS_TLSS_EXT.1.1 test 4.1 | No | 'TLS as a server' not selected<br>in FCS_TLS_EXT. 1.1 |
| | TD0442 | Updated TLS Ciphersuites | Yes | |

## 2.1 Conformance Rationale

This ST conforms to the PP_MDF_V3.3/MOD_BT_V1.0/MOD_WLANC_V1.0/PKG_TLS_V1.1, with Use Case H.2 (Enterprise-owned device for specialized, high security use) selected from PP_MDF_V3.3. For clarity, this combined set will be referred to as MDF/BT/WLANC/TLS. As previously outlined, the security problem definition, security objectives, and security requirements have been extracted from the PP.

This Security Target claims conformance to CC:2022 and to a Protection Profile based on CC v3.1, as specified in (CCMC-2023-04-001 - Transition Policy to CC:2022 and CEM:2022).

The maintainer of the PPs for which this ST claims conformance has not stipulated transition details. This ST is prepared to pass the CC:2022 ASE class if it is to ignore the conformance claim to the CC v3.1 Protection Profile and treat the Security Target as stand-alone.

No points of conflict have been identified. Particularly:

- The changes in the involved components (both SFRs and SARs) have been analyzed and no needed adjustments have been identified. The writing changes are only minor that do not imply an impact on the assurance level. Some changes in the dependencies have also been identified but they do not imply any conflict due to the fact that either the dependencies are already solved, or they do not apply due to the particular definition of the security problem in the Protection Profiles, Modules and Packages.
- Extended components haven't been replaced by components in CC:2022 Part 2 because no direct mapping from CC:2022 Part 2 components to the defined extended components have been found.
- All the evaluation methods and activities specified in the used Protection Profiles, Modules and Packages, have been used as-is (see section 5.2 of this ST). It has been identified that there is no need for adaptation according to (CC:2022 Part 4) guidelines.

The level of assurance is not reduced by using this approximation.

# 3 Security Problem Description

## 3.1 Threats

Mobile devices are subject to the threats of traditional computer systems along with those entailed by their mobile nature. The threats considered in this document are those of network eavesdropping, network attacks, physical access, malicious or flawed applications, persistent presence, and backup as detailed in Table 7.

**Table 7. Security threats**

| **Module** | Threat Name | **Threat** |
|---|---|---|
| **PP_MDF_V3.3** | **T.NETWORK_EAVESDROP** | An attacker is positioned on a wireless communications channel or elsewhere on the network infrastructure. Attackers may monitor and gain access to data exchanged between the Mobile Device and other endpoints. |
| | **T.NETWORK_ATTACK** | 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 to compromise the Mobile Device. These attacks include malicious software update of any applications or system software on the device. These attacks also include malicious web pages or email attachments, which are usually delivered to devices over the network. |
| | **T.PHYSICAL_ACCESS** | 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 through direct and possibly destructive access to its storage media. Note: Defending against device re-use after physical compromise is out of scope for this Protection Profile. |
| | **T.MALICIOUS_APP** | Applications loaded onto the Mobile Device may include malicious or exploitable code. This code could be included intentionally or unknowingly by the developer, perhaps as part of a software library. Malicious apps may attempt to exfiltrate data to which they have access. They may also conduct attacks against the platform's system software, which will provide them with additional privileges and the ability to conduct further malicious activities. Malicious applications may be able to control the device's sensors (GPS, camera, microphone) to gather intelligence about the user's surroundings even when those activities do not involve data resident or transmitted from the device. Flawed applications may give an attacker access to perform network-based or physical attacks that otherwise would have been prevented. |
| | **T.PERSISTENT_PRESENCE** | 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 this case, the device and its data may be controlled by an adversary as well as by its legitimate owner. |
| **MOD_WLANC_V1.0** | **T.TSF_FAILURE** | 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** | 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 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** | 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

The specific conditions listed in Table 8 are assumed to exist in the TOE's Operational Environment. These include both practical realities in the development of the TOE security requirements and the essential environmental conditions on the use of the TOE.

**Table 8. Security assumptions**

| **Module** | Assumption Name | **Assumption** |
|---|---|---|
| **PP_MDF_V3.3** | **A.CONFIG** | 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** | It is assumed that the mobile user will immediately notify the administrator if the Mobile Device is lost or stolen. |
| | **A.PRECAUTION** | It is assumed that the mobile user exercises precautions to reduce the risk of loss or theft of the Mobile Device. |
| | **A.PROPER_USER** | Mobile Device users are not willfully negligent or hostile and use the device within compliance of a reasonable Enterprise security policy. |
| **MOD_WLANC_V1.0** | **A.NO_TOE_BYPASS** | Information cannot flow between the wireless client and the internal wired network without passing through the TOE. |
| | **A.TRUSTED_ADMIN** | TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner. |

## 3.3 Organizational Security Policies

There are no additional OSPs to be defined

# 4 Security Objectives

The security objectives defined are tailored for mobile devices and are therefore applicable to the KATIM X3M running KATIM OS 13, as the TOE.

## 4.1 Security Objectives for the TOE

Table 9 describes the relevant security objectives for the operational environment of the TOE.

**Table 9. Security objectives for the TOE**

| Module | Security Objective Name | Security Objective |
|---|---|---|
| **PP_MDF_V3.3** | **O.PROTECTED_COMMS** | To address the network eavesdropping (T.NETWORK_EAVESDROP) and network attack (T.NETWORK_ATTACK) threats described in Section 3.1 Threats, concerning wireless transmission of Enterprise and user data and configuration data between the TOE and remote network entities, conformant TOEs will use a trusted communication path. The TOE must be capable of communicating using mutually authenticated TLS, EAP-TLS, HTTPS, 802.1X, and 802.11-2012. The TOE may optionally communicate using these standard protocols: IPsec, mutually-authenticated DTLS, or Bluetooth. These 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.<br><br>While conformant TOEs must support all of the choices specified in the ST including any optional SFRs defined in this PP, they may support additional algorithms and protocols. If such additional mechanisms are not evaluated, guidance must be given to the administrator to make clear the fact that they were not evaluated. |
| | **O.STORAGE** | To address the issue of loss of confidentiality of user data in the event of loss of a Mobile Device (T.PHYSICAL_ACCESS), 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** | 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** | To address the issue of loss of confidentiality of user data in the event of loss of a Mobile Device (T.PHYSICAL_ACCESS), 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.<br>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.<br>Repeated attempts by a user to authorize to the TSF will be limited or throttled to enforce a delay between unsuccessful attempts. |
| | **O.INTEGRITY** | 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.<br>To address the issue of an application containing malicious or flawed code (T.MALICIOUS_APP), 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. |
| **MOD_WLANC_V1.0** | **O.AUTH_COMM** | 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** | The TOE will 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.SELF_TEST** | The TOE will provide the capability to test some subset of its security functionality to ensure it is operating properly. |
| | **O.SYSTEM_MONITORING** | The TOE will provide the capability to generate audit data. |
| | **O.TOE_ADMINISTRATION** | The TOE will provide mechanisms to allow administrators to be able to configure the TOE. |
| | **O.WIRELESS_ACCESS_POINT_CONNECTION** | 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

Table 10 describes the relevant security objectives for the operational environment of the TOE.

### Table 10. Security objectives for operational environment

| Module | Security Objective Name | Security Objective |
|---|---|---|
| **PP_MDF_V3.3** | **OE.CONFIG** | TOE administrators will configure the Mobile Device security functions correctly to create the intended security policy |
| | **OE.NOTIFY** | The Mobile User will immediately notify the administrator if the Mobile Device is lost or stolen. |
| | **OE.PRECAUTION** | The mobile device user exercises precautions to reduce the risk of loss or theft of the Mobile Device. |
| | **OE.DATA_PROPER_USER** | Administrators take measures to ensure that mobile device users are adequately vetted against malicious intent and are made aware of the expectations for appropriate use of the device. |
| **MOD_WLANC_V1.0** | **OE.NO_TOE_BYPASS** | Information cannot flow between external and internal networks located in different enclaves without passing through the TOE. |
| | **OE.TRUSTED_ADMIN** | TOE administrators are trusted to follow and apply all administrator guidance in a trusted manner. |

## 4.3 Security Objectives Rationale

Table 11 describes how the assumptions, threats, and organizational security policies map to the security objectives.

### Table 11. Security Objectives Rationale

| Module | Threat, Assumption, or OSP | Security Objectives | Rationale |
|---|---|---|---|
| **PP_MDF_V3.3** | **T.NETWORK_EAVESDROP** | **O.PROTECTED_COMMS** | The threat T.NETWORK_EAVESDROP is countered by O.PROTECTED_COMMS as this provides the capability to communicate using one (or more) standard protocols as a means to maintain the confidentiality of data that are transmitted outside of the TOE. |
| | | **O.CONFIG** | The threat T.NETWORK_EAVESDROP is countered by O.CONFIG as this provides a secure configuration of the mobile device to protect data that it processes. |
| | | **O.AUTH** | The threat T.NETWORK_EAVESDROP is countered by O.AUTH as this provides authentication of the endpoints of a trusted communication path. |
| | **T.NETWORK_ATTACK** | **O.PROTECTED_COMMS** | The threat T.NETWORK_ATTACK is countered by O.PROTECTED_COMMS as this provides the capability to communicate using one (or more) standard protocols as a means to maintain |

| | | | |
|---|---|---|---|
| | | | the confidentiality of data that are transmitted outside of the TOE. |
| | | **O.CONFIG** | The threat T.NETWORK_ATTACK is countered by O.CONFIG as this provides a secure configuration of the mobile device to protect data that it processes. |
| | | **O.AUTH** | The threat T.NETWORK_ATTACK is countered by O.AUTH as this provides authentication of the endpoints of a trusted communication path. |
| | **T.PHYSICAL_ACCESS** | **O.STORAGE** | The threat T.PHYSICAL_ACCESS is countered by O.STORAGE as this provides the capability to encrypt all user and enterprise data and authentication keys to ensure the confidentiality of data that it stores. |
| | | **O.AUTH** | The threat T.PHYSICAL_ACCESS is countered by O.AUTH as this provides the capability to authenticate the user prior to accessing protected functionality and data. |
| | **T.MALICIOUS_APP** | **O.PROTECTED_COMMS** | The threat T.MALICIOUS_APP is countered by O.PROTECTED_COMMS as this provides the capability to communicate using one (or more) standard protocols as a means to maintain the confidentiality of data that are transmitted outside of the TOE. |
| | | **O.CONFIG** | The threat T.MALICIOUS_APP is countered by O.CONFIG as this provides the capability to configure and apply security policies to ensure the Mobile Device can protect user and enterprise data that it may store or process. |
| | | **O.AUTH** | The threat T.MALICIOUS_APP is countered by O.AUTH as this provides the capability to authenticate the user and endpoints of a trusted path to ensure they are communicating with an authorized entity with appropriate privileges. |
| | | **O.INTEGRITY** | The threat T.MALICIOUS_APP is countered by O.INTEGRITY as this provides the capability to perform self-tests to ensure the integrity of critical functionality, software/firmware and data has been maintained. |
| | **T.PERSISTENT_PRESENCE** | **O.INTEGRITY** | The threat T.PERSISTENT_PRESENCE is countered by O.INTEGRITY as this provides the capability to perform self-tests to ensure the integrity of critical functionality, software/firmware and data has been maintained. |

| | | | |
|---|---|---|---|
| | **A.CONFIG** | **OE.CONFIG** | The operational environment objective OE.CONFIG is realized through A.CONFIG. |
| | **A.NOTIFY** | **OE.NOTIFY** | The operational environment objective OE.NOTIFY is realized through A.NOTIFY. |
| | **A.PRECAUTION** | **OE.PRECAUTION** | The operational environment objective OE.PRECAUTION is realized through A.PRECAUTION. |
| | **A.PROPER_USER** | **OE.DATA_PROPER_USER** | The operational environment objective OE.DATA_PROPER_USER is realized through A.PROPER_USER. |
| **MOD_WLANC_V1.0** | **T.TSF_FAILURE** | **O.SELF_TEST** | The threat T.TSF_FAILURE is mitigated by O.SELF_TEST as this defines a mechanism for ensuring the reliability of the TSF by detecting potential failure conditions. |
| | **T.UNAUTHORIZED_ACCESS** | **O.AUTH_COMM** | The threat T.UNAUTHORIZED_ACCESS is mitigated in part by O.AUTH_COMM by ensuring the authenticity of any remote endpoint that the TSF connects to. |
| | | **O.CRYPTOGRAPHIC_FUNCTIONS** | The threat T.UNAUTHORIZED_ACCESS is mitigated in part by O.CRYPTOGRAPHIC_FUNCTIONS by ensuring the confidentiality and integrity of data in transit to protect against man-in-the-middle attacks. |
| | | **O.TOE_ADMINISTRATION** | The threat T.UNAUTHORIZED_ACCESS is mitigated in part by O.TOE_ADMINISTRATION by using the TOE platform's authentication mechanism to ensure that only authorized administrators can configure the TOE's behavior. |
| | | **O.WIRELESS_ACCESS_POINT_CONNECTION** | The threat T.UNAUTHORIZED_ACCESS is mitigated in part by this objective because it provides a mechanism to restrict the remote entities that the TOE is permitted to communicate with. |
| | **T.UNDETECTED_ACTIONS** | **O.SYSTEM_MONITORING** | The threat T.UNDETECTED_ACTIONS is mitigated by O.SYSTEM_MONITORING by enforcing an auditing mechanism that can be used to track security-relevant TOE behavior. |
| | **A.NO_TOE_BYPASS** | **OE.NO_TOE_BYPASS** | The operational environment objective OE.NO_TOE_BYPASS is realized through A.NO_TOE_BYPASS. |
| | **A.TRUSTED_ADMIN** | **OE.TRUSTED_ADMIN** | The Operational Environment objective OE.TRUSTED ADMIN is realized through A.TRUSTED_ADMIN. |
| **MOD_BT_V1.0** | **T.NETWORK_EAVESDROP** | **O.PROTECTED_COMMS** | The threat T.NETWORK_EAVESDROP is countered by O.PROTECTED_COMMS as this provides the capability to communicate using Bluetooth as a |

| | | | |
|---|---|---|---|
| | | | means to maintain the confidentiality of data that are transmitted outside of the TOE. |
| | **T.NETWORK_ATTACK** | **O.PROTECTED_COMMS** | The threat T.NETWORK_ATTACK is countered by O.PROTECTED_COMMS as this provides the capability to communicate using Bluetooth as a means to maintain the confidentiality of data that are transmitted outside of the TOE. |

# 5 Extended Components Definition

All extended requirements in this ST originate from the MDF/BT/WLANC/TLS. These documents define the following extended requirements listed in Table 12, and since the requirements are not redefined within the ST, the MDF/BT/WLANC/TLS should be examined for additional details of the CC extensions.

**Table 12. Extended component requirements**

| Module | Extended Component ID | Extended Component Name |
|---|---|---|
| PP_MDF_V3.3 | FCS_CKM_EXT.1 | Cryptographic Key Support |
| | FCS_CKM_EXT.2 | Cryptographic Key Random Generation |
| | FCS_CKM_EXT.3 | Cryptographic Key Generation |
| | FCS_CKM_EXT.4 | Key Destruction |
| | FCS_CKM_EXT.5 | TSF Wipe |
| | FCS_CKM_EXT.6 | Salt Generation |
| | FCS_HTTPS_EXT.1 | HTTPS Protocol |
| | FCS_IV_EXT.1 | Initialization Vector Generation |
| | FCS_RBG_EXT.1 | Random Bit Generation |
| | FCS_SRV_EXT.1 | Cryptographic Algorithm Services |
| | FCS_STG_EXT.1 | Cryptographic Key Storage |
| | FCS_STG_EXT.2 | Encrypted Cryptographic Key Storage |
| | FCS_STG_EXT.3 | Integrity of Encrypted Key Storage |
| | FDP_ACF_EXT.1 | Security Access Control for System Services |
| | FDP_ACF_EXT.2 | Security Access Control for System Services |
| | FDP_DAR_EXT.1 | Protected Data Encryption |
| | FDP_DAR_EXT.2 | Sensitive Data Encryption |
| | FDP_IFC_EXT.1 | Subset Information Flow Control |
| | FDP_STG_EXT.1 | User Data Storage |
| | FDP_UPC_EXT.1/APPS | Inter-TSF User Data Transfer Protection (Applications) |
| | FDP_UPC_EXT.1/BLUETOOTH | Inter-TSF User Data Transfer Protection (Bluetooth) |
| | FIA_AFL_EXT.1 | Authentication Failure Handling |
| | FIA_PMG_EXT.1 | Password Management |
| | FIA_TRT_EXT.1 | Authentication Throttling |
| | FIA_UAU_EXT.1 | Authentication for Cryptographic Operation |
| | FIA_UAU_EXT.2 | Timing of Authentication |
| | FIA_X509_EXT.1 | X.509 Validation of Certificates |

| | | |
|---|---|---|
| | FIA_X509_EXT.2 | X.509 Certificate Authentication |
| | FIA_X509_EXT.3 | Request Validation of Certificates |
| | FMT_MOF_EXT.1 | Management of Security Functions Behavior |
| | FMT_SMF_EXT.2 | Specification of Remediation Actions |
| | FPT_AEX_EXT.1 | Application Address Space Layout Randomization |
| | FPT_AEX_EXT.2 | Memory Page Permissions |
| | FPT_AEX_EXT.3 | Stack Overflow Protection |
| | FPT_AEX_EXT.4 | Domain Isolation |
| | FPT_JTA_EXT.1 | JTAG Disablement |
| | FPT_KST_EXT.1 | Key Storage |
| | FPT_KST_EXT.2 | No Key Transmission |
| | FPT_KST_EXT.3 | No Plaintext Key Export |
| | FPT_NOT_EXT.1 | Self-Test Notification |
| | FPT_TST_EXT.1 | TSF Cryptographic Functionality Testing |
| | FPT_TST_EXT.2/PREKERNEL | TSF Integrity Checking (Pre-Kernel) |
| | FPT_TUD_EXT.1 | TSF Version Query |
| | FPT_TUD_EXT.2 | TSF Update Verification |
| | FPT_TUD_EXT.3 | Application Signing |
| | FTA_SSL_EXT.1 | TSF- and User-initiated Locked State |
| | FTP_ITC_EXT.1 | Trusted Channel Communication |
| | ALC_TSU_EXT.1 | Timely Security Updates |
| MOD_BT_V1.0 | FMT_SMF_EXT.1/BT | Specification of Management Functions |
| | FCS_CKM_EXT.8 | Bluetooth Key Generation |
| | FIA_BLT_EXT.1 | Bluetooth User Authorization |
| | FIA_BLT_EXT.2 | Bluetooth Mutual Authentication |
| | FIA_BLT_EXT.3 | Rejection of Duplicate Bluetooth Connections |
| | FIA_BLT_EXT.4 | Secure Simple Pairing |
| | FIA_BLT_EXT.6 | Trusted Bluetooth Device User Authorization |
| | FIA_BLT_EXT.7 | Untrusted Bluetooth Device User Authorization |
| | FTP_BLT_EXT.1 | Bluetooth Encryption |
| | FTP_BLT_EXT.2 | Persistence of Bluetooth Encryption |
| | FTP_BLT_EXT.3/BR | Bluetooth Encryption Parameters (BR/ EDR) |
| | FTP_BLT_EXT.3/LE | Bluetooth Encryption Parameters (LE) |
| | FCS_TLSC_EXT.1/WLAN | TLS Client Protocol (EAP-TLS for WLAN) |

| | | |
|---|---|---|
| MOD_WLANC_ V 1.0 | FCS_TLSC_EXT.2/WLAN | TLS Client Support for Supported Groups Extension (EAP-TLS for WLAN) |
| | FCS_WPA_EXT.1 | Supported WPA Versions |
| | FIA_PAE_EXT.1 | Port Access Entity Authentication |
| | FIA_X509_EXT.1/WLAN | X.509 Certificate Validation |
| | FIA_X509_EXT.2/WLAN | X.509 Certificate Authentication (EAP- TLS for WLAN) |
| | FIA_X509_EXT.6 | X.509 Certificate Storage and Management |
| | FMT_SMF.1/WLAN | Specification of Management Functions (WLAN Client) |
| | FPT_TST_EXT.3/WLAN | TSF Cryptographic Functionality Testing (WLAN Client) |
| | FTA_WSE_EXT.1 | Wireless Network Access |
| | FTP_ITC.1/WLAN | Trusted Channel Communication (Wireless LAN) |
| PKG_TLS_V1.1 | FCS_TLS_EXT.1 | TLS Protocol |
| | FCS_TLSC_EXT.1 | TLS Client Protocol |
| | FCS_TLSC_EXT.2 | TLS Client Support for Mutual Authentication |
| | FCS_TLSC_EXT.4 | TLS Client Support for Renegotiation |
| | FCS_TLSC_EXT.5 | TLS Client Support for Supported Groups Extension |

# 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 are from the MDF/BT/WLANC/TLS documents. The refinements and operations already performed in the MDF/BT/WLANC/TLS are not identified (e.g., highlighted) here, rather the requirements have been copied from the MDF/BT/WLANC/TLS and any residual operations have been completed herein. Of particular note, the MDF/BT/WLANC/TLS made a number of refinements and completed some of the SFR operations defined in the Common Criteria (CC) and that PP should be consulted to identify those changes if necessary.

The SARs are from the MDF/BT/WLANC/TLS documents, and includes all the relevant SARs. The SARs are effectively refined since requirement-specific 'Evaluation Activities' are defined in the MDF/BT/WLANC/TLS that serve to ensure corresponding evaluations will yield more practical and consistent assurance. The MDF/BT/WLANC/TLS should be consulted for the assurance activity definitions.

## 6.1 TOE Security Functional Requirements

The following table identifies the SFRs that are satisfied by KATIM X3M on KATIM OS 13 TOE:

**Table 13. TOE Security Functional Components**

| Requirement Class | PP | Requirement Component |
|---|---|---|
| FAU: Security Audit | PP_MDF_V3.3 | FAU_GEN.1 Audit Data Generation |
| | MOD_BT_V1.0 | FAU_GEN.1/BT Audit Data Generation (Bluetooth) |
| | MOD_WLANC_V1.0 | FAU_GEN.1/WLAN Audit Data Generation (Wireless LAN) |
| | PP_MDF_V3.3 | FAU_SAR.1 Audit Review |
| | PP_MDF_V3.3 | FAU_STG.1 Audit Storage Protection |
| | PP_MDF_V3.3 | FAU_STG.4 Prevention of Audit Data Loss |
| FCS:<br>Cryptographic Support | PP_MDF_V3.3 | FCS_CKM.1 Cryptographic Key Generation |
| | MOD_WLANC_V1.0 | FCS_CKM.1/WPA Cryptographic Key Generation (Symmetric Keys for WPA2/WPA3 Connections) |
| | PP_MDF_V3.3 | FCS_CKM.2/UNLOCKED Cryptographic Key Establishment |
| | PP_MDF_V3.3 | FCS_CKM.2/LOCKED Cryptographic Key Establishment |
| | MOD_WLANC_V1.0 | FCS_CKM.2/WLAN Cryptographic Key Distribution (Group Temporal Key for WLAN) |
| | PP_MDF_V3.3 | FCS_CKM_EXT.1 Cryptographic Key Support |
| | PP_MDF_V3.3 | FCS_CKM_EXT.2 Cryptographic Key Random Generation |
| | PP_MDF_V3.3 | FCS_CKM_EXT.3 Cryptographic Key Generation |
| | PP_MDF_V3.3 | FCS_CKM_EXT.4 Key Destruction |
| | PP_MDF_V3.3 | FCS_CKM_EXT.5 TSF Wipe |
| | PP_MDF_V3.3 | FCS_CKM_EXT.6 Salt Generation |
| | MOD_BT_V1.0 | FCS_CKM_EXT.8 Bluetooth Key Generation |

| | | |
|---|---|---|
| | PP_MDF_V3.3 | FCS_COP.1/ENCRYPT Cryptographic operation |
| | PP_MDF_V3.3 | FCS_COP.1/HASH Cryptographic operation |
| | PP_MDF_V3.3 | FCS_COP.1/SIGN Cryptographic operation |
| | PP_MDF_V3.3 | FCS_COP.1/KEYHMAC Cryptographic operation |
| | PP_MDF_V3.3 | FCS_COP.1/CONDITION Cryptographic operation |
| | PP_MDF_V3.3 | FCS_HTTPS_EXT.1 HTTPS Protocol |
| | PP_MDF_V3.3 | FCS_IV_EXT.1 Initialization Vector Generation |
| | PP_MDF_V3.3 | FCS_RBG_EXT.1 Random Bit Generation |
| | PP_MDF_V3.3 | FCS_SRV_EXT.1 Cryptographic Algorithm Services |
| | PP_MDF_V3.3 | FCS_STG_EXT.1 Cryptographic Key Storage |
| | PP_MDF_V3.3 | FCS_STG_EXT.2 Encrypted Cryptographic Key Storage |
| | PP_MDF_V3.3 | FCS_STG_EXT.3 Integrity of Encrypted Key Storage |
| | PKG_TLS_V1.1 | FCS_TLS_EXT.1 TLS Protocol |
| | PKG_TLS_V1.1 | FCS_TLSC_EXT.1 TLS Client Protocol |
| | MOD_WLANC_V1.0 | FCS_TLSC_EXT.1/WLAN TLS Client Protocol (EAP-TLS for WLAN) |
| | MOD_WLANC_V1.0 | FCS_TLSC_EXT.2/WLAN TLS Client Support for Supported Groups Extension (EAP-TLS for WLAN) |
| | PKG_TLS_V1.1 | FCS_TLSC_EXT.4 TLS Client Support for Renegotiation |
| | PKG_TLS_V1.1 | FCS_TLSC_EXT.2 TLS Client Support for Mutual Authentication |
| | PKG_TLS_V1.1 | FCS_TLSC_EXT.5 TLS Client Support for Supported Groups Extension (EAP-TLS for WLAN) |
| | MOD_WLANC_V1.0 | FCS_WPA_EXT.1 Supported WPA Versions |
| FDP: User Data Protection | PP_MDF_V3.3 | FDP_ACF_EXT.1 Security Access Control for System Services |
| | PP_MDF_V3.3 | FDP_ACF_EXT.2 Security Access Control for System Resources |
| | PP_MDF_V3.3 | FDP_DAR_EXT.1 Protected Data Encryption |
| | PP_MDF_V3.3 | FDP_DAR_EXT.2 Sensitive Data Encryption |
| | PP_MDF_V3.3 | FDP_IFC_EXT.1 Subset Information Flow Control |
| | PP_MDF_V3.3 | FDP_STG_EXT.1 User Data Storage |
| | PP_MDF_V3.3 | FDP_UPC_EXT.1/APPS Inter-TSF User Data Transfer Protection (Applications) |
| | PP_MDF_V3.3 | FDP_UPC_EXT.1/BLUETOOTH Inter-TSF User<br>Data Transfer Protection (Bluetooth) |
| FIA: Identification and Authentication | PP_MDF_V3.3 | FIA_AFL_EXT.1 Authentication Failure Handling |
| | MOD_BT_V1.0 | FIA_BLT_EXT.1 Bluetooth User Authorization |
| | MOD_BT_V1.0 | FIA_BLT_EXT.2 Bluetooth Mutual Authentication |
| | MOD_BT_V1.0 | FIA_BLT_EXT.3 Rejection of Duplicate Bluetooth Connections |

| | | |
|---|---|---|
| | MOD_BT_V1.0 | FIA_BLT_EXT.4 Secure Simple Pairing |
| | MOD_BT_V1.0 | FIA_BLT_EXT.6 Trusted Bluetooth Device User Authorization |
| | MOD_BT_V1.0 | FIA_BLT_EXT.7 Untrusted Bluetooth Device User Authorization |
| | MOD_WLANC_V1.0 | FIA_PAE_EXT.1 Port Access Entity Authentication |
| | PP_MDF_V3.3 | FIA_PMG_EXT.1 Password Management |
| | PP_MDF_V3.3 | FIA_TRT_EXT.1 Authentication Throttling |
| | PP_MDF_V3.3 | FIA_UAU.5 Multiple Authentication Mechanisms |
| | PP_MDF_V3.3 | FIA_UAU.6/CREDENTIAL Re-Authentication (Credential Change) |
| | PP_MDF_V3.3 | FIA_UAU.6/LOCKED Re-Authentication (TSF Lock) |
| | PP_MDF_V3.3 | FIA_UAU.7 Protected Authentication Feedback |
| | PP_MDF_V3.3 | FIA_UAU_EXT.1 Authentication for Cryptographic Operation |
| | PP_MDF_V3.3 | FIA_UAU_EXT.2 Timing of Authentication |
| | PP_MDF_V3.3 | FIA_X509_EXT.1 X.509 Validation of Certificates |
| | MOD_WLANC_V1.0 | FIA_X509_EXT.1/WLAN X.509 Certificate Validation |
| | PP_MDF_V3.3 | FIA_X509_EXT.2 X.509 Certificate Authentication |
| | MOD_WLANC_V1.0 | FIA_X509_EXT.2/WLAN X.509 Certificate Authentication (EAP-TLS for WLAN) |
| | PP_MDF_V3.3 | FIA_X509_EXT.3 Request Validation of Certificates |
| | MOD_WLANC_V1.0 | FIA_X509_EXT.6 X.509 Certificate Storage and Management |
| FMT: Security Management | PP_MDF_V3.3 | FMT_MOF_EXT.1 Management of Security Functions Behavior |
| | PP_MDF_V3.3 | FMT_SMF.1 Specification of Management Functions |
| | MOD_BT_V1.0 | FMT_SMF_EXT.1/BT Specification of Management Functions |
| | MOD_WLANC_V1.0 | FMT_SMF.1/WLAN Specification of Management Functions (WLAN Client) |
| | PP_MDF_V3.3 | FMT_SMF_EXT.2 Specification of Remediation Actions |
| FPT: Protection of the TSF | PP_MDF_V3.3 | FPT_AEX_EXT.1 Application Address Space Layout Randomization |
| | PP_MDF_V3.3 | FPT_AEX_EXT.2 Memory Page Permissions |
| | PP_MDF_V3.3 | FPT_AEX_EXT.3 Stack Overflow Protection |
| | PP_MDF_V3.3 | FPT_AEX_EXT.4 Domain Isolation |
| | PP_MDF_V3.3 | FPT_JTA_EXT.1 JTAG Disablement |
| | PP_MDF_V3.3 | FPT_KST_EXT.1 Key Storage |
| | PP_MDF_V3.3 | FPT_KST_EXT.2 No Key Transmission |
| | PP_MDF_V3.3 | FPT_KST_EXT.3 No Plaintext Key Export |
| | PP_MDF_V3.3 | FPT_NOT_EXT.1 Self-Test Notification |
| | PP_MDF_V3.3 | FPT_STM.1 Reliable time stamps |

| | | |
|---|---|---|
| | PP_MDF_V3.3 | FPT_TST_EXT.1 TSF Cryptographic Functionality Testing |
| | PP_MDF_V3.3 | FPT_TST_EXT.2/PREKERNEL TSF Integrity Checking (Pre-Kernel) |
| | MOD_WLANC_V1.0 | FPT_TST_EXT.3/WLAN TSF Cryptographic Functionality Testing (WLAN Client) |
| | PP_MDF_V3.3 | FPT_TUD_EXT.1 TSF Version Query |
| | PP_MDF_V3.3 | FPT_TUD_EXT.2 TSF Update Verification |
| | PP_MDF_V3.3 | FPT_TUD_EXT.3 Application Signing |
| FTA: TOE Access | PP_MDF_V3.3 | FTA_SSL_EXT.1 TSF- and User-initiated Locked State |
| | PP_MDF_V3.3 | FTA_TAB.1 Default TOE Access Banners |
| | MOD_WLANC_V1.0 | FTA_WSE_EXT.1 Wireless Network Access |
| FTP: Trusted Path/Channels | MOD_BT_V1.0 | FTP_BLT_EXT.1 Bluetooth Encryption |
| | MOD_BT_V1.0 | FTP_BLT_EXT.2 Persistence of Bluetooth Encryption |
| | MOD_BT_V1.0 | FTP_BLT_EXT.3/BR Bluetooth Encryption Parameters (BR/EDR) |
| | MOD_BT_V1.0 | FTP_BLT_EXT.3/LE Bluetooth Encryption Parameters (LE) |
| | PP_MDF_V3.3 | FTP_ITC_EXT.1 Trusted Channel Communication |
| | MOD_WLANC_V1.0 | FTP_ITC.1/WLAN Trusted Channel Communication (Wireless LAN) |

## 6.1.1 Security Audit (FAU)

### 6.1.1.1 PP_MDF_V3.3:FAU_GEN.1 Audit Data Generation

**FAU_GEN.1.1**

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

1. Start-up and shutdown of the audit functions
2. All auditable events for the [*not selected*] level of audit
3. *[All administrative actions*
4. *Start-up and shutdown of the OS*
5. *Specifically defined auditable events in Table 2 of the PP_MDF_V3.3*
6. [***no additional auditable events***]]

**Table 14. PP_MDF_V3.3 Audit Events**

| Requirement | Audit Event | Content |
|---|---|---|
| FCS_CKM.1 | [None] | |
| FCS_CKM_EXT.1 | [None] | |
| FCS_CKM_EXT.5 | [None] | |
| FCS_STG_EXT.1 | [None] | |
| FCS_STG_EXT.1 | Import or destruction of key | Identity of key, role and identity of requester |
| FCS_STG_EXT.3 | Failure to verify integrity of stored key | Identity of key being verified |

| | | |
|---|---|---|
| FDP_DAR_EXT.1 | [None] | |
| FDP_DAR_EXT.2 | [None] | |
| 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_NOT_EXT.1 | [None] | [No additional information] |
| FPT_TST_EXT.1 | Initiation of self-test | |
| FPT_TST_EXT.1 | Failure of self-test | [No additional information] |
| FPT_TST_EXT.2/ PREKERNEL | Start-up of TOE | |
| FPT_TST_EXT.2/ PREKERNEL | [None] | [No additional information] |

**FAU_GEN.1.2**

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

1. Date and time of the event
2. Type of event
3. Subject identity
4. The outcome (success or failure) of the event
5. Additional information in Table 2 of the PP_MD_V3.3
6. [***no additional information***]

#### 6.1.1.2 MOD_BT_V1.0:FAU_GEN.1/BT Audit Data Generation (Bluetooth)

**FAU_GEN.1.1/BT**

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. [*Specifically defined auditable events in the Auditable Events table (Table 2 of the MOD_BT_V1.0)*]

**Table 15. MOD_BT_V1.0 Audit Events**

| Requirement | Audit Event | Content |
|---|---|---|
| FIA_BLT_EXT.1 | Failed user authorization of Bluetooth device. | User authorization decision (e.g., user rejected connection, incorrect pin entry). |
| FIA_BLT_EXT.1 | Failed user authorization for local Bluetooth Service. | Bluetooth address and name of device. Bluetooth profile. Identity of local service with [service ID]. |
| FIA_BLT_EXT.2 | Initiation of Bluetooth connection. | Bluetooth address and name of device. |
| FIA_BLT_EXT.2 | Failure of Bluetooth connection. | Reason for failure. |

**FAU_GEN.1.2/BT**

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

1. Date and time of the event
2. Type of event
3. Subject identity
4. The outcome (success or failure) of the event

5. For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [*Additional information in the Auditable Events table (of the MOD_BT_V1.0)*]

### 6.1.1.3 MOD_ WLANC_V1.0:FAU_GEN.1/WLAN Audit Data Generation (Wireless LAN)

**FAU_GEN.1.1/WLAN**

The TSF shall [***invoke platform-provided functionality***] 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 auditable events for mandatory SFRs specified in Table 2 and selected SFRs in Table 5 (of the MOD_WLANC_V1.0)*]

## Table 16. MOD_WLANC_V1.0 Audit Events

| Requirement | Audit Event | Content |
|---|---|---|
| FCS_TLSC_EXT. 1/WLAN | Failure to establish an EAP-TLS session. | Reason for failure.<br>Non-TOE endpoint of connection. |
| FCS_TLSC_EXT. 1/WLAN | 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. |
| FIA_X509_EXT.6 | Attempts to load certificates. | |
| FIA_X509_EXT.6 | Attempts to revoke certificates. | |
| FPT_TST_EXT.3/ WLAN | Execution of this set of TSF self-tests. | |
| FPT_TST_EXT.3/ WLAN | [None] | [None] |
| FTA_WSE_EXT.1 | All attempts to connect to access points. | For each access point record the [*Certificate Check Message and the last [2] octets*] of the MAC Address<br>Success and failures (including reason for failure). |
| FTP_ITC.1/ WLAN | All attempts to establish a trusted channel. | Identification of the non-TOE endpoint of the channel. |

**FAU_GEN.1.2/WLAN**

The [***TOE platform***] shall record within each audit record at least the following information:

1. Date and time of the event, type of event, subject identity, (if relevant) the outcome (success or failure) of the event

2. For each audit event type, based on the auditable event definitions of the functional components included in the PP-Module/ST, [*Additional Audit Record Contents as specified in Table 2 and Table 5 (of the MOD_WLANC_V1.0)*]

### 6.1.1.4 PP_MDF_V3.3:FAU_ SAR.1 Audit Review

**FAU_SAR.1.1**

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

**FAU_SAR.1.2**

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

#### 6.1.1.5 PP_MDF_V3.3:FAU_ STG.1 Audit Storage Protection

**FAU_STG.1.1**

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

**FAU_STG.1.2**

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

#### 6.1.1.6 PP_MDF_V3.3:FAU_ STG. 4 Prevention of Audit Data Loss

**FAU_STG.4.1**

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

### 6.1.2 Cryptographic Support (FCS)

#### 6.1.2.1 PP_MDF_V3.3: FCS_CKM.1 - Cryptographic Key Generation

**FCS_CKM.1.1**

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

- ***RSA schemes using cryptographic key sizes of [3072-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, P-521] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4],***

].

#### 6.1.2.2 MOD_WLANC_V1.0:FCS_CKM.1/WPA - Cryptographic Key Generation (Symmetric Keys for WPA2/WPA3 Connections)

**FCS_CKM.1.1/WPA**

The TSF shall generate symmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm [***PRF-384 and [PRF-512, PRF-704] (as defined in IEEE 802.11-2012)***] and specified key sizes [***256 bits and [128 bits, 192 bits***]] using a Random Bit Generator as specified in FCS_RBG_EXT.1.

#### 6.1.2.3 PP_MDF_V3.3:FCS_CKM.2/UNLOCKED - Cryptographic Key Establishment

**FCS_CKM.2.1/UNLOCKED**

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

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

].

#### 6.1.2.4 PP_MDF_V3.3:FCS_CKM.2/LOCKED - Cryptographic Key Establishment

**FCS_CKM.2.1/LOCKED**

The TSF shall perform cryptographic key establishment in accordance with a specified cryptographic key establishment method: [ ***[RSA-based key establishment schemes] that meet the following: [NIST Special Publication800-56B, "Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography"]*** ] for the purposes of encrypting sensitive data received while the device is locked.

### 6.1.2.5 MOD_WLANC_V1.0:FCS_CKM.2/WLAN - Cryptographic Key Establishment Distribution (Group Temporal Key for 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 (as defined in RFC 3394) in an EAPOL-Key frame (as defined in IEEE 802.11-2012 for the packet format and timing considerations] **and does not expose the cryptographic keys**.

### 6.1.2.6 PP_MDF_V3.3:FCS_CKM_EXT.1 - Cryptographic Key Support

**FCS_CKM_EXT.1.1**

The TSF shall support [***immutable hardware***] REKs with a [***symmetric***] key of strength [***256 bits***].

**FCS_CKM_EXT.1.2**

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

**FCS_CKM_EXT.1.3**

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

### 6.1.2.7 PP_MDF_V3.3:FCS_CKM_EXT.2 – Cryptographic Key Random Generation

**FCS_CKM_EXT.2.1**

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

### 6.1.2.8 PP_MDF_V3.3:FCS_CKM_EXT. 3 - Cryptographic Key Generation

**FCS_CKM_EXT.3.1**

The TSF shall use [

- ***asymmetric KEKs of [128-bits] security strength***
- ***symmetric KEKs of [256-bit] security strength corresponding to at least the security strength of the keys encrypted by the KEK***

].

**FCS_CKM_EXT.3.2**

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

- Derive the KEK from a Password Authentication Factor according to FCS_COP.1.1/CONDITION and

[

- ***Generate the KEK using an RBG that meets this profile (as specified in FCS_RBG_EXT.1)***
- ***Generate the KEK using a key generation scheme that meets this profile (as specified in FCS_CKM.1)***
- ***Combine the KEK from other KEKs in a way that preserves the effective entropy of each factor by [concatenating the keys and using a KDF (as described in SP 800-108), concatenating the keys and using a KDF (as described in SP 800-56C), encrypting one key with another]***

].

### 6.1.2.9 PP_MDF_V3.3:FCS_CKM_EXT. 4 - Key Destruction

**FCS_CKM_EXT.4.1**

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

- By clearing the KEK encrypting the target key
- In accordance with the following rules
  - For volatile memory, the destruction shall be executed by a single direct overwrite [***consisting of zeros***].
  - For non-volatile EEPROM, the destruction shall be executed by a single direct overwrite consisting of a pseudo random pattern using the TSF's RBG (as specified in FCS_RBG_EXT.1), followed by a read-verify.
  - For non-volatile flash memory, that is not wear-leveled, the destruction shall be executed [***by a block erase that erases the reference to memory that stores data as well as the data itself***].
  - For non-volatile flash memory, that is wear-leveled, the destruction shall be executed [***by a block erase***].
  - For non-volatile memory other than EEPROM and flash, the destruction shall be executed by a single direct overwrite with a random pattern that is changed before each write.

**FCS_CKM_EXT.4.2**

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

### 6.1.2.10 PP_MDF_V3.3:FCS_CKM_EXT. 5 - TSF Wipe

**FCS_CKM_EXT.5.1**

The TSF shall wipe all protected data by [

- ***Cryptographically erasing the encrypted DEKs or the KEKs in non-volatile memory by following the requirements in FCS_CKM_EXT.4.1***
- ***Overwriting all PD according to the following rules:***
  - ***For EEPROM, the destruction shall be executed by a single direct overwrite consisting of a pseudo random pattern using the TSF's RBG (as specified in FCS_RBG_EXT.1, followed by a read-verify***.
  - ***For flash memory, that is not wear-leveled, the destruction shall be executed [by a block erase that erases the reference to memory that stores data as well as the data itself].***
  - ***For flash memory, that is wear-leveled, the destruction shall be executed [by a block erase].***
  - ***For non-volatile memory other than EEPROM and flash, the destruction shall be executed by a single direct overwrite with a random pattern that is changed before each write.***

].

**FCS_CKM_EXT.5.2**

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

### 6.1.2.11 PP_MDF_V3.3:FCS_CKM_EXT. 6 - Salt Generation

**FCS_CKM_EXT.6.1**

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

### 6.1.2.12 MOD_BT_V1.0:FCS_CKM_EXT.8 - Bluetooth Key Generation

**FCS_CKM_EXT.8.1**

The TSF shall generate public/private ECDH key pairs every [***time a connection between devices is established***].

### 6.1.2.13 PP_MDF_V3.3:FCS_COP.1/ENCRYPT - Cryptographic Operation

**FCS_COP.1.1/ENCRYPT**

The TSF shall perform [*encryption/decryption*] in accordance with a specified cryptographic algorithm: [

- AES-CBC (as defined in FIPS PUB 197, and NIST SP 800-38A) mode
- AES-CCMP (as defined in FIPS PUB 197, NIST SP 800-38C and IEEE 802.11-2012), and
- [
  - ***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**]*].

### 6.1.2.14 PP_MDF_V3.3:FCS_ COP.1/HASH - Cryptographic Operation

**FCS_COP.1.1/HASH**

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

### 6.1.2.15 PP_MDF_V3.3:FCS_ COP.1/SIGN - Cryptographic Operation

**FCS_COP.1.1/SIGN**

The TSF shall perform [*cryptographic signature services (generation and verification)*] in accordance with a specified cryptographic algorithm [

- ***[RSA schemes] using cryptographic key sizes of [2048-bit or greater] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 4]***
- ***[ECDSA schemes] using ["NIST curves" P-384 and [P-256, P-521]] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5]***

].

### 6.1.2.16 PP_MDF_V3.3:FCS_ COP.1/KEYHMAC - Cryptographic Operation

**FCS_COP.1.1/KEYHMAC**

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

### 6.1.2.17 PP_MDF_V3.3:FCS_ COP.1/CONDITION - Cryptographic Operation

**FCS_COP.1.1/CONDITION**

The TSF shall perform conditioning in accordance with a specified cryptographic algorithm HMAC-[***SHA-256, SHA-512***] using a salt, and [***[key stretching with scrypt]***] and output cryptographic key sizes [***256, 512***] that meet the following: [***no standard***].

### 6.1.2.18 PP_MDF_V3.3:FCS_ HTTPS_EXT.1 - HTTPS Protocol

**FCS_HTTPS_EXT.1.1**

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

**FCS_HTTPS_EXT.1.2**

The TSF shall implement HTTPS using TLS as defined in [*the Functional Package for Transport Layer Security (TLS), version 1.1*].

**FCS_HTTPS_EXT.1.3**

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

### 6.1.2.19 PP_MDF_V3.3:FCS_ IV_EXT.1 - Initialization Vector Generation

**FCS_IV_EXT.1.1**

The TSF shall generate IVs in accordance with [*Table11: References and IV Requirements for NIST-approved Cipher Modes*].

### 6.1.2.20 PP_MDF_V3.3:FCS_RBG_EXT.1 - Random Bit Generation

**FCS_RBG_EXT.1.1**

The TSF shall perform all deterministic random bit generation services in accordance with NIST Special Publication 800-90A using [***Hash_DRBG (any), 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.

### 6.1.2.21 PP_MDF_V3.3:FCS_SRV_EXT.1 - Cryptographic Algorithm Services

FCS_SRV_EXT.1.1

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

- All mandatory and [***selected algorithms***] in FCS_CKM.2/LOCKED
- The following algorithms in FCS_COP.1/ENCRYPT: AES-CBC, [***AES-GCM***]
- All selected algorithms in FCS_COP.1/SIGN
- All mandatory and selected algorithms in FCS_COP.1/HASH
- All mandatory and selected algorithms in FCS_COP.1/KEYHMAC
- [
    - ***All mandatory and [selected algorithms] in FCS_CKM.1***

  ]

].

### 6.1.2.22 PP_MDF_V3.3:FCS_STG_EXT.1 - Cryptographic Key Storage

FCS_STG_EXT.1.1

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

FCS_STG_EXT.1.2

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

FCS_STG_EXT.1.3

The TSF shall be capable of destroying keys or 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 or secret the use of the key or 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 or secret to request that the key or secret be destroyed. Exceptions may only be explicitly authorized by *[**a common application developer**]*.

### 6.1.2.23 PP_MDF_V3.3:FCS_STG_EXT.2 - Encrypted Cryptographic Key Storage

FCS_STG_EXT.2.1

The TSF shall encrypt all DEKs, KEKs, [**WPA2/WPA3 PSK, Bluetooth Keys**] and [**all software-based key storage**] by KEKs that are [

- ***Protected by the REK with [***
    - ***encryption by a KEK chaining from a REK***

- *encryption by a KEK that is derived from a REK*

  *]*

- *Protected by the REK and the password with [*
  - *encryption by a KEK chaining to a REK and the password-derived KEK*
  - *encryption by a KEK that is derived from a REK and the password-derived KEK*

  *]*

].

FCS_STG_EXT.2.2

DEKs, KEKs, [***WPA2/WPA3 PSK, Bluetooth Keys***] and [***all software-based key storage***] shall be encrypted using one of the following methods: [

- ***using a SP800-56B key establishment scheme***
- ***using AES in the [GCM mode]***

].

### 6.1.2.24 PP_MDF_V3.3:FCS_STG_EXT.3 - Integrity of Encrypted Key Storage

FCS_STG_EXT.3.1

The TSF shall protect the integrity of any encrypted DEKs and KEKs and [**long-term trusted channel key material**] by [

- *[GCM]* ***cipher mode for encryption according to FCS_STG_EXT.2***

].

FCS_STG_EXT.3.2

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

### 6.1.2.25 PKG_TLS_V1.1:FCS_TLS_EXT.1 - TLS Protocol

FCS_TLS_EXT.1

The product shall implement [***TLS as a client***]**.**

### 6.1.2.26 PKG_TLS_V1.1:FCS_TLSC_EXT.1 - TLS Client Protocol

FCS_TLSC_EXT.1.1

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

- ***TLS_RSA_WITH_AES_256_GCM_SHA384***
- ***TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256***
- ***TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384***

] and also supports functionality for [

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

].

FCS_TLSC_EXT.1.2

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

FCS_TLSC_EXT.1.3

The product shall not establish a trusted channel if the server certificate is invalid [***with no exceptions***].

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

**FCS_TLSC_EXT.2.1**

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

### 6.1.2.28 MOD_WLANC_V1.0:FCS_TLSC_EXT.1/WLAN - TLS Client Protocol (EAP-TLS for WLAN)

**FCS_TLSC_EXT.1.1/WLAN**

The TSF shall implement TLS 1.2 (RFC 5246) and [**no earlier TLS versions**] in support of the EAP-TLS protocol as specified in RFC 5216 supporting the following cipher suites: [

- ***TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256***
- ***TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384***
- ***TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA***
- ***TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA***
- ***TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA***
- ***TLS_RSA_WITH_AES_256_GCM_SHA384***
- ***TLS_RSA_WITH_AES_128_CBC_SHA***
- ***TLS_RSA_WITH_AES_256_CBC_SHA***

].

**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.

### 6.1.2.29 MOD_WLANC_V1.0:FCS_TLSC_EXT.2/WLAN TLS - Client Support for Supported Groups Extension (EAP-TLS for WLAN)

**FCS_TLSC_EXT.2.1/WLAN**

The TSF shall present the Supported Groups extension in the Client Hello with the following NIST curves: [***secp256r1, secp384r1***].

### 6.1.2.30 PKG_TLS_V1.1:FCS_TLSC_EXT.4 – TLS Client Support for Renegotiation

**FCS_TLSC_EXT.4.1**

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

### 6.1.2.31 PKG_TLS_V1.1: FCS_TLSC_EXT.5 TLS - Client Support for Supported Groups Extension

**FCS_TLSC_EXT.5.1**

The product shall present the Supported Groups Extension in the Client Hello with the supported groups [

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

].

#### 6.1.2.32 MOD_WLANC_V1.0: FCS_WPA_EXT.1 - Supported WPA Versions

**FCS_WPA_EXT.1.1**

The TSF shall support WPA3 and [***WPA2***] security type.

### 6.1.3 User Data Protection (FDP)

#### 6.1.3.1 PP_MDF_V3.3:FDP_ACF_EXT.1 - Security Access Control for System Services

**FDP_ACF_EXT.1.1**

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

**FDP_ACF_EXT.1.2**

The TSF shall provide an access control policy that prevents [***application, groups of applications***] from accessing [all] data stored by other [***application, groups of applications***]. Exceptions may only be explicitly authorized for such sharing by [***a common application developer (for sharing between applications)***].

#### 6.1.3.2 M PP_MDF_V3.3:FDP_ACF_EXT.2 Security Access Control for System Resources

**FDP_ACF_EXT.2.1**

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

#### 6.1.3.3 PP_MDF_V3.3:FDP_DAR_EXT.1 Protected Data Encryption

**FDP_DAR_EXT.1.1**

Encryption shall cover all protected data.

**FDP_DAR_EXT.1.2**

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

#### 6.1.3.4 PP_MDF_V3.3:FDP_DAR_EXT.2 Sensitive Data Encryption

**FDP_DAR_EXT.2.1**

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

**FDP_DAR_EXT.2.2**

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

**FDP_DAR_EXT.2.3**

The TSF shall encrypt any stored symmetric key and any stored private key of the asymmetric keys 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.

### 6.1.3.5 PP_MDF_V3.3:FDP_IFC_EXT.1 Subset Information Flow Control

**FDP_IFC_EXT.1.1**

The TSF shall [***provide an interface which allows a VPN client to protect all IP traffic using IPsec***] with the exception of IP traffic needed to manage the VPN connection,
and [***traffic needed to determine if the network connection has connectivity to the internet and responses to local ICMP echo requests on the local subnet***], when the VPN is enabled.

### 6.1.3.6 PP_MDF_V3.3:FDP_STG_EXT.1 User Data Storage

**FDP_STG_EXT.1.1**

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

### 6.1.3.7 PP_MDF_V3.3:FDP_UPC_EXT.1/APPS Inter-TSF User Data Transfer Protection (Applications)

**FDP_UPC_EXT.1.1/APPS**

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

- Mutually authenticated TLS as defined in the Functional Package for Transport Layer Security (TLS), version 1.1,
- HTTPS

and [

- ***no other protocol***

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

**FDP_UPC_EXT.1.2/APPS**

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

### 6.1.3.8 PP_MDF_V3.3:FDP_UPC_EXT.1/BLUETOOTH Inter-TSF User Data Transfer Protection (Bluetooth)

**FDP_UPC_EXT.1.1/BLUETOOTH**

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

- Bluetooth BR/EDR in accordance with the PP-Module for Bluetooth, version 1.0,

and [

- ***Bluetooth LE in accordance with the PP-Module for Bluetooth, version 1.0***

]] 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/BLUETOOTH**

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 PP_MDF_V3.3:FIA_AFL_EXT.1 Authentication Failure Handling

**FIA_AFL_EXT.1.1**

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

**FIA_AFL_EXT.1.2**

The TSF shall detect when a configurable positive integer within [**0 and 10**] 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 MOD_BT_V1.0:FIA_BLT_EXT.1 Bluetooth User Authorization

**FIA_BLT_EXT.1.1**

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

#### 6.1.4.3 MOD_BT_V1.0:FIA_BLT_EXT.2 Bluetooth Mutual Authentication

**FIA_BLT_EXT.2.1**

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

#### 6.1.4.4 MOD_BT_V1.0:FIA_BLT_EXT.3 Rejection of Duplicate Bluetooth Connections

**FIA_BLT_EXT.3.1**

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

#### 6.1.4.5 MOD_BT_V1.0:FIA_BLT_EXT.4 Secure Simple Pairing

**FIA_BLT_EXT.4.1**

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

**FIA_BLT_EXT.4.2**

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

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

**FIA_BLT_EXT.6.1**

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

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

**FIA_BLT_EXT.7.1**

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

### 6.1.4.8 MOD_WLANC_V1.0:FIA_PAE_EXT.1 Port Access Entity Authentication

**FIA_PAE_EXT.1.1**

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

### 6.1.4.9 PP_MDF_V3.3:FIA_PMG_EXT.1 Password Management

**FIA_PMG_EXT.1.1**

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

1. Passwords shall be able to be composed of any combination of [***upper and lower case letters***], numbers, and special characters: [ ***! @ # $ % ^ &amp; * ( ) + = _ / - ' " : ; , ? `~ \ | &lt; &gt; { } [ ]*** ]
2. Password length up to [**16**] characters shall be supported.

### 6.1.4.10 PP_MDF_V3.3:FIA_TRT_EXT.1 Authentication Throttling

**FIA_TRT_EXT.1.1**

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

### 6.1.4.11 PP_MDF_V3.3:FIA_ UAU.5 Multiple Authentication Mechanisms

**FIA_UAU.5.1**

The TSF shall provide password and to support user authentication.

**FIA_UAU.5.2**

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

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

- **User passwords are required after reboot to unlock the user's Credential encrypted (CE files) and keystore keys.**

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

- **The TOE verifies user credentials (password) via the gatekeeper, which compares the entered credential to a derived value or template.**

**To change protected settings or issue certain commands:**

- **The TOE requires password after a reboot, when changing settings (Screen lock settings), and when factory resetting**.

].

### 6.1.4.12 PP_MDF_V3.3:FIA_UAU.6/CREDENTIAL Re-Authentication (Credential Change)

**FIA_UAU.6.1/CREDENTIAL**

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 PP_MDF_V3.3:FIA_UAU.6/LOCKED Re-Authentication (TSF Lock)

**FIA_UAU.6.1/LOCKED**

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 PP_MDF_V3.3:FIA_UAU.7 Protected Authentication Feedback

**FIA_UAU.7.1**

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

### 6.1.4.15 PP_MDF_V3.3:FIA_UAU_EXT.1 Authentication for Cryptographic Operation

**FIA_UAU_EXT.1.1**

The TSF shall require the user to present the Password Authentication Factor prior to decryption of protected data and encrypted DEKs, KEKs and [***all software-based key storage***] at startup.

### 6.1.4.16 PP_MDF_V3.3:FIA_UAU_EXT.2 Timing of Authentication

**FIA_UAU_EXT.2.1**

The TSF shall allow [

- ***Power off/reboot***
- ***Change volume of the different streams***
- ***Take screenshots***
- ***Change display brightness***
- ***Toggle Wi-Fi (but no ability to change networks)***
- ***Toggle Bluetooth (but no ability to manage devices)***
- ***Turn shield mode on (but not off)***
- ***Toggle Do Not Disturb***
- ***Toggle flashlight***
- ***Toggle rotation lock***
- ***Toggle hotspot***
- ***Toggle battery saver***
- ***Toggle airplane mode***
- ***Toggle day/night theme***
- ***Toggle wireless power sharing***
- ***Toggle nightlight (screen blue filter for eye strain)***
- ***Toggle color inversion***
- ***Terminate / snooze alarms***
- ***Answer MT calls***
- ***Make emergency MO calls***

] 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 PP_MDF_V3.3:FIA_X509_EXT.1 X.509 Validation of Certificates

**FIA_X509_EXT.1.1**

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

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a certificate in the Trust Anchor Database.
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.
- The TSF shall validate that any CA certificate includes caSigning purpose in the key usage field.
- The TSF shall validate the revocation status of the certificate using [***OCSP as specified in RFC 6960***].
- The TSF shall validate the extendedKeyUsage field according to the following rules:
  - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing Purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.

  - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
  - Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) in the extendedKeyUsage field. [conditional]
  - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the extendedKeyUsage field.
  - OCSP certificates presented for OCSP responses shall have the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the extendedKeyUsage field. [conditional]

**FIA_X509_EXT.1.2**

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

### 6.1.4.18 MOD_WLANC_V1.0:FIA_X509_EXT.1/WLAN X.509 Certificate Validation

**FIA_X509_EXT.1.1/WLAN**

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

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

**FIA_X509_EXT.1.2/WLAN**

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

### 6.1.4.19 PP_MDF_V3.3:FIA_X509_EXT.2 X.509 Certificate Authentication

**FIA_X509_EXT.2.1**

The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [mutually authenticated TLS as defined in the Functional Package for Transport Layer Security, HTTPS, [***no other protocol***]] 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 [***not accept the certificate***].

### 6.1.4.20 MOD_WLANC_V1.0:FIA_X509_EXT.2/WLAN X.509 Certificate Authentication (EAP-TLS for 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*]].

### 6.1.4.21 PP_MDF_V3.3:FIA_X509_EXT.3 Request Validation of Certificates

**FIA_X509_EXT.3.1**

The TSF shall provide a certificate validation service to applications.

**FIA_X509_EXT.3.2**

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

#### 6.1.4.22 MOD_WLANC_V1.0:FIA_X509_EXT.6 Certificate Storage and Management

**FIA_X509_EXT.6.1**

The TSF shall [***invoke [software-based key storage] to store and protect***] certificate(s) from unauthorized deletion and modification.

**FIA_X509_EXT.6.2**

The TSF shall [***rely on [the TOE certificate management system] to load X.509v3 certificates into [software-based key storage]***] for use by the TSF.

### 6.1.5 Security Management (FMT)

#### 6.1.5.1 PP_MDF_V3.3:FMT_MOF_EXT.1 Management of Security Functions Behavior

**FMT_MOF_EXT.1.1**

The TSF shall restrict the ability to perform the functions in [column 4 of Table 17] to the user.

**FMT_MOF_EXT.1.2**

The TSF shall restrict the ability to perform the functions [in column 6 of Table 17] to the administrator when the device is enrolled and according to the administrator-configured policy.

#### 6.1.5.2 PP_MDF_V3.3:FMT_SMF.1 Specification of Management Functions

**FMT_SMF.1.1**

The TSF shall be capable of performing the following management functions (M = Mandatory, I = Implemented):

**Table 17. Security Management Functions**

| # | Management Function | Impl. | User Only | Admin | Admin Only |
|---|---|---|---|---|---|
| 1 | configure password policy:<br>• Minimum password length<br>• Minimum password complexity<br>• Maximum password lifetime<br>The administrator can configure the required password characteristics (minimum length, complexity, and lifetime) using the KATIM OS MDM APIs.<br>Length: an integer value of characters<br>Complexity: Unspecified, Something, Numeric, Alphabetic, Alphanumeric, Complex.<br>Lifetime: an integer value of seconds (0 = no maximum) | M | | M | M |
| 2 | configure session locking policy:<br>• Screen-lock enabled/disabled<br>• Screen lock timeout<br>• Number of authentication failures<br>The administrator can configure the session locking policy using the KATIM OS MDM APIs.<br>Screen lock timeout: an integer number of milliseconds before the TOE locks. | M | | M | M |

| | | | | | |
|---|---|---|---|---|---|
| | An integer number (0 to 9,223,372,036,854,775,807 [negative integers and zero means no lockout]).<br>Authentication failures: an integer number (-2,147,483,648 to 2,147,483,648[negative integers and zero means no limit]) of failures before a wipe action is initiated. This only applies to password authentication, fingerprint attempts do not increment this counter | | | | |
| 3 | enable/disable the VPN protection:<br>• Across device<br>• [***no other method***]<br>Both users (using the TOE's settings UI) and administrator (using the TOE's MDM APIs) can configure a third-party VPN client and then enable the VPN client to protect traffic. The User can set up VPN protection, but if an admin enables VPN protection, the user cannot disable it. | M | | | I |
| 4 | enable/disable [NFC, Bluetooth, WiFi] | M | | I | I |
| | enable/disable [cellular]<br>The administrator (using the TOE's MDM APIs) can manage NFC, Bluetooth, and WiFi radios. In that case, the user cannot override the administrator setting.<br>The cellular radio can be disabled by the administrator (using the TOE's MDM APIs), but the user (using the TOE's settings UI) is able to override this and turn the radio back on.<br>The TOE's radios operate at frequencies of 2.4 GHz (NFC/Bluetooth), 2.4/5 GHz (Wi-Fi), and 850, 900, 1800, 1900 MHz (4G/LTE). The radios are initialized during the initial power-up sequence. If the radio is supposed to be off (by setting), it will be turned off after the initial check. | M | I | | |
| 5 | enable/disable [microphone, camera]:<br>• Across device,<br>• [on a per-app basis]<br>An administrator can enable/disable the device's microphone via an MDM API. Once the microphone has been disabled, the user cannot re-enable it until the administrator enables it.<br>In the user's settings, a user can view a permission by type (i.e. camera, microphone). The user can access this by going to the settings UI (Settings > Privacy > Permission manager > <camera/ microphone>) and revoking any applications. | M | | I | I |
| 6 | transition to the locked state<br>Both the user and administrators (using the TOE's MDM APIs) can transition the TOE into a locked state. | M | | M | |
| 7 | TSF wipe of protected data<br>Both users (using the TOE's settings UI) and administrators (using the TOE's MDM APIs) can force the TOE to perform a full wipe (factory reset) of data. | M | | M | |

| | | | | | |
|---|---|---|---|---|---|
| 8 | configure application installation policy by: [<br>• restricting the sources of applications,<br>• denying installation of applications]<br>The administrator (using the TOE’s MDM APIs) can configure the TOE so that applications cannot be installed and can also block the use of the Google Play Store. | M | | M | M |
| 9 | import keys or secrets into the secure key storage<br>Both users and administrators (using the TOE’s MDM APIs) can import secret keys into the secure key storage. | M | | I | |
| 10 | destroy imported keys or secrets and [no other keys or secrets] in the secure key storage<br>Both users and administrators (using the TOE’s MDM APIs) can destroy secret keys in the secure key storage. | M | | I | |
| 11 | import X.509v3 certificates into the Trust Anchor Database<br>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 [no other X.509v3 certificates] in the Trust Anchor Database<br>Both users and administrators (using the MDM APIs) can remove imported X.509v3 certificates from the Trust Anchor Database as well as disable any of the TOE’s default Root CA certificates (in the latter case, the CA certificate still resides in the TOE’s read-only system partition; however, the TOE will treat that Root CA certificate and any certificate chaining to it as untrusted). | M | | I | |
| 13 | enroll the TOE in management<br>TOE users can enroll the TOE in management according to the instructions specific to a given MDM. Presumably any enrollment would involve at least some user functions (e.g., install an MDM agent application) on the TOE prior to enrollment. | M | | | |
| 14 | remove applications<br>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<br>Users can check for updates and cause the device to update if an update is available. An administrator can use MDM APIs to query the version of the TOE and query the installed applications and an MDM agent on the TOE could issue pop-ups, initiate updates, block communication, etc. until any necessary updates are completed | M | | M | I |
| 16 | install applications<br>Both users and administrators (using the TOE’s MDM APIs) can install applications on the TOE. | M | | M | |

| | | | | | |
|---|---|---|---|---|---|
| 17 | remove Enterprise applications<br>An administrator (using the TOE’s MDM APIs) can uninstall Enterprise installed applications on the TOE. | M | | M | |
| 18 | enable/disable display notification in the locked state of: [<br>• all notifications]<br>Notifications can be configured to display in the following formats:<br>• Users &amp; administrators: show all notification content<br>• Users: hide sensitive content<br>• Users & administrators: hide notifications entirely<br>If the administrator sets any of the above settings, the user cannot change it. | M | | I | I |
| 19 | enable data-at rest protection<br>The TOE always encrypts its user data storage | M | | | |
| 20 | enable removable media’s data-at-rest protection<br>The device does not support removable media. | | | | |
| 21 | enable/disable location services:<br>• Across device<br>• [no other method]<br>The administrator (using the TOE’s MDM APIs) can enable or disable location services.<br>An additional MDM API can prohibit TOE users’ ability to enable and disable location services. | M | | I | I |
| 22 | enable/disable the use of [***Biometric Authentication Factor***] | I | | I | |
| 23 | configure whether to allow or<br>disallow establishment of [assignment: configurable trusted channel in FTP_ITC_EXT.1.1 or FDP_UPC_EXT.1.1/APPS] if the peer or server certificate is deemed invalid. | | | | |
| 24 | enable/disable all data signaling over [assignment: list of externally accessible hardware ports] | | | | |
| 25 | enable/disable [Bluetooth tethering]<br>The administrator (using the TOE’s MDM APIs) can enable/disable all tethering methods (i.e. all or none disabled).<br>The TOE acts as a server (acting as an access point, a USB Ethernet adapter, and as a Bluetooth Ethernet adapter respectively) in order to share its network connection with another device. | I | | I | I |

| | | | | | |
|---|---|---|---|---|---|
| 26 | enable/disable developer modes<br>The administrator (using the TOE's MDM APIs) can disable Developer Mode.<br>Unless disabled by the administrator, TOE users can enable and disable Developer Mode. | I | | I | I |
| 27 | enable/disable bypass of local user authentication<br>N/A – It is not possible to bypass local user auth for this TOE | | | | |
| 28 | wipe Enterprise data<br>An administrator (using the TOE's MDM APIs) can remove Enterprise applications and their data. | I | | I | |
| 29 | approve [selection: import, removal] by applications of X.509v3 certificates in the Trust Anchor Database | | | | |
| 30 | configure whether to allow or disallow establishment of a trusted channel if the<br>TSF cannot establish a connection to determine the validity of a certificate | | | | |
| 31 | enable/disable the cellular protocols used to connect to cellular network base stations | | | | |
| 32 | read audit logs kept by the TSF | I | | | I |
| 33 | configure [selection: certificate, public-key] used to validate digital signature on applications | | | | |
| 34 | approve exceptions for shared use of keys or secrets by multiple applications | | | | |
| 35 | approve exceptions for destruction of keys or secrets by applications that did not import the key or secret | | | | |
| 36 | configure the unlock banner<br>The administrator (using the TOE's MDM APIs) can specify text to always be shown on the lock screen. | I | | I | |
| 37 | configure the auditable items | | | | |
| 38 | retrieve TSF-software integrity verification values | | | | |
| 39 | enable/disable [selection:<br>• USB mass storage mode<br>• USB data transfer without user authentication<br>• USB data transfer without authentication of the connecting system<br>] | | | | |
| 40 | enable/disable backup of<br>[selection: all applications, selected applications, selected groups of applications, configuration data] to [selection: locally connected system, remote system] | | | | |

| | | | | | |
|---|---|---|---|---|---|
| 41 | enable/disable [<br>• Hotspot functionality authenticated by [pre- shared key],<br>• USB tethering authenticated by [no authentication]]<br>The administrator (using the TOE's MDM APIs) can disable the Wi-Fi hotspot and USB tethering.<br>Unless disabled by the administrator, TOE users can configure the Wi-Fi hotspot with a pre- shared key and can configure USB tethering (with no authentication, though the device must be unlocked to establish the initial tethering connection). | I | | I | I |
| 42 | approve exceptions for sharing data between [selection: applications, groups of applications] | | | | |
| 43 | place applications into application process groups based on [assignment: enterprise configuration settings] | | | | |
| 44 | unenroll the TOE from management | | | | |
| 45 | enable/disable the Always On VPN protection<br>• Across device<br>• [no other method]<br>The administrator (using the TOE's MDM APIs) can specify whether a VPN connection is required for the device to access any network services. The configuration would specify the VPN connection(s) required. | I | | I | I |
| 46 | revoke Biometric template | | | | |
| 47 | [assignment: list of other management functions to be provided by the TSF] | | | | |

### 6.1.5.3 MOD_BT_V1.0:FMT_SMF_EXT.1/BT Specification of Management Functions

**FMT_SMF_EXT.1.1/BT**

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

## Table 18. Bluetooth Security Management Functions

| # | Management Function | Impl. | User Only | Admi n | Admi n Only |
|---|---|---|---|---|---|
| BT-1 | Configure the Bluetooth trusted channel.<br>• Disable/enable the Discoverable (for BR/EDR) and Advertising (for LE) modes; | M | | | |
| BT-2 | Change the Bluetooth device name (separately for BR/EDR and LE); | | | | |
| BT-3 | Provide separate controls for turning the BR/EDR and LE radios on and off; | | | | |

| | | | | | |
|---|---|---|---|---|---|
| BT-4 | Allow/disallow the following additional wireless technologies to be used with Bluetooth: [selection: Wi-Fi, NFC, [assignment: other wireless technologies]]; | | | | |
| BT-5 | Configure allowable methods of Out of Band pairing (for BR/EDR and LE); | | | | |
| BT-6 | Disable/enable the Discoverable (for BR/EDR) and Advertising (for LE) modes separately; | | | | |
| BT-7 | Disable/enable the Connectable mode (for BR/EDR and LE); | | | | |
| BT-8 | Disable/enable the Bluetooth [assignment: list of Bluetooth service and/or profiles available on the OS (for BR/EDR and LE)]: | | | | |
| BT-9 | Specify minimum level of security for each pairing (for BR/EDR and LE); | | | | |

### 6.1.5.4 MOD_WLANC_V1.0:FMT_SMF.1/WLAN Specification of Management Functions (WLAN Client)

**FMT_SMF.1.1/WLAN**

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

## Table 19. WLAN Security Management Functions

| # | Management Function | Impl. | User Only | Admin | Admin Only |
|---|---|---|---|---|---|
| WL-1 | Configure security policy for each wireless network:<br>• [***specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s)***]<br>• security type<br>• authentication protocol<br>• client credentials to be used for authentication | M | | M | |
| WL-2 | Specify wireless networks (SSIDs) to which the TSF may connect.<br>An administrator can specify a list of wireless networks to which the TOE may connect and can restrict the TOE to only allow a connection to the specified networks. | M | | M | |
| WL-3 | Enable/disable wireless network bridging capability (for example, bridging a connection between the WLAN and cellular radios to function as a hotspot) authenticated by [***pre-shared key***] | M | | M | |
| WL-4 | Enable/disable certificate revocation list checking; | | | | |
| WL-5 | Disable ad hoc wireless client-to-client connection capability | | | | |
| WL-6 | Disable roaming capability; | | | | |
| WL-7 | Enable/disable IEEE 802.1X pre-authentication; | | | | |
| WL-8 | Loading X.509 certificates into the TOE | | | | |

| | | | | | |
|---|---|---|---|---|---|
| WL-9 | Revoke X.509 certificates loaded into the TOE | | | | |
| WL-10 | Enable/disable and configure PMK caching:<br>• set the amount of time (in minutes) PMK entries are cached.<br>• set the maximum number of PMK entries that can be cached. | | | | |
| WL-11 | Configure security policy for each wireless network: set wireless frequency band to [selection: 2.4 GHz, 5 GHz, 6 GHz] | | | | |

#### 6.1.5.5 PP_MDF_V3.3:FMT_SMF_EXT.2 Specification of Remediation Actions

**FMT_SMF_EXT.2.1**

The TSF shall offer [***wipe of protected data, wipe of sensitive data, remove Enterprise applications, remove all device-stored Enterprise resource data***] upon un-enrollment and [***factory reset***].

### 6.1.6 Protection of the TSF (FPT)

#### 6.1.6.1 FPT_AEX_EXT.1 Application Address Space Layout Randomization

**FPT_AEX_EXT.1.1**

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

**FPT_AEX_EXT.1.2**

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

#### 6.1.6.2 FPT_AEX_EXT.2 Memory Page Permissions

**FPT_AEX_EXT.2.1**

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

#### 6.1.6.3 FPT_AEX_EXT.3 Stack Overflow Protection

**FPT_AEX_EXT.3.1**

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

#### 6.1.6.4 FPT_AEX_EXT.4 Domain Isolation

**FPT_AEX_EXT.4.1**

The TSF shall protect itself from modification by untrusted subjects.

**FPT_AEX_EXT.4.2**

The TSF shall enforce isolation of address space between applications.

#### 6.1.6.5 FPT_JTA_EXT.1 JTAG Disablement

**FPT_JTA_EXT.1.1**

The TSF shall [***disable access through hardware***] to JTAG.

### 6.1.6.6 PP_MDF_V3.3:FPT_KST_EXT.1 Key Storage

**FPT_KST_EXT.1.1**

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

### 6.1.6.7 PP_MDF_V3.3:FPT_KST_EXT.2 No Key Transmission

**FPT_KST_EXT.2.1**

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

### 6.1.6.8 FPT_KST_EXT.3 No Plaintext Key Export

**FPT_KST_EXT.3.1**

The TSF shall ensure it is not possible for the TOE users to export plaintext keys.

### 6.1.6.9 FPT_NOT_EXT.1 Self-Test Notification

**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-tests
- TSF software integrity verification failures
- [***no other failures***]

### 6.1.6.10 FPT_STM.1 Reliable time stamps

**FPT_STM.1.1**

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

### 6.1.6.11 FPT_TST_EXT.1 TSF Cryptographic Functionality Testing

**FPT_TST_EXT.1.1**

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

### 6.1.6.12 FPT_TST_EXT.2/PREKERNEL TSF Integrity Checking (Pre-Kernel)

**FPT_TST_EXT.2.1/PREKERNEL**

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

### 6.1.6.13 MOD_WLANC_V1.0:FPT_TST_EXT.3/WLAN TSF Cryptographic Functionality Testing (WLAN Client)

**FPT_TST_EXT.3.1/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.3.2/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.

### 6.1.6.14 FPT_TUD_EXT.1 TSF Version Query

**FPT_TUD_EXT.1.1**

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

**FPT_TUD_EXT.1.2**

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

**FPT_TUD_EXT.1.3**

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

#### 6.1.6.15 FPT_TUD_EXT.2 TSF Update Verification

**FPT_TUD_EXT.2.1**

The TSF shall verify software updates to the Application Processor system software and [[***baseband processor***]] 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***].

#### 6.1.6.16 FPT_TUD_EXT.3 Application Signing

**FPT_TUD_EXT.3.1**

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

### 6.1.7 TOE Access (FTA)

#### 6.1.7.1 PP_MDF_V3.3:FTA_SSL_EXT.1 TSF- and User-initiated Locked State

**FTA_SSL_EXT.1.1**

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

**FTA_SSL_EXT.1.2**

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

**FTA_SSL_EXT.1.3**

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

- Clearing or overwriting display devices, obscuring the previous contents;
- [**no other actions**].

#### 6.1.7.2 PP_MDF_V3.3:FTA_TAB.1 Default TOE Access Banners

**FTA_TAB.1.1**

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

#### 6.1.7.3 MOD_WLANC_V1.0:FTA_WSE_EXT.1 Wireless Network Access

**FTA_WSE_EXT.1.1**

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

## 6.1.8 Trusted Path/Channels (FTP)

### 6.1.8.1 MOD_BT_V1.0:FTP_BLT_EXT.1 Bluetooth Encryption

**FTP_BLT_EXT.1.1**

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

**FTP_BLT_EXT.1.2**

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

### 6.1.8.2 MOD_BT_V1.0:FTP_BLT_EXT.2 Persistence of Bluetooth Encryption

**FTP_BLT_EXT.2.1**

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

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

**FTP_BLT_EXT.3.1/BR**

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

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

**FTP_BLT_EXT.3.1/LE**

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

### 6.1.8.5 PP_MDF_V3.3:FTP_ITC_EXT.1 Trusted Channel Communication

**FTP_ITC_EXT.1.1**

The TSF shall use

- 802.11-2012 in accordance with the [PP-Module for Wireless LAN Clients, version 1.0],
- 802.1X in accordance with the [PP-Module for Wireless LAN Clients, version 1.0],
- EAP-TLS in accordance with the [PP-Module for Wireless LAN Clients, version 1.0],
- mutually authenticated TLS in accordance with [the Functional Package for Transport Layer Security (TLS), version 1.1]
- and [***HTTPS***] protocols to provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels, provides assured identification of its end points, protects channel data from disclosure, and detects modification of the channel data.

**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 [***OTA updates***].

### 6.1.8.6 MOD_WLANC_V1.0:FTP_ITC.1/WLAN Trusted Channel Communication (Wireless LAN)

**FTP_ITC.1.1/WLAN**

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

**FTP_ITC.1.2/WLAN**

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

**FTP_ITC.1.3/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.

### 6.2.1 Development (ADV)

#### 6.2.1.1 ADV_FSP.1 Basic Functional Specification

**ADV_FSP.1.1D**

The developer shall provide a functional specification.

**ADV_FSP.1.2D**

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

**ADV_FSP.1.1C**

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

**ADV_FSP.1.2C**

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

**ADV_FSP.1.3C**

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

**ADV_FSP.1.4C**

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

**ADV_FSP.1.1E**

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

**ADV_FSP.1.2E**

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

### 6.2.2 Guidance Documents (AGD)

#### 6.2.2.1 AGD_OPE.1 Operational User Guidance

**AGD_OPE.1.1D**

The developer shall provide operational user guidance.

**AGD_OPE.1.1C**

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

**AGD_OPE.1.2C**

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

**AGD_OPE.1.3C**

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

**AGD_OPE.1.4C**

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

**AGD_OPE.1.5C**

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

**AGD_OPE.1.6C**

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

**AGD_OPE.1.7C**

The operational user guidance shall be clear and reasonable.

**AGD_OPE.1.1E**

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

#### 6.2.2.2 AGD_PRE.1 Preparative Procedures

**AGD_PRE.1.1D**

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

**AGD_PRE.1.1C**

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

**AGD_PRE.1.2C**

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

**AGD_PRE.1.1E**

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

**AGD_PRE.1.2E**

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

### 6.2.3 Life-cycle support (ALC)

#### 6.2.3.1 ALC_CMC.1 Labeling of the TOE

**ALC_CMC.1.1D**

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

**ALC_CMC.1.1C**

The TOE shall be labeled with its unique reference.

**ALC_CMC.1.1E**

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

#### 6.2.3.2 ALC_CMS.1 TOE CM Coverage

**ALC_CMS.1.1D**

The developer shall provide a configuration list for the TOE.

**ALC_CMS.1.1C**

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

**ALC_CMS.1.2C**

The configuration list shall uniquely identify the configuration items.

**ALC_CMS.1.1E**

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

#### 6.2.3.3 ALC_TSU_EXT.1 Timely Security Updates

**ALC_TSU_EXT.1.1D**

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

**ALC_TSU_EXT.1.1C**

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

**ALC_TSU_EXT.1.2C**

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

**ALC_TSU_EXT.1.3C**

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

**ALC_TSU_EXT.1.4C**

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

**ALC_TSU_EXT.1.1E**

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

## 6.2.4 Tests (ATE)

### 6.2.4.1 ATE_IND.1 Independent Testing - Conformance

**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 AVA_VAN.1 Vulnerability Survey

**AVA_VAN.1.1D**

The developer shall provide the TOE for testing.

**AVA_VAN.1.1C**

The TOE shall be suitable for testing.

**AVA_VAN.1.1E**

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

**AVA_VAN.1.2E**

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

**AVA_VAN.1.3E**

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

# 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

### 7.1.1 PP_MDF_V3.3:FAU_GEN.1, MOD_BT_V1.0:FAU_GEN.1/BT, MOD_WLANC_V1.0:FAU_GEN.1/WLAN

The TOE employs two AOSP logging mechanisms to fulfill the logging requirements outlined in Table 2 of PP_MDF_V3.3, Table 2 of the MOD_BT_V1.0, and Table 2 of the MOD_WLANC_V1.0: SecurityLog events and Logcat events:

1. SecurityLog Events:

The complete list of auditable events for SecurityLog is available at https://developer.android.com/reference/android/app/admin/SecurityLog. These constants represent SecurityLog keywords used for logging, accompanied by descriptions of the events and any associated data/variables included in the audit record.

Additional event details retrievable through MDM requests can be found at: SecurityLog.SecurityEvent.

A log entry contains a keyword describing the event, the event's timestamp (date and time), and event-specific values indicating success, failure, and other relevant information.

2. Logcat Events:

Logcat events also include date, time, and event-specific values. Additionally, they provide a user ID to identify the individual responsible for generating the event.

Logcat entries are human-readable and do not require administrators to interpret the log structure.

Certain audit records, although logged, may not be accessible to the administrator due to specific conditions. The audit events and corresponding reasons for inaccessibility are outlined below:

**FAU_GEN.1** – Shutdown of Audit Functions: When the audit log functionality is terminated, the associated security log buffer is deallocated, making the audit records unavailable for review by the administrator.

**PP_MDF_V3.3: FAU_GEN.1** – Shutdown of the OS: Security logs are stored in volatile memory. When the system undergoes a shutdown, the audit record indicating system shutdown is cleared, making it inaccessible.

**PP_MDF_V3.3: FPT_TST_EXT.1** – Failure of Self-Test: Self-tests are executed prior to the initialization of audit logging. Although a log indicating the success or failure of the self-test is queued for logging, it will only be recorded once the security log system is fully initialized.

### 7.1.2 PP_MDF_V3.3:FAU_SAR.1

The TOE provides a Mobile Device Management (MDM) API that enables a Device-Owner MDM agent to access and read security logs.

### 7.1.3 PP_MDF_V3.3:FAU_STG.1

The TOE stores all security audit records in volatile memory, where access is restricted to the logd daemon. Only device owner applications have permission to invoke the MDM API to retrieve a copy of these security logs. New logs can be appended, but there is no supported mechanism for deleting or modifying existing logs in memory. However, the act of reading the security logs via the API clears the log buffer at the time of retrieval. Logcat events are similarly stored in memory, with access restricted to an administrator through an MDM API.

### 7.1.4 PP_MDF_V3.3:FAU_STG.4

The TOE stores SecurityLog and Logcat events in memory using circular log buffers with sizes of 10KB and 256KB, respectively. Both log buffers operate in a circular fashion: when full, the oldest log entries are overwritten by new auditable events. These logs remain in memory until they are either overwritten, or the device undergoes a restart.

## 7.2 Cryptographic support

### 7.2.1 PP_MDF_V3.3:FCS_CKM.1

The TOE supports asymmetric key generation for all types in accordance with FIPS 186-4 through its BoringSSL crypto library. The TOE itself does not generate any RSA/ECDSA authentication key pairs for TOE functionality (the user or administrator must load certificates for use with WPA2/WPA3 with EAP-TLS authentication); however, the TOE provides key generation APIs to mobile applications to allow them to generate RSA/ECDSA key pairs.

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

### 7.2.2 MOD_WLANC_V1.0:FCS_CKM.1/WPA

The TOE adheres to IEEE 802.11-2012 for key generation. The TOE's wpa_supplicant provides PRF384, PRF512 and PRF704 for derivation of 128-bit, 192-bit and 256-bit AES Temporal Keys (using the HMAC implementation provided by BoringSSL) and employs its BoringSSL AES-256 DRBG when generating random values used in the EAP-TLS and 802.11 4-way handshake. The TOE supports the AES-128 CCMP and AES-256 GCMP encryption modes.

The TOE has successfully completed certification (including WPA2/WPA3 Enterprise) and received Wi-Fi CERTIFIED Interoperability Certificates from the Wi-Fi Alliance. The Wi-Fi Alliance maintains a website providing further information about the testing program: http://www.wi-fi.org/certification.

**Table 20. WiFi Alliance Certificate**

| Device Name | Model Number | Wi-Fi Alliance Certificate Numbers |
|---|---|---|
| KATIM | X3M | WFA131102 |

### 7.2.3 PP_MDF_V3.3:FCS_CKM.2/UNLOCKED

The TOE supports key establishment using RSA/ECDH as part of EAP-TLS and TLS session establishment.

### 7.2.4 PP_MDF_V3.3:FCS_CKM.2/LOCKED

The TOE provides an SDP library for applications that uses a hybrid crypto scheme based on 4096-bit RSA based key establishment. Applications can utilize this library to implement SDP that encrypts incoming data received while the phone is locked in a manner compliant with this requirement.

### 7.2.5 MOD_WLANC_V1.0:FCS_CKM.2/WLAN

The TOE adheres to RFC 3394 and 802.11-2012 standards and unwraps the GTK (sent encrypted with the WPA2/WPA3 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).

### 7.2.6 PP_MDF_V3.3:FCS_CKM_EXT.1

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

### 7.2.7 PP_MDF_V3.3:FCS_CKM_EXT.2

The TOE utilizes its approved RBGs to generate DEKs for the various use cases. When generating AES keys for itself (for example, the TOE's sensitive data encryption keys or for the Secure Key Storage), the TOE utilizes the RAND_bytes() API call from its BoringSSL AES-256 CTR_DRBG to generate a 256-bit AES key. The TOE also utilizes that same DRBG when servicing API requests from mobile applications wishing to generate AES keys (either 128 or 256-bit). In all cases, the TOE generates DEKs using a compliant RBG seeded with sufficient entropy so as to ensure that the generated key cannot be recovered with less work than a full exhaustive search of the key space.

### 7.2.8 PP_MDF_V3.3:FCS_CKM_EXT.3

The TOE takes the user-entered password and conditions/stretches this value before combining the factor with other KEK. The TOE generates all non-derived KEKs using the RAND_bytes() API call from its BoringSSL AES-256 CTR_DRBG to ensure a full256-bits of strength for asymmetric/symmetric keys, respectively. The TOE also combines KEKs by encrypting one KEK with the other as to preserve entropy.

### 7.2.9 PP_MDF_V3.3:FCS_CKM_EXT.4

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

### 7.2.10 PP_MDF_V3.3:FCS_CKM_EXT.5

The TOE stores all protected data in encrypted form within the user data partition (either protected data or sensitive data). Upon request, the TOE cryptographically erases the Data-At-Rest DEK protecting the user data partition, clears those keys from memory, reformats the partition, and performs a power-cycle. The TOE's clearing of the keys follows the requirements of FCS_CKM_EXT.4.

### 7.2.11 PP_MDF_V3.3:FCS_CKM_EXT.6

The TOE generates salt nonces (which are just salt values used in WPA2/WPA3) using its AES-256 CTR_DRBG.

### Table 21. Salt Nonces

| Salt Value and Size | RBG Origin | Salt Storage Location |
|---|---|---|
| User password salt (128-bit) | BoringSSL's AES-256 CTR_DRBG | Flash filesystem |
| TLS client_random (256-bit) | BoringSSL's AES-256 CTR_DRBG | N/A (ephemeral) |
| TLS pre_master_secret (384-bit) | BoringSSL's AES-256 CTR_DRBG | N/A (ephemeral) |
| TLS ECDHE private value (256, 384) | BoringSSL's AES-256 CTR_DRBG | N/A (ephemeral) |
| WPA2/WPA3 4-way handshake supplicant nonce (SNonce) | BoringSSL's AES-256 CTR_DRBG | N/A (ephemeral) |

### 7.2.12 MOD_BT_V1.0:FCS_CKM_EXT.8

The TOE generates new ECDH key pairs every time a connection with a Bluetooth device is established.

### 7.2.13 PP_MDF_V3.3:FCS_COP.1/ENCRYPT

### 7.2.14 PP_MDF_V3.3:FCS_COP.1/HASH

### 7.2.15 PP_MDF_V3.3:FCS_COP.1/SIGN

### 7.2.16 PP_MDF_V3.3:FCS_COP.1/KEYHMAC

### 7.2.17 PP_MDF_V3.3:FCS_COP.1/CONDITION

The TOE implements cryptographic algorithms in accordance with the respective NIST standards. These algorithms are in software and hardware, depending on the implementation.

The TOE's BoringSSL Library (version fips-20220613 with both Processor Algorithm Accelerators (PAA) and without PAA) provides the following algorithms.

### Table 22. BoringSSL Cryptographic Algorithms

| SFR | Algorithm | Keys | NIST Standard |
|---|---|---|---|
| FCS_CKM.1 | RSA Key Generation | 2048,3072,4096 | FIPS 186-4, RSA |
| | ECDSA ECC Key Generation | P-256, P-384, P-521 | FIPS 186-4, ECDSA |
| | ECDH Key Generation | curve25519 | RFC 7748 |
| FCS_CKM.2 | RSA-based Key Exchange | Vendor affirm 800-56B | |
| | ECC-based Key Exchange | P256, P384, P521 | SP 800-56A, CVL KAS ECC |
| FCS_COP.1/ENCRYPT | AES CBC, GCM | 128,256 | FIPS 197, SP 800-38A |
| FCS_COP.1/HASH | SHA Hashing | 1/256/384/512 | FIPS 180-4 |
| FCS_COP.1/SIGN | RSA Sign/Verify | 2048, 3072, 4096 | FIPS 186-4, RSA |
| | ECDSA Sign/Verify | P256, P384, P521 | FIPS 186-4, ECDSA |

| | | | |
|---|---|---|---|
| FCS_COP.1 /KEYHMAC | HMAC-SHA | 1/256/384/512 | FIPS 198-1 & 180-4 |
| FCS_RBG_EXT.1 | DRBG Bit Generation | 256 | SP 800-90A (Counter mode) |

The TOE currently utilizes the Linux kernel 5.15 and provides a Kernel Crypto API consisting of the following algorithms.

### Table 23. Kernel Cryptographic Algorithms

| SFR | Algorithm | Keys | NIST Standard |
|---|---|---|---|
| FCS_COP.1/ENCRYPT | AES CBC*/XTS** | 128/256 | Vendor affirm FIPS 197, SP 800-38A/E |
| FCS_CKM_EXT.3. | HKDF (HMAC- SHA512) | | Vendor affirm SP 800-56A (RFC5869) |

* AES-CBC operates in CTS mode.

** For XTS, the keys are split into two values, a 128/256-bit encryption key and a 128/256-bit tweak key (total of 256/512 bits).

The device comes with a Wi-Fi chipset that provides AES-GCM-128/256-bit and AES-GMAC-128/256-bit encryption to meet FCS_COP.1/ENCRYPT that meet the NIST Standards FIPS 197, SP 800-38C.

### Table 24. WiFi Chipset

| Device | Wi-Fi Chipset | Cert # |
|---|---|---|
| KATIM X3M | WCN7850 | WFA131102 |

The QSM8550 Qualcomm platform provides the following components:

### Table 25. Qualcomm Hardware Cryptographic Algorithms

| Component Name | Component Version | Certificate Number | FIPS Certificate Level |
|---|---|---|---|
| Qualcomm Pseudo Random Number Generator (PRNG) | 3.1.0 | 4778<br>*https://csrc.nist.gov/projects/cryptographic- module-validation-program/Certificate/4778* | Level 3 |

The QSM8550 application processor provides the following cryptographic algorithms in its Trusted Execution Environment. These algorithms provide lower-level services upon which some of the security functionalities rely, and are not directly used by the KATIM OS 13 operating system.

### Table 26. TEE Cryptographic Algorithms

| SFR | Algorithm | Algorithm | NIST Standard |
|---|---|---|---|
| FCS_RBG_EXT.1 | BoringSSL DRBG Bit Generation | 256 | SP 800-90A (Counter mode) |
| FCS_CKM_EXT.3 | QSEE KDF | NIST SP 800-108 (counter based) | |

The TOE's application processor includes a source of hardware entropy that the TOE distributes throughout, and the TOE's RBGs make use of that entropy when seeding/instantiating themselves. The TOE's BoringSSL library supports the TOE's cryptographic

KATIM OS Runtime methods (through KATIM OS's conscrypt JNI provider) afforded to mobile applications and supports KATIM OS user-space processes and daemons (e.g., wpa_supplicant). The TOE stretches the user’s password to create a password-derived key. The TOE stretching function uses a series of steps to increase the memory required for key derivation (thus thwarting GPU-acceleration, off-line brute force, and precomputed dictionary attacks) and ensure proper conditioning and stretching of the user’s password.

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

The password-derived key is combined with the hardware REK in storage, preventing the ability to perform offline attacks, and online attacks are limited due to the TOE’s configuration of the maximum no more than 10 incorrect password attempts. The use of the password derivation function in combination with the device configuration forces the attacker to only be able to perform an exhaustive key search to unlock the device without access to the password, and then subject to the configured limit of 10 or less attempts before the device is wiped.

As part of the TLS, the TOE uses SHA with ciphersuites and digital signatures. The TLS ciphersuites support using SHA-1, SHA-256 and SHA-384. SHA functionality is also provided to mobile applications and can also be used as part of HMAC generation. For mobile applications generating a MAC, the HMAC operations in a byte-oriented mode and can use SHA-1 (with a 160-bit key) to generate a 160-bit MAC, SHA-256 (with a 256-bit key) to generate a 256-bit MAC, SHA-384 (with a 384-bit key) to generate a 384- bit MAC and SHA-512 (with a 512-bit key) to generate a 512-bit MAC. FIPS 198-1 & 180-4 dictate the block size used, and they specify block sizes/output MAC lengths of 512/160, 512/160, 1024/384, and 1024/512-bits for HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 respectively.

### 7.2.18 PP_MDF_V3.3:FCS_HTTPS_EXT.1

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

### 7.2.19 PP_MDF_V3.3:FCS_IV_EXT.1

The TOE generates IVs by reading from /dev/urandom for use with all keys. The TOE generates IVs for data storage encryption and for key storage encryption. The TOE uses XTS-AES and AES-CBC mode for data encryption and AES-GCM for key storage.

### 7.2.20 PP_MDF_V3.3:FCS_RBG_EXT.1

The TOE provides a number of different RBGs including:

1. An Application Processor DRBG in hardware:
    1. A SHA-256 Hash_DRBG in Qualcomm Snapdragon processors
2. An AES-256 CTR_DRBG provided by BoringSSL. This is the only accredited and supported DRBG present in the system and available to independently developed applications. As such, the TOE provides mobile applications access (through an KATIM OS Java API) to random data drawn from its AES-256 CTR_DRBG

The TOE initializes its AP DRBG with enough data from its AP hardware noise source to ensure at least 256-bits of entropy. The TOE then uses its AP DRBG to seed *the Linux kernel's RNG pool and* an entropy daemon that uses the BoringSSL AES-256 CTR_DRBG to provide random bits for user space. The entropy daemon starts early in the boot process to ensure availability to the rest of the system.

The TOE seeds its BoringSSL AES-256 CTR_DRBG using 384-bits of data from the entropy daemon, thus ensuring at least 256-bits of entropy. The TOE uses its BoringSSL DRBG for all random generation including salts.

### 7.2.21 PP_MDF_V3.3:FCS_SRV_EXT.1

The TOE provides applications access to the cryptographic operations including encryption (AES), hashing (SHA), signing and verification (RSA & ECDSA), key hashing (HMAC), keyed message digests (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 operating system's Java API, through the native BoringSSL API, and through the application processor module (user and kernel) APIs.

### 7.2.22 PP_MDF_V3.3:FCS_STG_EXT.1

The TOE provides the user, administrator and mobile applications the ability to import and use asymmetric public and private keys into the TOE's software-based Secure Key Storage. Certificates are stored in files using UID-based permissions and an API virtualizes the access. Additionally, the user and administrator can request the TOE to destroy the keys stored in the Secure Key Storage. While normally mobile applications cannot use or destroy the keys of another application, applications that share a common application developer (and are thus signed by the same developer key) may do so. In other words, applications with a common developer (and which explicitly declare a shared UUID in their application manifest) may use and destroy each other's keys located within the Secure Key Storage.

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

### 7.2.23 PP_MDF_V3.3:FCS_STG_EXT.2

The TOE employs a key hierarchy that protects all DEKs and KEKs by encryption with either the REK or by the REK and password derived KEK.

The TOE encrypts Long-term Trusted channel Key Material (LTTCKM, i.e., Bluetooth and Wi-Fi keys) values using AES-256 GCM encryption and stores the encrypted values within their respective configuration files.

All keys are 256-bits in size. The TOE generates keys using its BoringSSL AES-256 CTR_DRBG (for the Java and native layer), and the Qualcomm Snapdragon processor SHA-256 Hash_DRBG is used by Trusted Applications in TrustZone. By utilizing only 256-bit KEKs, the TOE ensures that all keys are encrypted by an equal or larger sized key.

In the case of Wi-Fi, the TOE utilizes the 802.11-2012 KCK and KEK keys to unwrap (decrypt) the WPA2/WPA3 Group Temporal Key received from the access point. The TOE protects persistent Wi-Fi keys (user certificates and private keys) by storing them in the Key Store.

### 7.2.24 PP_MDF_V3.3:FCS_STG_EXT.3

The TOE protects the integrity of all DEKs and KEKs (including LTTCKM keys) stored in Flash by using authenticated encryption/decryption methods (GCM).

### 7.2.25 PKG_TLS_V1.1:FCS_TLS_EXT.1

### 7.2.26 PKG_TLS_V1.1:FCS_TLSC_EXT.1

### 7.2.27 PKG_TLS_V1.1:FCS_TLSC_EXT.2

The TOE provides mobile applications (through its operating system API) the use of TLS version 1.2 as a client, including support for the selections chosen in section 6 for FCS_TLSC_EXT.1 (and the TOE requires no configuration other than using the appropriate library APIs as described in the Admin Guidance).

When an application uses the combined APIs provided in the Admin Guide to attempt to establish a trusted channel connection based on TLS or HTTPS, the TOE supports only Subject Alternative Name (SAN) (DNS and IP address) as reference identifiers (the TOE does not accept reference identifiers in the Common Name[CN]). The TOE supports client (mutual) authentication (only a certificate is required to provide support for mutual authentication).

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

While the TOE supports the use of wildcards in X.509 reference identifiers (SAN only). The TOE supports certificate pinning through operating system's network security configuration. This configuration allows a mobile application to specify one or more certificate public key hashes (SHA-256) along with the domain and optionally an expiry date. If the TOE cannot determine the revocation status of a peer certificate, the TOE rejects the certificate and rejects the connection.

### 7.2.28 PKG_TLS_V1.1:FCS_TLSC_EXT. 4

The TOE includes the 'renegotiation_info' TLS extension in its TLS client hello message.

### 7.2.29 PKG_TLS_V1.1:FCS_TLSC_EXT.5

The TOE in its evaluated configuration and, by design, supports elliptic curves for TLS (P-256 and P-384) and has a fixed set of supported curves (thus the admin cannot and need not configure any curves).

### 7.2.30 MOD_WLANC_V1.0:FCS_TLSC_EXT.1/WLAN

### 7.2.31 MOD_WLANC_V1.0:FCS_TLSC_EXT.2/WLAN

The TSF supports TLS version 1.2 and the selected cipher suites utilizing SHA-1, SHA-256, and SHA-384 (see the selections in section 6 for FCS_TLSC_EXT.1/WLAN) for use with EAP-TLS as part of WPA2/WPA3. The TOE in its evaluated configuration and, by design, supports only evaluated elliptic curves (P-256 & P-384 and no others) and has a fixed set of supported curves (thus the admin cannot and need not configure any curves).

The TOE allows the user to load and utilize authentication certificates for EAP-TLS used with WPA3/WPA2. The operating system UI

*Settings -> Security -> Advanced settings -> Encryption & credentials -> Install a certificate -> Wi-Fi certificate*

allows the user to import an RSA or ECDSA certificate for use with Wi-Fi.

### 7.2.32 MOD_WLANC_V1.0:FCS_WPA_EXT.1

The TSF support WPA2 and WPA3 security types for Wi-Fi networks.

## 7.3 User data protection

### 7.3.1 PP_MDF_V3.3:FDP_ACF_EXT.1

The TOE provides a mechanism for protecting (restricting application) access to system services. This mechanism is based on permissions that are associated with services. If an application requires a service, it needs to be granted (by the user) the permission for that service.

The permission mechanism is two-layered. A permission assigned to a service contains two attributes: a base permission and protection flags. Protection flags can be used for refinement of the base permission.

The TOE provides the following base permissions to applications (for API Level 33):

1. Normal - A lower-risk permission that gives an application access to isolated application-level features, with minimal risk to other applications, the system, or the user. The system automatically grants this type of permission to a requesting application at installation, without asking for the user's explicit approval (though the user always has the option to review these permissions before installing).

2. Dangerous - A higher-risk permission that would give a requesting application access to private user data or control over the device that can negatively impact the user. Because this type of permission introduces potential risk, the system cannot automatically grant it to the requesting application. For example, any dangerous permissions requested by an application will be displayed to the user and require confirmation before proceeding or some other approach can be taken to avoid the user automatically allowing the use of such facilities.

3. Signature - A permission that the system is to grant only if the requesting application is signed with the same certificate as the application that declared the permission. If the certificates match, the system automatically grants the permission without notifying the user or asking for the user's explicit approval.

4. Internal - a permission that is managed internally by the system and only granted according to the protection flags.

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

An example of a dangerous privilege would be access to location services to determine the location of the mobile device: android.permission.ACCESS_FINE_LOCATION. The TOE controls access to Dangerous permissions during the running of the application. The TOE prompts the user to review the application's requested permissions (by displaying a description of each permission group, into which individual permissions map, that an application requested access to). If the user approves, then the application is allowed to continue running. If the user disapproves, the devices continues to run, but cannot use the services protected by the denied permissions. Thereafter, the mobile device grants that application during execution access to the set of permissions declared in its Manifest file.

An example of a signature permission is the android.permission.BIND_VPN_SERVICE that an application must declare in order to utilize the VpnService APIs of the device. Because the permission is a Signature permission, the mobile device only grants this permission to an application (2nd installed app) that requests this permission and that has been signed with the same developer key used to sign the application (1st installed app) declaring the permission (in the case of the example, the operating system itself).

An example of an internal permission is the android.permission.SET_DEFAULT_ACCOUNT_FOR_CONTACTS, which is only granted to system applications fulfilling the Contacts app role to allow the default account for new contacts to be set.

The TOE provides the following protection flags:

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

The operating system (Level 33) API (details found here https://developer.android.com/reference/packages) provides services to mobile applications. While the operating system provides a large number of individual permissions, they are grouped into categories or features that provide similar functionality for the simplicity of the user interaction. These groupings do not affect the permissions themselves; it is only a way to group them together for the user presentation. Table 27 shows a series of functional categories centered on common functionality.

## Table 27. Functional Categories

| Service Features | Description |
|---|---|
| Sensitive I/O Devices & Sensors | Location services, Audio &Video capture, Body sensors |
| User Personal Information &Credentials | Contacts, Calendar, Call logs, SMS |

| | |
|---|---|
| Metadata & Device ID Information | IMEI, Phone Number |
| Data Storage Protection | App data, App cache |
| System Settings & Application Management | Date time, Reboot/Shutdown, Sleep, Force-close application, Administrator Enrollment |
| Wi-Fi, Bluetooth, USB Access | Wi-Fi, Bluetooth, USB tethering, debugging and file transfer |
| Mobile Device Management &amp; Administration | MDM APIs |
| Peripheral Hardware | NFC, Camera, Headphones |
| Security & Encryption | Certificate/Key Management, Password, Revocation rules |

### 7.3.2 PP_MDF_V3.3:FDP_ACF_EXT.1.2

Applications coming from the same developer can allow sharing of data between them. The developer can sign applications' APKs with a common certificate or key and set the permissions of their application to allow data sharing in their manifest.

### 7.3.3 PP_MDF_V3.3:FDP_ACF_EXT.2

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

### 7.3.4 PP_MDF_V3.3:FDP_DAR_EXT.1

The TOE provides AES-256 XTS FBE encryption for all data (user data and TSF data) stored on the user data partition. TSF data contains items relating to key storage for TSF keys that are not stored in the system's Key Store. The TOE separately encrypts those TSF keys and data. The TOE also has a read-only file system where system executables, libraries, and their configuration data are stored. For encrypting the data partition on the internal Flash, the TOE uses an AES-256 bit DEK with XTS feedback mode. Each file has a unique DEK.

### 7.3.5 PP_MDF_V3.3:FDP_DAR_EXT.2

The vendor provides the NIAPSEC library for Sensitive Data Protection (SDP) that application developers must use to opt-in for sensitive data protection. When developers opt-in for SDP, all data that is received on the device destined for that application is treated as sensitive. This library calls into the TOE to generate an RSA key that acts as a master KEK for the SDP encryption process. When an application that has opted-in for SDP receives incoming data while the device is locked, an AES symmetric DEK is generated to encrypt that data. The public key from the master RSA KEK above is then used to encrypt the AES DEK. Once the device is unlocked, the RSA KEK private key is re-derived and can be used to decrypt the AES DEK for each piece of information that was stored while the device was locked. The TOE then takes that decrypted data and re-encrypts it following FDP_DAR_EXT.1.

### 7.3.6 PP_MDF_V3.3:FDP_IFC_EXT.1

The TOE will route all traffic other than traffic necessary to establish the VPN connection to the VPN gateway (when the gateway's configuration specifies so) when the Always-On-VPN is enabled. The TOE includes an interceptor kernel module that controls inbound and output packets. When a VPN is active, the interceptor will route all incoming packets to the VPN and conversely route all outbound packets to the VPN before they are output.

Note that when the TOE tries to connect to a Wi-Fi network, it performs a standard captive portal check which sends traffic that bypasses the full tunnel VPN configuration in order to detect whether the Wi-Fi network restricts Internet access until one has authenticated or agreed to usage terms through a captive portal.

The only exception to all traffic being routed to the VPN is in the instance of ICMP echo requests. The TOE uses ICMP echo responses on the local subnet to facilitate network troubleshooting and categorizes it as a part of ARP. As such, if an ICMP echo request is issued on the subnet the TOE is part of, it will respond with an ICMP echo response, but no other instances of traffic will be routed outside of the VPN.

### 7.3.7 PP_MDF_V3.3:FDP_STG_EXT.1

The TOE's Trusted Anchor Database consists of the built-in certificates residing in /system/etc/security/cacerts/ and any additional MDM loaded certificates, which are stored in /data/misc/user/0/cacerts-added/.

The former location is on the read-only system partition. The TOE prevents applications (other than administrator/MDM agent applications) from modifying files in the latter location using Linux file permissions. Only applications registered as an administrator (such as an MDM Agent Application) have the ability to access these files.

### 7.3.8 PP_MDF_V3.3:FDP_UPC_EXT.1/APPS

The TOE provides APIs allowing non-TSF applications (mobile applications) the ability to establish a secure channel using TLS, HTTPS, and Bluetooth DR/EDR and LE. Additionally, the vendor provides the NIAPSEC library for application developers to use for Hostname Checking, Revocation Checking, and TLS Ciphersuite restriction. Application developers must utilize this library to ensure the device behaves in the evaluated configuration. Mobile applications can use the following APIs for TLS and HTTPS:

javax.net.ssl.SSLContext: https://developer.android.com/reference/javax/net/ssl/SSLSocket

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

Bluetooth:

android.bluetooth:

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

### 7.3.9 PP_MDF_V3.3:FDP_UPC_EXT.1/BLUETOOTH

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

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

## 7.4 Identification and authentication

### 7.4.1 PP_MDF_V3.3:FIA_AFL_EXT.1

The TOE keeps the number of failed password login attempts since the last successful one in non-volatile memory. If that number reaches the predefined maximum, the TOE performs a full wipe of all user data. Only password authentication attempts are being compared to the maximum value because only the password authentication factor is considered critical. Fingerprint unlock failed attempts cannot cause a wipe.

The default value for the maximum number of failed login attempts is 10 and can be adjusted via the MDM API to a value between 0 and 10. During an authentication attempt, the TOE first increments the failed attempt counter, and then checks the validity of the password by sending it to Gatekeeper.

Gatekeeper stores the attempt counter in non-volatile secure storage and increments it before validating the password. If the password validation succeeds, the counter is reset to zero. If the validation fails and the counter value reaches the specified maximum, the device is wiped. Since the counter is stored in non-volatile memory and incremented before validation, failed attempt count will be preserved even if power is lost.

The device can also be unlocked by the user using their fingerprint. The TOE permits up to 5 failed fingerprint unlock attempts and then disables the fingerprint authentication. It is enabled again only when the user enters the correct password to unlock the device. If the phone is restarted, fingerprint sensor is disabled until the user unlocks the device with the password.

### 7.4.2 MOD_BT_V1.0:FIA_BLT_EXT.1

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

### 7.4.3 MOD_BT_V1.0:FIA_BLT_EXT.2

The TOE does not allow data transfer with unauthorized remote devices. All Bluetooth connections require initial approval by the user in an authorization popup. Bluetooth pairing (RFCOMM connections) is done by confirming/entering a displayed numeric passcode in the user interface. TOE support for OBEX (OBject EXchange) through L2CAP (Logical Link Control and Adaptation Protocol) requires the user to explicitly authorize the transfer via a popup that will be displayed to the user.

### 7.4.4 MOD_BT_V1.0:FIA_BLT_EXT.3

The TOE keeps track of active sessions and ignores duplicate Bluetooth connection attempts from devices for which it already has and active connection.

### 7.4.5 MOD_BT_V1.0:FIA_BLT_EXT.4

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

### 7.4.6 MOD_BT_V1.0:FIA_BLT_EXT.6

The TOE requires explicit user authorization before granting trusted (paired) remote devices access to services associated with the OPP and MAP Bluetooth profiles.

### 7.4.7 MOD_BT_V1.0:FIA_BLT_EXT.7

The TOE requires untrusted (unpaired) remote devices to have explicit user authorization before granting them access to services associated with all Bluetooth profiles.

### 7.4.8 MOD_WLANC_V1.0:FIA_PAE_EXT.1

The TOE can join WPA2-802.1X (802.11i) and WPA3-Enterprise 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).

### 7.4.9 PP_MDF_V3.3:FIA_PMG_EXT.1

The TOE authenticates the user through a password that can consist of basic Latin characters (upper and lower case, numbers, and the special characters noted in the selection (see the selections in section 6 for FIA_PMG_EXT.1)). The default password requirements are that it must have a minimum of 4 characters but no more than 16 and that it contains at least one letter. These defaults can be changed using the MDM API.

### 7.4.10 PP_MDF_V3.3:FIA_TRT_EXT.1

GateKeeper throttling is used to prevent brute-force password attacks. After a user enters an incorrect password GateKeeper APIs return a value in milliseconds in which the caller must wait before attempting another validation. Any attempts before the defined amount of time has passed will be ignored by GateKeeper. Gatekeeper also keeps a count of the number of failed validation attempts since the last successful attempt. Based on the value of this counter the amount of time that the caller has to wait before attempting a new password authentication increases exponentially.

### 7.4.11 PP_MDF_V3.3:FIA_UAU.5

The TOE allows the user to authenticate using a password. After boot, the first unlock screen requires the user to enter their password to unlock the device. The fingerprint sensor is disabled until the user enters the correct password for the first time.

After the device is locked in normal use, the user has the ability to unlock it either by entering their password or by using the fingerprint authentication. Throttling of these inputs is described in section 6 for FIA_AFL_EXT.1.

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

### 7.4.12 PP_MDF_V3.3:FIA_UAU.6/CREDENTIAL, PP_MDF_V3.3:FIA_UAU.6/LOCKED

The TOE requires the user to enter their password in order to unlock the TOE. Additionally the TOE requires the user to confirm their current password when accessing the

Settings > Security > Screen lock

menu in the TOE's user interface. Only after entering their current user password can the user then elect to change their password.

### 7.4.13 PP_MDF_V3.3:FIA_UAU.7

The TOE allows users to input their passwords from the lock screen. By default, the TOE does not display any characters of the entered credentials. However, there is a settings option that allows for the brief momentary display of each character as it is typed, followed by obscuring the character with a dot symbol once the next character is entered. This setting can be toggled to accommodate user preferences.

### 7.4.14 PP_MDF_V3.3:FIA_UAU_EXT.1

The TOE's key hierarchy requires the user's password in order to derive the KEK_* keys in order to decrypt other KEKs and DEKs. Thus, until it has the user's password, the TOE cannot decrypt the DEK utilized by FBE to decrypt the user's protected data.

### 7.4.15 PP_MDF_V3.3:FIA_UAU_EXT.2

The TOE, when configured to require a user password (like in the CC mode case), allows a user to perform the actions assigned in FIA_UAU_EXT.2.1 (see selections in section 6 for FIA_UAU_EXT.2) without first successfully authenticating. The TOE automatically names and saves (to the internal storage) any screen shots taken from the lock screen, and the TOE provides the user no opportunity to name them or change where they are stored.

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

### 7.4.16 PP_MDF_V3.3:FIA_X509_EXT.1, MOD_WLANC_V1.0:FIA_X509_EXT.1/WLAN

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

### 7.4.17 PP_MDF_V3.3:FIA_X509_EXT.2, MOD_WLANC_V1.0:FIA_X509_EXT.2/WLAN, MOD_WLANC_V1.0:FIA_X509_EXT.6

The TOE uses X.509v3 certificates during EAP-TLS, TLS, and HTTPS. The TOE contains a preinstalled set of default Trusted Credentials. The user cannot remove any of these built-in default CA certificates. The administrator/MDM can also import a new trusted CA certificate into the Trust Anchor Database.

The TOE does not establish TLS connections itself (beyond EAP-TLS used for WPA2/WPA3 Wi-Fi connections) but provides a series of APIs to mobile applications for checking the validity of a peer certificate. The mobile application, after correctly using the specified APIs, can be assured as to the validity of the peer certificate and be assured that the TOE will not establish the trusted connection if the peer certificate cannot be verified (including validity, certification path, and revocation [through OCSP]). If, during the process of certificate verification, the TOE cannot establish a connection with the server acting as the OCSP Responder, the TOE will not deem the server's certificate as valid and will not establish a TLS connection with the server.

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

The TOE, when acting as a WPA2/WPA3 supplicant uses X.509 certificates for EAP-TLS authentication. Because the TOE may not have network connectivity to a revocation server prior to being admitted to the WPA2/WPA3 network and because the TOE cannot determine the IP address or hostname of the authentication server (the Wi-Fi access point proxies the supplicant's authentication request to the server), the TOE will accept the certificate of the server.

### 7.4.18 PP_MDF_V3.3:FIA_X509_EXT.3

Applications that require compliant revocation checking must utilize the NIAPSEC library. This vendor provide library offers the following functions for certificate path validation and revocation checking:

- public boolean isValid(List<Certificate> certs)
- public boolean isValid(Certificate cert)

The first function allows for validation and revocation checking of a list of certificates. The second function checks a single certificate. Revocation checking is done using OCSP. Section 6 for FIA_X509_EXT.2/WLAN details how the TOE handles revocation checking.

## 7.5 Security management

### 7.5.1 PP_MDF_V3.3:FMT_MOF_EXT.1, PP_MDF_V3.3:FMT_SMF.1

The TOE provides the management functions described in Table 18 - Security Management Functions. The table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted. It is worth noting that the TOE's ability to specify authorized application repositories takes the form of allowing enterprise applications (i.e., restricting applications to only those applications installed by an MDM Agent).

### 7.5.2 MOD_WLANC_v1.0:FMT_SMF.1/WLAN

The TOE provides the management functions described in Table 20 - WLAN Security Management Functions. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted.

### 7.5.3 MOD_BT_V1.0:FMT_SMF_EXT.1/BT

The TOE provides the management functions described in Table 19 - Bluetooth Security Management Functions. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted.

### 7.5.4 PP_MDF_V3.3:FMT_SMF_EXT.2

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

## 7.6 Protection of the TSF

### 7.6.1 PP_MDF_V3.3:FPT_AEX_EXT.1

The Linux kernel of the TOE's KATIM 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.

### 7.6.2 PP_MDF_V3.3:FPT_AEX_EXT.2

The TOE utilizes 5.15 Linux kernel, whose memory management unit (MMU) enforces read, write, and execute permissions on all pages of virtual memory and ensures that write and execute permissions are not simultaneously granted on all memory. The operating system sets the ARM eXecute Never (XN) bit on memory pages and the TOE's ARMv9 Application Processor's Memory Management Unit (MMU) circuitry enforces the XN bits. KATIM OS supports 'Hardware-based No eXecute (NX) to prevent code execution on the stack and heap. The ARMv9 Architecture Reference Manual contains additional details about the MMU of ARM-based processors: https://developer.arm.com/documentation/ddi0487/ka.

### 7.6.3 PP_MDF_V3.3:FPT_AEX_EXT.3

The TOE's 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. KATIM builds the TOE (all executable binaries and libraries) using -fstack-protector to make the stack and heap non-executable.

### 7.6.4 PP_MDF_V3.3:FPT_AEX_EXT.4

The TOE protects itself from modification by untrusted subjects using a variety of methods. The first protection employed by the TOE is a Secure Boot process that uses cryptographic signatures to ensure the authenticity and integrity of the bootloader and kernel using data fused into the device processor. The TOE protects its REK by limiting access to only trusted applications within the TEE (Trusted Execution Environment). The TOE key manager includes a TEE module that utilizes the REK to protect all other keys in the key hierarchy. All TEE applications are cryptographically signed, and when invoked at runtime (at the behest of an untrusted application), the TEE will only load the trusted application after successfully verifying its cryptographic signature.

Additionally, the TOE's 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.

### 7.6.5 PP_MDF_V3.3:FPT_JTA_EXT.1

The TOE prevents access to its processor's JTAG interface in hardware by removing the PCB traces from the SoC pins to the JTAG test connector. In addition, the JTAG interface is also disabled in software by the fuse configuration.

### 7.6.6 PP_MDF_V3.3:FPT_KST_EXT.1

The TOE does not store any plaintext key in its internal Flash; the TOE encrypts all keys before storing them. This ensures that irrespective of how the TOE powers down (e.g., a user commands the TOE to power down, the TOE reboots itself, or battery depletes), all keys stored in the internal Flash are wrapped with a KEK. Please refer to section 7.2 of the TSS for further information (including the KEK used) regarding the encryption of keys stored in the internal Flash. As the TOE encrypts all keys stored in Flash, upon boot-up, the TOE must first decrypt any keys in order to utilize them.

### 7.6.7 PP_MDF_V3.3:FPT_KST_EXT.2

The TOE utilizes cryptographic libraries including BoringSSL, application processor cryptography (which leverages AP hardware), and the following system-level executables that utilize KEKs: vold, wpa_supplicant, and the KATIM OS Key Store.

1. vold and Linux kernel provides Data-At-Rest encryption of the user data partition in Flash
2. wpa_supplicant provides WPA2/WPA3 services
3. the operating system's Key Store application provides key generation, storage, deletion services to mobile applications and to user through the UI

The TOE ensures that plaintext key material is not exported by not allowing the REK to be exported and by ensuring that only authenticated entities can request utilization of the REK. Furthermore, the TOE only allows the system-level executables access to plaintext DEK values needed for their operation. The TSF software (the system-level executables) protects those plaintext DEK values in memory both by not providing any access to these values and by clearing them when no longer needed (in compliance with FCS_CKM_EXT.4).

### 7.6.8 PP_MDF_V3.3:FPT_KST_EXT.3

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

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

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

### 7.6.9 PP_MDF_V3.3:FPT_NOT_EXT.1

When the TOE encounters a self-test failure or software integrity verification failure, a failure is message is displayed to the screen, and the TOE shuts down. If the failure persists over several boot attempts, the needs to take the device to a service point for a factory reset.

### 7.6.10 PP_MDF_V3.3:FPT_STM.1

The TOE requires time for the Package Manager (which installs and verifies APK signatures and certificates), image verifier, wpa_supplicant, and Key Store applications. These TOE components obtain time from the TOE using system API calls [e.g., time() or gettimeofday()]. An application (unless a system application is residing in /system/priv-app or signed by the vendor) cannot modify the system time as mobile applications need the KATIM OS 'SET_TIME' permission to do so. Likewise, only a process with root privileges can directly modify the system time using system-level APIs. Further, this stored time is used both for the time/date tags in audit logs and is used to track inactivity timeouts that force the TOE into a locked state.

By default, the TOE prioritizes NTP as the time source over Cellular Carrier time because NTP is more accurate and reliable. In situations where NTP isn't available, the framework falls back on Cellular Carrier time (obtained through the Carrier's network time server). It is also possible to let the user set the date and time through the TOE's user interface.

### 7.6.11 PP_MDF_V3.3:FPT_TST_EXT.1

The TOE automatically performs known answer power on self-tests (POST) on its cryptographic algorithms to ensure that they are functioning correctly. Each component providing cryptography performs these tests. Should any of the tests fail, the TOE displays a boot failure error message, halts the boot process, and shuts down.

**Table 28. Power-on Cryptographic Algorithm Self-Tests**

| Algorithm | Implemented in | Description |
|---|---|---|
| AES encryption/decryption | BoringSSL | Comparison of known answer to calculated value |

| | | |
|---|---|---|
| ECDH key agreement | BoringSSL | Comparison of known answer to calculated value |
| DRBG random bit generation | BoringSSL | Comparison of known answer to calculated value |
| ECDSA sign/verify | BoringSSL | Comparison of known answer to calculated value |
| HMAC-SHA | BoringSSL | Comparison of known answer to calculated value |
| RSA sign/verify | BoringSSL | Comparison of known answer to calculated value |
| SHA hashing | BoringSSL | Comparison of known answer to calculated value |

### 7.6.12 PP_MDF_V3.3:FPT_TST_EXT.2/PREKERNEL, MOD_WLANC_V1.0:FPT_TST_EXT.3/WLAN

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

### 7.6.13 PP_MDF_V3.3:FPT_TUD_EXT.1

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

### 7.6.14 PP_MDF_V3.3:FPT_TUD_EXT.2

The TOE verifies all updates to the TOE software using a public key chaining that is rooted in the Root Public Key, a hardware protected key whose SHA-256 hash resides inside the application processor's internal fuses. If this verification fail, the software update will fail and the update will not be installed.

The application processor verifies the bootloader's authenticity and integrity thus tying the bootloader and subsequent stages to a the mentioned fixed/fused hardware root of trust.

### 7.6.15 PP_MDF_V3.3:FPT_TUD_EXT.3

The TOE requires applications to have a valid signature before they can be installed on the device.
The contents of the application installation package (APK) are hashed and then signed and the signature is packed with the APK. When an application is downloaded, its signature is verified before installation.

## 7.7 TOE access

### 7.7.1 PP_MDF_V3.3:FTA_SSL_EXT.1

The TOE transitions to its locked state either immediately after the user initiates a lock by pressing the power button or after a configurable period of inactivity. After going to the locked state, the TOE will display a lock screen to obscure the previous contents. The TOE's lock screen still displays notifications, battery life, signal strength, and carrier network and allows the user to perform the functions listed in section 6 for FIA_UAU_EXT.2. The user cannot act on notifications without authenticating first.

The administrator can also force the device into the locked state through the use of an MDM.

On power up, the TOE boots to the device lock screen. At that point, the user can only make emergency calls, receive calls, enter their password or see notifications from apps that do not require user authentication, i.e. the apps that rely on Device Encrypted (and not Credential Encrypted) storage.

### 7.7.2 PP_MDF_V3.3:FTA_TAB.1

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

### 7.7.3 MOD_WLANC_V1.0:FTA_WSE_EXT.1

The TOE allows an administrator to specify (using an MDM) a list of wireless networks (SSIDs) to which the user may direct the TOE to connect. When not enrolled with an MDM, the TOE allows the user to control to which wireless networks the TOE should connect, but does not provide an explicit list of such networks, rather the user may scan for available wireless network (or directly enter a specific wireless network), and then connect.

## 7.8 Trusted path/channels

### 7.8.1 MOD_BT_V1.0:FTP_BLT_EXT.1

### 7.8.2 MOD_BT_V1.0:FTP_BLT_EXT.3/BR

### 7.8.3 MOD_BT_V1.0:FTP_BLT_EXT.3/LE

The TOE mandates the use of encryption over Bluetooth BD/EDR and LE connections using keys of at least 128-bits.

### 7.8.4 MOD_BT_V1.0:FTP_BLT_EXT.2

The TOE will terminate a connection with a remote device if the remote device stops encryption.

### 7.8.5 PP_MDF_V3.3:FTP_ITC_EXT.1:

The TOE provides secured (encrypted and mutually authenticated) communication channels between itself and other trusted IT products through the use of TLS and HTTPS. The TOE provides access to HTTPS and TLS via public APIs to applications requiring encrypted end-to-end trusted channel.

### 7.8.6 MOD_WLANC_V1.0:FTP_ITC.1/WLAN:

The TOE provides secured (encrypted and mutually authenticated) communication channels between itself and other trusted IT products through the use of IEEE 802.11-2012, 802.1X, and EAP-TLS. The TOE allows applications to initiate communication via the trusted channel, and the TOE initiates communications via the WPA2/WPA3 (IEEE 802.11-2012, 802.1X with EAP-TLS) trusted channel for connection to a wireless access point.