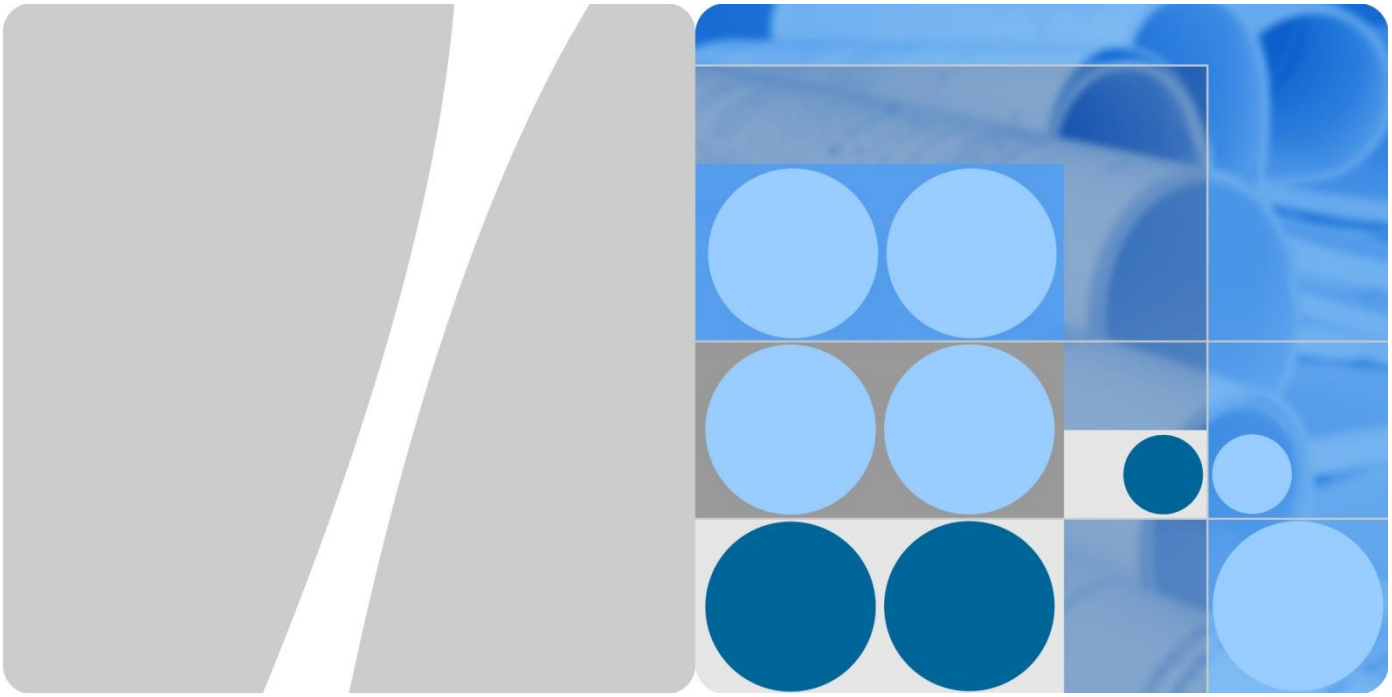


HarmonyOS 3.0 on MatePad Pro Security Target

Issue 1.0

Date 2023-12-15

HUAWEI DEVICE CO., LTD.

Trademarks and Permissions


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

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

Notice

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

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

Huawei Device Co., Ltd.

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

Website: http://www.huawei.com

PSIRT Email: PSIRT@huawei.com

About This Document

Purpose

This Security Target is for the evaluation of HarmonyOS 3.0 on MatePad Pro.

Change History

Changes between document issues are cumulative. The latest document issue contains all the changes made in earlier issues.

Date	Version	Change Description	Author
2023-12-15	1.0	First release	SGS Brightsight

Contents

Tables

1 Security Target Introduction

This section contains the Security Target (ST) and Target of Evaluation (TOE) identifications, TOE overview, and TOE description. The TOE is HarmonyOS 3.0 on MatePad Pro, as listed in section 1.4 “TOE Description”, provided by Huawei Device Co., Ltd.

The Security Target contains the following additional sections:

- Conformance Claims (Section 2)
- Security Problem Definition (Section 3)
- Security Objectives (Section 4)
- Extended Components Definition (Section 5)
- Security Requirements (Section 6)
- TOE Summary Specification (Section 7)
- Abbreviations, Terminology and References (Section 8)

1.1 Security Target Reference

ST Title: HarmonyOS 3.0 on MatePad Pro Security Target

ST Version: 1.0

ST Date: 2023-12-15

Developer: Huawei Device Co., Ltd.

1.2 TOE Reference

TOE identification: HarmonyOS 3.0 on MatePad Pro, as listed in Table 1-1.

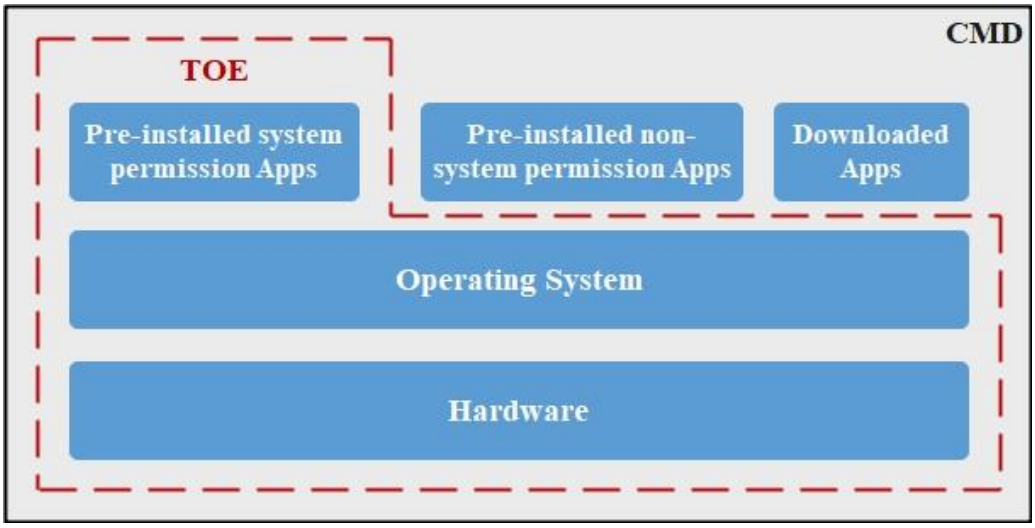
TOE Developer: Huawei Device Co., Ltd.

Evaluation Sponsor: Huawei Device Co., Ltd.

1.3 TOE Overview

The Target of Evaluation (TOE) is the Huawei MatePad Pro running the HarmonyOS 3.0 operating system, please refer to Table 1-1 for the list of evaluated devices.

The TOE is consumer mobile devices (CMD), includes hardware, an operating system and pre-installed system permission apps that are delivered with the CMD out of the box. Pre-installed non- system permission apps and apps that are installed later on by the user (Downloaded Apps) are not considered part of the TOE.



The hardware of the TOE includes the hardware platform, physical enclosure and peripheral components such as sensors and the display. The hardware does not include any devices removable by a human user, including the certified UICC according to [18]. Any data on these devices or services associated with these devices are out of the scope of the TOE. Any data on these devices or services associated with these devices is out of scope of the TOE.

The operating system of the TOE controls and manages the hardware and the apps (both pre-installed and downloaded) and provides the user operation interface and application programming interface(s). The operating system provides secure domain and isolation of apps.

The pre-installed apps are apps that are already present on the TOE when it is delivered to the consumer. Pre-installed apps can be divided into two kinds:

1) Pre-installed system permission apps: apps which have permissions to manage the operating system (such as power off), provide core functions (such as SMS and Telephone), or access to underlying software and hardware interfaces. These permissions are granted by the operating system and cannot be revoked by the human user. These permissions are denoted as system permissions. System permission apps include apps that provide core operating system functionality or security enforcement functionality, apps signed by a platform key from the operating system provider, and other apps allowed by the TOE developer to get such permissions. System permission apps can also require permissions granted by the human user in addition to system permissions.

2) Pre-installed non-system permission apps: apps which do not have system permission. Non- system permission apps can have permissions granted by human user to access to user data and/or permissions granted by the operating system which are necessarily to the operation of apps. Non-system permission apps can usually be un-installed.

Downloaded Apps are apps that are downloaded and installed by the user, and can subsequently be uninstalled by the human user. Downloaded Apps do not have system permissions.

The TOE are consumer mobile devices with wireless connectivity, high computation power and rich user interface. Users can customize the device by downloading apps and changing settings. Users can perform a wide range of actions with the TOE, such as make video calls, perform various productivity tasks, play games, music and videos, and access the Internet. The major security features are:

- User authentication

- Secure communication with other devices
- Secure updating of TOE software
- Integrity verification of the TOE at start-up
- Protecting user data at different levels of security
- Granting/revoking permissions to apps
- Protecting user privacy
- Protection against physical attacks

The evaluated devices are the HUAWEI MatePad Pro (Model Number: WGR-W09).

The software identification for the evaluated devices is as follows:

- Kernel Version: 5.10.43
- Build Number: 3.0.0.116

Table 1-1 The Detailed Description of Evaluated Devices

Device Name	Model Number	Chipset Vendor	CPU	Build Arch/ISA[1]	OS Version	Kernel Version
HUAWEI MatePad Pro	WGR-W09	HiSilicon	Kirin 9000E	ARM 64	HarmonyOS 3.0	5.10.43

1.3.1 Required non-TOE Hardware/Software/Firmware

The TOE requires a certified UICC cellular mobile communication interface in order to provide WAN connectivity to the TOE.

1.4 TOE Description

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

The TOE also provides users with the ability to protect user data at different levels of security. The TOE supports classification of user data according to risk levels and usage scenarios and protects user data to ensure it is accessed by the right person on the right device in the right condition.

1.4.1 Physical Boundaries

The TOE's physical boundary is the physical perimeter of its enclosure. The TOE does not include that are installed later on by the user (Downloaded Apps) and pre-installed non-

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

system permission apps that run on top of the operating system, but does include controls that limit application behavior.

TOE components comprise the items identified in Table 1-2 and Table 1-4.

Table 1-2 TOE Components

TOE Part	Unique identifier	Form of delivery	Delivery method
HUAWEI MatePad Pro Device	See Table 1-1	Hardware device	Physical delivery
HarmonyOS (REE OS)	version: 3.0.0.116 Checksum: d9a9a6c0ca06832b2abf6be616eed611	Software package	Preloaded in device
TEE OS	iTrustee 6.1, including the following TAs[2]: task_gatekeeper TA task_keymaster TCIS TA Facerecognition TA FBE TA		
REE Apps	See Table 1-3		
Secure Monitor	as part of the REE OS		
TOE Guidance	See Table 1-4		

Table 1-3 REE apps

Application Name	Version number
System	12
Android Services Library	20.1.0.0
Android Shared Library	1
androidhwext	12
Bluetooth	29.1.0.0
Bluetooth MIDI Service	12
Bookmark Provider	12
Call Log Backup/Restore	12
Cell Broadcast Service	12
Certificate Installer	12

[2] These TAs developed by Huawei are preconfigured in the iTrustee and released with iTrustee v6.1. Each TA has no independent version.

Companion Device Manager	12
Files	20.1.0.1
Basic Daydreams	12
Photo Screensavers	12
Emergency	30.0.0.111
External Storage	12
OsuLogin	12
HTML Viewer	12
ProjectMenuAct	10.00.18
Input Devices	12
Gestural Navigation Bar	1.0
Key Chain	12
Fused Location	12
Work profile setup	12
MmsService	12
MTP Host	12
Network manager	12
Tethering	12
One Time Init	10
PacProcessor	12
Permission controller	12.2.1.100
Phone Services	11.0.0.100
Print Service Recommendation Service	1.3.0
Print Spooler	12
Blocked Numbers Storage	12
Calendar Storage	12.0.0.305
Contacts Storage	30.0.0.106
Download Manager	12
Downloads	12
Media Storage	12
Settings Storage	12
Phone/Messaging Storage	5.0.0.1

User Dictionary	12
ProxyHandler	12
SecureElementService	9.0.2
Call Management	9.0.1.1
Settings	11.1.1.214
Shell	12
Intent Filter Verification Service	1.0
Storage Manager	12
System UI	11.0.0.1
VpnDialogs	12
Live Wallpaper Picker	12
ScreenReader	12.0.0.305
Tablet Clone	12.1.0.160
Scrollshot	13.0.0.115
AirLink	1.0
Huawei AML	12.0.0.100
Navigation Dock	13.0.0.130
Wireless Projection	3.0.0.213
Find Device	11.1.8.300
HMS Services Framework	11.1.1.100
HwAps	12.0.0.2
Software update	103.0.0.644
Huawei Share	12.0.0.020
System Share	12
Huawei Home	13.0.0.103
ProjectMenu	11.1.0.000
Push Service	11.1.9.300
Themes	12.0.1.130
Weather	11.1.2.314
Wi-Fi Direct	11.1.0.160
AppGallery	12.0.1.301
AREngineServer	4.0.1.100

HwAssetSync	11.1.0.310
HwAssetSyncService	11.1.0.309
Multi-Screen Collaboration	13.0.0.022
AudioAccessoryManager	12.0.0.122
BehaviorAuth	12.0.1.303
Betaclub	11.0.0.324
Import via Bluetooth	8.0.0.200
Browser	12.0.2.301
Calculator	13.0.0.120
Calendar	12.0.12.314
Camera	12.0.0.377
HwCameraKit	1.1.6
UserIAMBase	11.0.0.303
Compass	13.0.0.115
Contacts	13.0.1.130
Contacts Sync	13.0.1.120
Scan Business Card	13.0.1.120
ContentSensor	21.1.17.200
Smart Collaboration	3.0.0.123
DEF	1.0
Clock	13.0.0.120
TrustedDeviceAuth	10.0.0.300
HwGroupManager	11.0.0.301
Multi-device management	3.0.0.070
HwDistributedKeyManager	11.1.0.146
HwDistributedPasteboard	3.0.0.165
Multi-Device Collaboration	3.0.0.067
HwEasyGo	10.0.0.102
Email	13.0.2.150
Quick App Center	11.6.1.201
FeatureFramework	10.1.0.401
Huawei Stylus Service	11.1.1.302

XRKit	12.0.0.114
Files	11.1.8.300
GameKit	11.1.0.001
AppAssistant	11.4.2.301
GameCenter	11.3.1.303
Health	12.1.2.308
HiAI Base	22.0.0.236
HUAWEI HiAI Engine	22.0.0.235
AI Voice	21.0.19.202
HiCard	11.0.4.201
Cloud Service	11.1.5.301
Cloud	11.1.8.300
AppAdvisor	12.0.1.301
AI Life Service	12.0.2.306
HUAWEI Video	8.8.60.334
AirSharingClient	3.0.0.224
AI Touch	21.1.17.221
hiview	15.0.0.107
HiViewTunnel	15.0.0.105
HiWrite	12.0.0.308
OTA Engine	102.0.1.186
HwDDMP	11.1.0.573
Smart Diagnosis Basic Services	13.0.0.114
Smart Diagnosis	13.0.0.114
Multi-Window	13.0.0.100
HMS Core	6.4.0.314
HwPanPayService	11.0.0.300
Books	9.1.18.303
Petal Search	11.0.1.332
HwStartupGuide	13.0.0.150
MeeTime Service	12.0.1.11
iAware	3.0.0.18

Device connection service	11.1.1.008
Histen	11.1.0.101
Assistant·TODAY	21.0.25.353
Kids Corner	11.1.0.123GP
LanguageDownloader	11.0.0.126
Location Service	13.0.0.017
Backup	12.1.0.155
Magazine Unlock	11.1.0.015
Petal Maps	1.9.0.309(003)
HwControlCenterSDK	12.0.0.004
MMITest	9.0.0
Motions	13.0.0.120
Multimodal Sensor Data Platform	11.0.1.29
HwAudioKit	1.0.5
hivideoplayengine	1.0.3
Music	12.11.16.383
Member Center	10.0.1.311
Huawei Data Management Services	11.1.1.519
HwNearby	3.0.0.263
Notepad	13.0.3.124
Number Identifier	30.0.0.114
Camera	3.0.0.356
Service Center	23.0.2.200
Health	11.10.4
HiWindow	3.0.1.001
Celia Keyboard	1.0.0.3
Gallery	2.0.5.160
SecurityPrivacyCenter	3.0.0.0
Optimizer	3.0.0.170
OneHopSvcClient	2.0.0.25
Tag service	2.0.0.24
Digital balance	12.0.0.319

Huawei Share	103.0.0.115
My HUAWEI	11.0.0.323
Gallery	11.0.20.171
HAware	12.1.15.0
Huawei Print	3.0.0.005
PrivateSpace	11.1.1.133
Profile Services	12.0.3.210
HwHiAIDSEngine	21.1.49.201
RemotePassword	11.0.0.300
AI Lens	21.1.17.221
Screen recording	13.0.0.120
AI Search	21.1.6.512
Fusion Search Service	11.1.0.462
PrivacyCenter	3.0.0.17
Sample Mgmt	7.12
Password Vault	11.1.0.122
HwSecurityPluginBase	12.0.0.100
SecurityServer	11.0.0.300
Smart screenshots	13.0.0.105
Recorder	12.0.1.135
M-Pencil Floating Menu	12.0.0.307
M-Pen Zone	11.1.0.133
HwSynergy	13.0.0.004
Optimizer	12.0.1.191
Huawei System Services	6.0.0.10
Feature Advisor	13.0.0.124
Tips	13.0.0.125
Smart Unlock	11.1.0.008
Device authentication service	11.0.0.301
TrustedThingsAuth	12.0.0.307
Petal Clip	12.0.3.380
waudio	11.1.0.1

Huawei WebView	12.0.2.302
Link Now	1.0.7.302
HwWifiproBqeService	8.0.0.203
Kika Keyboard	8.6.30
MyScript Calculator 2	2.1.2
Nebo for Huawei	3.2.2
SwiftKey factory settings	2.2.0.314
SwiftKey Keyboard	7.2.6.29
MediaLibrary	1.0.0

Table 1-4 TOE Guidance

TOE Part	Unique identifier	Form of delivery	Delivery method
AGD_OPE	HarmonyOS 3.0 on MatePad Pro-AGD_OPE_PRE v1.2	Electronic document	Huawei's support website
AGD_PRE			
Product Documentation	HUAWEI MatePad Pro User Guide-v0.2		

The TOE configuration consists of the following:

Users can configure screen-lock passwords, network connections, and set and modify user authentication methos (including passwords, patterns and PIN). After user authentication is successful, the user can access TOE to perform operations, such as modifying authentication data, installing mobile applications, querying and modifying mobile application permissions, downloading and installing the update package. For details, see section 3 in *HarmonyOS 3.0 on MatePad Pro-AGD_OPE_PRE*.

1.4.2 Logical Boundaries

This section summarizes the security functions provided by the TOE:

- Authentication of user and Trusted Peer Devices
- Secure communication
- Secure updating of TOE software and apps
- Self-protection and integrity verification of the TOE
- Protecting user data at different levels of security
- Permission management of apps
- Protection against tracking by App developers and Advertisers
- Protection against physical attacks

1.4.2.1 Authentication of user and Trusted Peer Devices

The TOE supports a number of features related to identification and authentication. From a user perspective, except for limited functions such as making phone calls to an emergency number and receiving notifications, a password/PIN/pattern must be correctly entered to unlock the TOE. Also, even when the TOE is unlocked the password/PIN/pattern must be re-entered to change the password/PIN/pattern. When the user uses a password or PIN for authentication, the user enters the password or PIN on the lock screen, the TOE will, by default, display the most recently entered character of the password or PIN briefly or until the user enters the next character in the password, at which point the TOE obscures the character by replacing the character with a dot symbol. If the user unlocks the device using pattern, the TOE will display the pattern for a brief time as it is entered by the user. The TOE defaults to requiring password or PIN to have a minimum of four characters but no more than 32., and the patterns should consist at least 4 and at most 9 connected points, where each point shall only be used once.

The TOE allows other devices to act as a trusted peer device for purposes such as screen sharing and collaborative editing. To be able to do this, these devices must first authenticate themselves, for instance by having the TOE scanning a QR code on the trusted peer device.

1.4.2.2 Secure communication

The TOE supports the use of HTTPs, TLS and 802.11-2012, 802.1X, to secure communications channels between itself and other trusted network devices, protected against unauthorized modification and unauthorized disclosure. These channels can subsequently be used by apps and by the TOE itself for various communication purposes.

1.4.2.3 Secure updating of TOE software and apps

The TOE can update the TOE software by downloading an update from a Trusted Update Source to address known vulnerabilities in a timely manner. The TOE checks integrity and authenticity of the update to determine that it is indeed from a Trusted Update Source and if so, updates itself with that update. The TOE includes mechanisms (i.e., verification of the digital signature of each new image) so that the TOE itself can be updated while ensuring that the updates will not introduce malicious or other unexpected changes in the TOE.

The TOE can update pre-installed non- non-system permission apps and Downloaded Apps by downloading an update from the App Distribution Platform. The TOE checks integrity and authenticity of the update to determine that it is indeed from the App Distribution Platform.

1.4.2.4 Self-protection and integrity verification of the TOE

The TOE is able to protect both itself and other apps against malicious apps who may try to hack into the TOE. The TOE also checks its own integrity every time it starts up to check whether it has been altered. The TOE includes functions to perform self-tests during initial start-up and software/firmware integrity checking so that it can detect when it is failing or may be corrupt. If any self-test fails, the TOE does not go into an operational mode.

1.4.2.5 Protecting user data at different levels of security

The TOE supports classification of user data according to risk levels and usage scenarios, and protect user data to ensure it is accessed by the right person on the right device in the right condition.

1.4.2.6 Permission management of apps

To ensure that apps can only access to user data on the TOE and services provided by the TOE which are essential to their operation and granted by the user and/or by the operating system. The TOE allows a user to restrict the services an application can access. The services that can be restricted on a per-app basis are as follows. Applications prompt the mobile device user to grant permission for the application to use system services when they are installed. Subsequently, mobile device users can perform access control for applications using system services through the Settings interface.

1.4.2.7 Protection against tracking by App developers and Advertisers

The TOE can provide an alias to App developers and Advertisers, so that they have limited tracking of the user. The user is able to replace that alias with another alias to limit this tracking.

1.4.2.8 Protection against physical attacks

The TOE is able to protect keys, data used to derive keys and other sensitive data from being read or modified by physical attacks.

2 Conformance Claims

2.1 Conformance claim

This Security Target is CC Part 2 extended and CC Part 3 conformant, with a claimed Evaluation Assurance Level of EAL2, augmented by ALC_FLR.3.

This Security Target claims conformance to the following specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April, 2017.
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1 Revision 5, April, 2017.

3 Security Problem Definition

3.1 Assets and Threat Agents

Assets to be protected:

- User data assets stored in the TOE, such as
  - user files: photos, videos, etc.;
  - user location: GPS information, location record, etc.;
  - non-TOE account information;
  - user communication: communication records, address books, emails, SMS, chat session, audio/video calls etc.;
  - credentials for other devices and/or services;
  - data collected by sensors: acceleration sensor, blood sugar, body fat ratio, heart rate, blood pressure, etc.;
  - App data: list of installed apps, user data in apps, etc.
- User data assets are grouped in three classes, Low, Medium and High (see P.DATA_CLASSIFICATION), each requiring more stringent protection.
- TSF data: data which is used for the enforcement of operations of security functions, such as configuration data, user authentication data.
- The operating system and apps included in the TOE.

Interfaces of the TOE:

- The local wireless interface(s): these are interface(s) to local wireless networks.
- The wide-area network interface(s): these are interface(s) to wide area networks such as the Internet. This interface generally lies on top of either the local wireless interface or interfaces such as GSM, 3G, 4G.
- The user interface: this interface includes the screen, buttons, speaker etc.
- The physical interface: this is the physical enclosure of the TOE, includes JTAG ports, USB (and similar) ports, charging ports, etc.

The application interface: this is the API that applications can use to interact with the underlying operating system.

3.2 Threat agents and threats

Threat Agents:

- TA.LOCAL: a threat agent in the general vicinity of a TOE when it is used, and therefore has access to the local wireless interface.
- TA.REMOTE: a threat agent with access to the wide-area interface.
- TA.PHYSICAL: a threat agent who has physical access to the TOE, and therefore to both the user interface and the physical interface.
- TA.FLAWAPP: a malicious or poorly programmed app that the user has installed on the TOE and that therefore has access to the application interface, and possibly to the local wireless interface and/or the wide-area network interface.

The present document identifies threats based on each interface defined in clause 5.1 on what nefarious actions each of the threat agents could perform on that interface. Similar threats were subsequently combined into one threat where possible. The threats are identified as below:

T.EAVESDROP – TA.LOCAL, TA.REMOTE or TA.FLAWAPP read communication between the TOE and other entities and thereby access confidential user data assets in transit.

T.SPOOF – TA.LOCAL or TA.REMOTE create a spoofed device or service and wait for the TOE to connect to that device or service. Once the TOE connects to the spoofed device or service the threat agents actively or passively extract user data assets from the TOE.

T.MODIFY-COMMS – TA.LOCAL or TA.REMOTE initiate or intercept communication between the TOE and other entities and thereby modify user data assets in transit.

T.COUNTERFEIT_DEVICE: TA.PHYSICAL or TA.LOCAL attempts to connect to the TOE and thereby gain access to user data assets with a device masquerading as a trusted peer device.

T.IMPERSONATE – TA.PHYSICAL impersonates the legitimate user of the TOE thereby gaining access to the User Data Assets.

T.PHYSICAL – TA.PHYSICAL attempts to gain access to assets by accessing physical interfaces of the TOE.

T.RECOVER_DATA – A user sells their TOE and attempts to remove all user data assets beforehand, but the new user (TA.PHYSICAL) is still able to retrieve some or all of these user data assets.

T.MODIFY_DEVICE - TA.PHYSICAL obtains a TOE, modifies that TOE, and reinserts that TOE into the supply chain. Later on a legitimate user buys this CMD and the modified CMD allows compromise of the assets of that user.

T.FLAWAPP – TA.FLAWAPP attempts to access to user data assets that it should not be able to access and subsequently modify them or export them to third parties. This can include additional data gathered from the TOE sensors (GPS, camera, microphone etc.). TA.FLAWAPP may also conduct attacks against the TOE, which will provide them with additional privileges and the ability to attempts to modify the security behaviour of the TOE and the behaviour or data held in other apps, or attempts to conduct other malicious activities.

T.PERSISTENT – Successful realisation of one of the other threats can lead to a persistent presence on the TOE constituting an ongoing threat in itself. The threat agent associated with the other threat can possibly control the device.

The mapping of threats with interfaces of the TOE is shown in the table below:

Threats	Local wireless interface	Wide-area network interface	User interface	Physical interface	Application interface
T.EAVESDROP	X	X			

T.SPOOF	X	X			
T.MODIFY-COMMS	X	X			
T.COUNTERFEIT_DEVICE	X				
T.IMPERSONATE			X		
T.PHYSICAL				X	
T.RECOVER_DATA			X	X	
T.MODIFY_DEVICE			X	X	X
T.FLAWAPP					X
T.PERSISTENT	X	X	X	X	X

3.3 Organisational Security Policies

P.DATA_CLASSIFICATION –The TOE developer classifies groups of user data assets into at least three different classes, according to policies such as the potential harm that may result to the human user if the user data asset were inappropriately accessed, used, or disclosed:

- Low: user data assets to which unauthorised access is expected to have limited adverse effect on the human user. Low user data assets can only be decrypted on the TOE.
- Medium: user data assets to which unauthorised access can have serious adverse effect on the human user. Medium user data assets can only be decrypted on the TOE following the first successfully user authentication.
- High: user data assets to which unauthorized access can have severe adverse effect on the human user. High user data assets can only be decrypted on the unlocked TOE following a successful user authentication.

3.4 Assumptions

A.APP_DISTRIBUTION_PLATFORM – It is assumed that human user will only install apps which have been downloaded from App Distribution Platform of the TOE developer or OS provider, which have performed best practise security checks on these apps. Security checks include but not limited to:

- detects malicious in-app behaviour,
- conducts privacy disclosure inspections for apps that call, collect or upload sensitive data from user without permission,
- scans apps for the presence of loopholes, vulnerabilities or backdoors,
- verify an App developer has a valid developer certificate.

A.Trusted_Update_Source - It is assumed that the TOE will trust the Trusted Update Source to install secure updates delivered by the update source.

A.Trusted_Cloud_Service - It is assumed that the cloud service provided by the TOE developer is secure and unauthorised parties cannot access to the user data assets on the TOE via remote cloud services.

A.PASSWORD_PIN_PATTERN – It is assumed that human user will not downgrade the authentication of the TOE or choose easily guessable passwords/PINs/Patterns.

A.Certified_UICC – It is assumed that the component employed for WAN connectivity is a certified UICC according to [18].

4 Security Objectives

4.1 Security Objectives for the TOE

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

O.PROTECT_COMMS - The TOE can setup communication channels with other devices/services that are protected against disclosure and allow detection (either by the TOE or by the other device/service) of any modification of data exchanged over these channels. The TOE authenticates such devices/services before allowing communication. The TOE also allows these services/devices to authenticate the TOE.

O.AUTHENTICATED_UPDATES –The TOE supports update of its operating system and apps, and only allows updates of its operating system and apps when these updates are authenticated as being from a trusted source.

O.PROTECT_ASSETS_AT_REST –The TOE ensures that assets are unreadable when not in use, e.g., by encryption.

O.SECURE_WIPE –The TOE is able to make user data assets permanently unreadable.

O.CRITICAL_STORAGE –The TOE provides storage for critical security parameters such that physical attackers are unable to access these parameters.

O.ACCESS_CONTROL –The TOE ensures that apps only gain access to user data assets that they are specifically allowed by the human user or the operating system to have access to.

O.SECURE_BOOT –The TOE, at the start of its boot process, checks the TSF to ensure it has not been tampered with.

O.AUTHENTICATE_USER – The TOE will identify and authenticate the human user of the TOE before allowing that human user has full access to the TOE functionality.

O.CRYPTOGRAPHY – The TOE provides best practice cryptographic functionality (encryption, decryption, signing, signature checking), to implement other security objectives.

O.RANDOMS – The TOE provides best practice random number generation functionality to support O.CRYPTOGRAPHY.

O.DATA_CLASSIFICATION - The TOE supports and encrypts at least three different classes of user data assets:

- Low: user data assets that can only be decrypted on the TOE.
- Medium: user data assets that can only be decrypted on the TOE following the first successfully user authentication.
- High: user data assets that can only be decrypted on the unlocked TOE following a successfully user authentication.

O.AUTHENTICATE_PEER_DEVICE - The TOE allows other devices to authenticate themselves to the TOE and become a trusted peer device.

O.PERSISTENT - The TOE is able to detect and remove malevolent persistent presences from itself.

O.SELF_PROTECTION - The TOE protects itself against apps attempting to modify TSF data and its security behaviour.

O.SEPARATION - The TOE provides a separate security domain for each app, protecting them against unauthorised modification by other apps.

4.2 Security Objectives for the Operational Environment

OE.APP_DISTRIBUTION_PLATFORM –The operational environment ensures that the human user of the TOE is instructed to use App Distribution Platform of the TOE developer or OS provider which performs security checks on these apps.

OE.Trusted_Update_Source - The operational environment ensures the security update delivered by the Trusted Update Source is sufficiently protected from tampering and is secure to be installed by the TOE.

OE.Trusted_Cloud_Service - The operational environment ensures the cloud services are sufficiently protected from unauthorised access.

OE.PASSWORD_PIN_PATTERN – The operational environment ensures that human user of the TOE is instructed not to disable or downgrade the authentication of the TOE or choose easily guessable passwords/PINs/patterns such as aaaa, 1234, birthdates and the like.

OE.Certified_UICC – The operational environment ensures that WAN interface is available to the TOE by means of a certified UICC according to [18].

4.3 Security Objectives Rationale

Threat	**Rationale**
T.EAVESDROP	This threat is countered by O.PROTECT_COMMS that enables protection of the communication channel(s) against disclosure. This objective is supported by O.CRYPTOGRAPHY to encrypt these channels and O.RANDOMS to provide random numbers for key generation.
T.SPOOF	This threat is countered by O.PROTECT_COMMS that ensures that the TOE authenticates devices it connects to.
T.MODIFY-COMMS	This threat is countered by: • O.PROTECT_COMMS that enables protection of the communication channel(s) against disclosure. This objective is supported by O.CRYPTOGRAPHY to encrypt these channels and O.RANDOMS to provide random numbers for key generation. • O.AUTHENTICATED_UPDATES preventing unauthorised updates of any part of the TOE
T.IMPERSONATE	This threat is countered by O.AUTHENTICATE_USER ensuring that only authenticated user can access the device functionality
T.PHYSICAL	This threat is countered by:

	• O.PROTECT_ASSETS_AT_REST ensuring that assets are encrypted (or erased) when not in use thus preventing a physical attacker who directly accesses TOE storage media from reading that data. This objective is supported by O.CRYPTOGRAPHY to encrypt/decrypt this data and O.CRITICAL_STORAGE to securely store the encryption/decryption keys • O.SECURE_BOOT to ensure that the integrity of the TOE is checked every time it boots thus preventing undetected loss of integrity from physical attack. • O.ACCESS_CONTROL to ensure that access to physical interface such as charging interface is controlled by the human user.
T.FLAWAPP	This threat is countered by: • OACCESS_CONTROL preventing the App from gaining access to user data assets which they do not have permission to access, so they cannot be modified or exported. • SELF_PROTECTION stating that the TOE protects itself against apps attempting to alter its security behaviour. • O.SEPARATION stating that the TOE provides a security domain for each app, thereby separating them from other apps.
T.PERSISTENT	This threat is countered by O.PERSISTENT and possibly by O.SECURE_BOOT allowing detection and removal of persistent presences of malevolent entities from the TOE.
T.MODIFY_DEVICE	This threat is countered by O.SECURE_BOOT that is able to detect integrity issues with the TOE and O.CRITICAL_STORAGE that prevents attackers from modifying the keys from O.SECURE_BOOT, which would make it ineffective.
T.COUNTERFEIT_DEVICE	This threat is countered by O.AUTHENTICATE_PEER_DEVICE which forces devices to authenticate in order to become a trusted peer device.
T.RECOVER_DATA	This threat is countered by O.SECURE_WIPE, ensuring that the user data assets are permanently unreadable
P.DATA_CLASSIFICATION	This policy is directly implemented by O.DATA_CLASSIFICATION
A.APP_DISTRIBUTION_PLATFORM	This assumption is directly implemented by OE.APP_DISTRIBUTION_PLATFORM
A.Trusted_Update_Source	This assumption is directly implemented by OE.Trusted_Update_Source.
A.Trusted_Cloud_Service	This assumption is directly implemented by OE.Trusted_Cloud_Service.
A.PASSWORD_PIN_PATTERN	This assumption is directly implemented by OE.PASSWORD_PIN_PATTERN
A.Certified_UICC	This assumption is directly implemented by OE.Certified_UICC

5 Extended Components Definition

5.1 Definition of the family Random Number Generation (FCS_RNG)

5.1.1 Family behaviour

This clause describes the functional requirements for the generation of random numbers, which can be used as secrets for cryptographic purposes or authentication. The security functional components are defined in an additional family (FCS_RNG) of the Class FCS (Cryptographic support). The components address the type of the random number generator and the quality of the random numbers.

5.1.2 Component levelling

FCS_RNG: Generation of Random Numbers	1

FCS_RNG.1, Generation of random numbers, requires that the random numbers meet a defined quality metric.

5.1.3 Management: FCS_RNG.1

There are no management activities foreseen.

5.1.4 Audit: FCS_RNG.1

There are no actions defined to be auditable.

5.1.5 FCS_RNG.1 Random number generation

Hierarchical to: No other components.

Dependencies: No dependencies.

FCS_RNG.1.1 The TSF shall provide a [selection: physical, non-physical true, deterministic, hybrid physical, hybrid deterministic] random number generator.

FCS_RNG.1.2 The TSF shall provide random numbers that meet [assignment: a defined quality metric].

5.2 Definition of the family Cryptographic Key Hierarchy (FCS_CKH)

5.2.1 Family behaviour

This clause describes the functional requirements for a cryptographic key hierarchy, a related set of keys that together protect data. Some keys in the key hierarchy will encrypt the data, while the other keys are used to encrypt and/or derive other keys in the key hierarchy. The sole security functional component is defined in an additional family (FCS_CKH) of the Class FCS (Cryptographic support). The component address the data that is protected, and how the keys in the key hierarchy are derived and protected.

5.2.2 Component levelling

FCS_CKH: Cryptographic key Hierarchy	1

FCS_CKH.1, Cryptographic key Hierarchy, requires definition of the key hierarchy and the data protected by the key hierarchy and how they keys in the key hierarchy are derived and protected.

5.2.3 Management: FCS_CKH.1

There are no management activities foreseen.

5.2.4 Audit: FCS_CKH.1

There are no actions defined to be auditable.

5.2.5 FCS_CKH.1 Cryptographic Key Hierarchy

Hierarchical to: No other components.

Dependencies: FCS_COP.1 Cryptographic Operation.

FCS_CKM.4 Cryptographic Key Destruction

FCS_CKH.1.1 The TSF shall support a key hierarchy for [assignment: list of data to be protected by the key hierarchy].

FCS_CKH.1.2 The TSF shall ensure that all keys in key hierarchy are derived and/or generated according to [assignment: description of how each key in the key hierarchy is derived and/or generated, with which key lengths and according to which standards].

FCS_CKH.1.3 The TSF shall ensure that all keys in the key hierarchy and all data used in deriving the keys in the hierarchy are protected according to [assignment: rules].

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.

Conventions

The following conventions have been applied in this document:

The **human user** defined in the description of the SFRs relates to the authorized TOE user.

Security Functional Requirements – Part 1 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.

- Assignments and selections are marked in bold face font.
- Iterations are marked by appending a suffix to the SFR identification.
- Refinements are marked in bold and italic face font.

6.1 TOE Security Functional Requirements

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

Table 6-1 TOE Security Functional Components

Requirement Class	Requirement Component
FCS: Cryptographic support	FCS_RNG.1 Random number generation
	FCS_COP.1_Update Cryptographic operation
	FCS_COP.1_User Data Assets) Cryptographic operation
	FCS_CKH.1_Low Cryptographic key hierarchy
	FCS_CKH.1_Medium/High Cryptographic key hierarchy
	FCS_CKM.4 Cryptographic key destruction
FDP: User data protection	FDP_ACC.1_Update Subset access control
	FDP_ACF.1_Update Security attribute based access control
	FDP_ACC.1_Permission Subset access control
	FDP_ACF.1_Permission Security attribute based access control
	FDP_ACC.1_User_Data_Asset_Decryption Subset access control
	FDP_ACF.1_User_Data_Asset_Decryption Security attribute based access control
FIA: Identification and authentication	FIA_UAU.1 Timing of authentication
	FIA_UAU.2 User authentication before any action
	FIA_UAU.5 Multiple authentication mechanisms
	FIA_UAU.6 Re-authenticating
	FIA_UAU.7 Protected authentication feedback

	FIA_SOS.1 Verification of secrets
	FIA_AFL.1_Password/PIN/Pattern Authentication failure handling
	FIA_UID.1 Timing of authentication
	FIA_UID.2 User identification before any action
FMT: Security management	FMT_SMF.1_Authentication Specification of Management Functions
	FMT_SMF.1_Permissions Specification of Management Functions
	FMT_SMF.1_Privacy Specification of Management Functions
	FMT_SMF.1_Update Specification of Management Functions
	FMT_MSA.1_Permissions Management of security attributes
	FMT_MSA.3_Permissions Static attribute initialisation
	FMT_MSA.1_User_data_Asset_Decryption Management of security attributes
	FMT_MSA.3_User_data_Asset_Decryption Static attribute initialisation
FPR: Privacy	FPR_PSE.1_Developers Pseudonymity
	FPR_PSE.1_Advertisers Pseudonymity
FPT: Protection of the TSF	FPT_PHP.3 Resistance to physical attack
	FPT_FLS.1 Failure with preservation of secure state
	FPT_TST.1 TSF testing
	FPT_RCV.2 Automated recovery
FTP: Trusted path/channels	
	FTP_ITC.1_HTTPS Inter-TSF trusted channel
	FTP_ITC.1_TLS Inter-TSF trusted channel
	FTP_ITC.1_WLAN Inter-TSF trusted channel

6.1.1 Cryptographic Support (FCS)

6.1.1.1 FCS_RNG.1 Random number generation

FCS_RNG.1.1 The TSF shall provide a [**deterministic**] random number generator.

FCS_RNG.1.2 The TSF shall provide random numbers that meet [**none**].

6.1.1.2 FCS_COP.1_Update Cryptographic operation

FCS_COP.1.1/Update The TSF shall perform [**signature verification of an Update_Package**] in accordance with a specified cryptographic algorithm [**ECDSA**] and cryptographic key sizes [**256**] that meet the following: [**ECDSA schemes using 'NIST curves' P-256 that meet the following: FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Section 5**].

6.1.1.3 FCS_COP.1_User_Data_Assets Cryptographic operation

FCS_COP.1.1/UserDataAssets The TSF shall perform [**encryption and decryption of stored Low, Medium and High user data assets]** in accordance with a specified cryptographic algorithm [**AES-XTS mode**] and cryptographic key sizes [**256-bit**] that meet the following: [**NIST SP 800-38E**].

NOTE: Low, Medium and High user data assets are defined in section 6.1.2.3.

6.1.1.4 FCS_CKH.1_Low Cryptographic key hierarchy

FCS_CKH.1.1/Low The TSF shall support a key hierarchy for [**the data encryption key(s) for Low user data assets**].

FCS_CKH.1.2/Low The TSF shall ensure that all keys in the key hierarchy are derived and/or generated according to [

- **256 bits KEKMK used to protect ClassKEK is derived from DUK (REK) by KBKDF that meets SP 800-108,**
- **256 bits ClassKEK is generated by DRBG SHA256 Hash that meets NIST SP 800-90A,**
- **256 bits DEK used to encrypt Low user data assets is derived from ClassKEK by KBKDF that meets SP 800-108**

] ***ensuring that the key hierarchy uses the DUK directly or indirectly in the derivation of the data encryption key(s) for Low user data assets.***

FCS_CKH.1.3/Low The TSF shall ensure that all keys in the key hierarchy and all data used in deriving the keys in the hierarchy are protected according to [**KEKMK is protected by REK, ClassKEK and DEK are protected by REK-derived key].**

6.1.1.5 FCS_CKH.1_Medium/High Cryptographic key hierarchy

FCS_CKH.1.1 The TSF shall support a key hierarchy for [**the data encryption key(s) for Medium and High user data assets**].

FCS_CKH.1.2 The TSF shall ensure that all keys in the key hierarchy are derived and/or generated according to [

- **256 bits KEKMK used to protect ClassKEK is derived from DUK (REK) by KBKDF that meets SP 800-108,**
- **256 bits password key is derived by PBKDF2 that meets SP 800-132,**
- **256 bits ClassKEKis generated by DRBG DRBG SHA256 Hash that meets NIST SP 800-90A,**
- **256 bits DEK used to encrypt Medium/High user data assets is derived from ClassKEK by KBKDF that meets SP 800-108**

] ***ensuring that the key hierarchy:***

- ***uses the DUK directly or indirectly in the derivation of the data encryption keys for Medium and High user data assets; and***
- ***uses the PIN, password, pattern directly or indirectly in the derivation of the data encryption keys for Medium and High user data assets.***

FCS_CKH.1.3 The TSF shall ensure that all keys in the key hierarchy and all data used in deriving the keys in the hierarchy are protected according to [

KEKMK is protected by REK, ClassKEK and DEK are protected by REK-derived key and password-derived key].

6.1.1.6 FCS_CKM.4 Cryptographic key destruction

FCS_CKM.4.1 The TSF shall destroy [**the data encryption key(s) for Low, Medium, and High user data assets**] in accordance with a specified cryptographic key destruction method [

- **For volatile memory, by a single direct overwrite consisting of [zeroes],**
- **For non-volatile EEPROM, by a single direct overwrite consisting of a random pattern, using the TSF's RNG, followed by a read-verify,**
-]

that meets the following: [**null**].

6.1.2 User Data Protection (FDP)

6.1.2.1 The Update Policy

The Update Policy defines:

- How often the TSF will check whether new Update Packages are available. Update Packages may replace (parts of) the TSF software, but also replace the cryptographic data used to determine the validity (see FCS_COP.1 (Update)) of any future Update Packages.
- The conditions under which the TSF use these Update Packages to update the TOE to a different version.

6.1.2.1.1 FDP_ACC.1_Update Subset access control

FDP_ACC.1.1/Update The TSF shall enforce the **Update Policy** on [

- **Subjects: the TSF,**
- **Objects: the TOE_software, APP, Update_Package,**
- **Operations: Check_For, Receive, Update_the_TOE_software, Update_App].**

6.1.2.1.2 FDP_ACF.1_Update Security attribute based access control

FDP_ACF.1.1/Update The TSF shall enforce the **Update Policy** to objects based on the following: [**the TOE_software [version number], App[version number] Update_Package[version number, signature]**].

FDP_ACF.1.2/Update The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

[

- **The TSF shall be able to Check_For an Update_Package every [24h] and inform the user when a new Update_package is available, and**
- **The TSF shall allow the TSF to Update_the_TOE_software with an Update_Package if and only if:**
  - **the TSF successfully verifies the signature of that Update_Package, and**
  - **the version number of the Update Package is not lower than the version number of the TOE_software, or**
  - **the update is pushed from the Trusted Update Source.**
- **The TSF shall allow the TSF to Update_App with an Update_Package if and only if:**
  - **the TSF successfully verifies the signature of that Update_Package, and**
  - **the version number of the Update_Package is not lower than the version number of the App, or**
  - **the update is pushed from the App Distribution Platform of the TOE developer or OS developer;**
- **the TSF shall Update_the_TOE_software and Update_App in an atomic way.]**

FDP_ACF.1.3/Update The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [**null**].

FDP_ACF.1.4/Update The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **[**

- **the TSF shall not allow any TSF-mediated actions during the_TOE_software updating.]**

NOTE: Failure of correct installation of the update is handled in FPT_FLS.1.

NOTE: Failure of correct installation of the update is handled in FPT_FLS.1.

6.1.2.2 The Permissions Policy

The Permissions Policy defines the access of Apps to objects such as the camera, microphone, and address book and that either users must permit this access, or that this access has been pre-granted (in the case of some Pre-installed Apps). The permission/revocation of this access is defined in FMT_SMF.1(Permissions).

6.1.2.2.1 FDP_ACC.1_Permissions Subset access control

FDP_ACC.1.1/Permissions The TSF shall enforce the **[Permissions Policy]** on **[**

- **Subjects: Downloaded Apps, Pre-installed system permission Apps, Pre-installed non-system permission Apps,**
- **Objects: camera, microphone, GPS, contacts, calendar, call log, stored pictures, text messages, IMEI, SN, the list of installed apps,**
- **Operations: read, write].**

6.1.2.2.2 FDP_ACF.1_Permissions Security attribute based access control

FDP_ACF.1.1/Permissions The TSF shall enforce the **[Permissions Policy]** to objects based on the following: **[**

- **Subjects:**
  - **Downloaded Apps,**
  - **Pre-installed system permission Apps ,**
  - **Pre-installed non-system permission Apps,**
- **Objects:**
  - **Camera**
    - **SA.Camera: Allows an application to use the camera to take photos and record videos.,**
  - **Microphone**
    - **SA.Microphone: Allows an application to use the microphone for audio recording.,**
  - **Location**
    - **SA.Location: Allows an application to obtain the location.,**
  - **Contacts**
    - **SA.Contacts: Allows an application to read, add, remove, and modify contacts.,**

- **Calendar**
  - **SA.Calendar: Allows an application to read, add, remove, or modify calendar events.,**

].

FDP_ACF.1.2/Permissions The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [

The APP has declared the required permissions in the APP configuration file using the reqPermissions attribute, and

- **A Downloaded App is allowed to read from an object if the permission of read from the object has been specifically allowed by the human user, and**
- **A Downloaded App is allowed to write to an object if the permission of write to the object has been specifically allowed by the human user, and**
- **A pre-installed system permission App or pre-installed non-system permission App is allowed to read from an object if:**
  - **This permission of read from the object was granted by the operating system, or**
  - **This permission of read from the object has been specifically allowed by the human user, and**
- **A pre-installed system permission App or pre-installed non-system permission App is allowed to write to an object if:**
  - **This permission of write to the object was granted by the operating system, or**
  - **This permission of write to the object has been specifically allowed by the human user.]**

FDP_ACF.1.3/Permissions The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **[null].**

FDP_ACF.1.4/Permissions The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **[null].**

6.1.2.3 The Data Classification Policy

The TOE supports the following classification of user data assets:

- Low: This data is to be encrypted in such a way that the data can only be decrypted on the TOE itself, when the TOE is successfully powered on, low data is decrypted and can be accessed without user authentication, e.g. alarm;
- Medium: This data is to be encrypted in such a way that the data can only be decrypted on the TOE and when the user is logged in after first successfully authentication, e.g. calendar;
- High: This data is to be encrypted the same as Medium, but additionally it can only be accessed when the screen of the TOE is unlocked, e.g. sensitive health data.

Each of these classes of user data assets is encrypted (see FCS_COP.1_User_Data_Assets) by a Data Encrypting Key (DEK). These DEKs, in turn, can be encrypted themselves by Key Encryption keys (KEKs), which can be encrypted with further KEKs etc.

Each DEK or KEK can be randomly generated, or it can be derived from other keys (such as the DUK) or data (such as the user password), or a combination of random generation and derivation. The whole structure of data and keys used to derive/encrypt a DEK is called a key hierarchy, which typically starts from the DUK and/or user credentials and have derivation/encryption of KEKs in the middle of the hierarchy (see the FCS_CKH requirements)

To prevent attackers from decrypting the user data assets all keys and data used for derivation in the hierarchy are to be protected, by being encrypted with a KEK, or by being discarded after use and re-generated/re-derived when needed, or by being securely stored (see FPT_PHP.3).

When a user wishes to dispose of the TOE and permanently delete all user data assets, this is done by deleting one or more of the appropriate keys from the key hierarchy thereby ensuring that the data can no longer be decrypted as either the key needed for decryption has been deleted or it is not possible to decrypt that key (see FCS_CKM.4).

6.1.2.3.1 FDP_ACC.1_User_Data_Asset_Decryption Subset access control

FDP_ACC.1.1/UserDataAsset The TSF shall enforce the **[User Data Asset Decryption Policy]** on [

- **Subjects: the TSF,**
- **Objects: user data assets,**
- **Operations: decrypt].**

6.1.2.3.2 FDP_ACF.1_User_Data_Asset_Decryption Security attribute based access control

FDP_ACF.1.1/UserDataAsset The TSF shall enforce the **User Data Asset Decryption policy** to objects based on the following: [**TSF, User Data Assets [Low, Medium, High]**].

FDP_ACF.1.2/UserDataAsset The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- **The TSF is allowed to decrypt Low user data assets if and only if the TOE is successfully powered on, and**
- **The TSF is allowed to decrypt Medium user data assets if and only if the TOE is successfully powered on and the user is successfully authenticated during the first authentication after power on , and**
- **The TSF is allowed to decrypt High user data assets if and only if the TOE is successfully powered on and the user is successfully authenticated and the screen of the TOE is not locked.**

NOTE: The phrase “successfully powered on” means that the TOE has successfully booted up and completed all tests required by FPT_TST.1.

FDP_ACF.1.3/Permissions The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **[null].**

FDP_ACF.1.4/Permissions The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **[null].**

6.1.3 Identification and Authentication (FIA)

6.1.3.1 FIA_UAU.1 Timing of authentication

FIA_UAU.1.1 The TSF shall allow [**make an emergency call, receive an incoming phone calls, take screen shots (automatically named and stored internally by the TOE), turn the TOE off, use the flashlight, use a calculator app, use a recorder app, use the world clock, stopwatch and timer functions in the clock app, browse/PIN/remove/like the magazine pictures in the lock-screen, use camera to take photos**] on behalf of the ***human*** user to be performed before the ***human*** user is authenticated.

FIA_UAU.1.2 The TSF shall require the ***human*** user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that ***human*** user.

6.1.3.2 FIA_UAU.2 User authentication before any action

FIA_UAU.2.1 The TSF shall require each ***trusted peer device*** to be successfully authenticated ***by:***

- ***Both TOE and trusted peer device being logged in at the same user account connecting to a trusted server;***
- ***The TOE scanning a QR code generated by the trusted peer device to establish a shared secret;***
- ***The trusted peer device scanning a QR code generated by the TOE to establish a shared secret;***
- ***The human user input a PIN generated by or stored in trusted peer device in the TOE;***
- ***The human user input a PIN generated by or stored in the TOE into trusted peer device;***

before allowing any other TSF-mediated actions on behalf of that ***trusted peer device***.

6.1.3.3 FIA_UAU.5 Multiple authentication mechanisms

FIA_UAU.5.1 The TSF shall provide [**password, PIN, pattern**] to support human user authentication.

FIA_UAU.5.2 The TSF shall authenticate any ***human*** user's claimed identity according to the [**user type in the correct passwords/PIN/pattern**]

6.1.3.4 FIA_UAU.6 Re-authenticating

FIA_UAU.6.1 The TSF shall re-authenticate the ***human*** user under the conditions: [

- **Attempted change of password, PIN or pattern, or**
- **Attempted unlocking of a locked TOE, or**
- **[null].]**

6.1.3.5 FIA_UAU.7 Protected authentication feedback

FIA_UAU.7.1 The TSF shall provide only **feedback that provides at most:**

- **information about the length of the password or PIN**
- **each password/PIN character for a brief moment as it is entered by the user**
- **the pattern for a brief time as it is entered by the user**

to the ***human*** user while the authentication is in progress.

6.1.3.6 FIA_SOS.1 Verification of secrets

FIA_SOS.1.1 The TSF shall provide a mechanism to verify that secrets meet:

- **for passwords and PINs length 4 or more.**
- **for patterns consists of at least 4 and at most 9 connected points, where each point shall only be used once.**

6.1.3.7 FIA_AFL.1_Password/PIN/Pattern Authentication failure handling

FIA_AFL.1.1/Password/PIN/Pattern The TSF shall detect when **[5]** unsuccessful authentication attempts occur related to **[Password, PIN, Pattern].**

FIA_AFL.1.2/Password/PIN/Pattern When the defined number of unsuccessful authentication attempts has been **[met]**, the TSF shall [**progressively lengthen the time to attempt an authentication**].

6.1.3.8 FIA_UID.1 Timing of identification

FIA_UID.1.1 The TSF shall allow [**make an emergency call, receive an incoming phone calls, take screen shots (automatically named and stored internally by the TOE), turn the TOE off, use the flashlight, use a calculator app, use a recorder app, use the world clock, stopwatch and timer functions in the clock app, browse/PIN/remove/like the magazine pictures in the lock-screen, use camera to take photos**] on behalf of the ***human*** user to be performed before the human user is identified.

FIA_UID.1.2 The TSF shall require each ***human*** user to be successfully identified before allowing any other TSF-mediated actions on behalf of that ***human*** user.

6.1.3.9 FIA_UID.2 User identification before any action

FIA_UID.2.1 The TSF shall require each ***trusted peer device*** to be successfully identified before allowing any other TSF-mediated actions on behalf of that ***trusted peer device***.

6.1.4 Security Management (FMT)

6.1.4.1 FMT_SMF.1_Authentication Specification of Management Functions

FMT_SMF.1.1/Authentication The TSF shall be capable of performing the following management functions ***by human users*:**

- **register the initial password/PIN/pattern**
- **change the password/PIN/pattern**

6.1.4.2 FMT_SMF.1_Permissions Specification of Management Functions

FMT_SMF.1.1/Permissions The TSF shall be capable of performing the following management functions ***by human users:***

- **view permissions granted to an App, and**
- **revoke permission from a Downloaded App or Pre-installed non-privileged App to have read and/or write access to an object, and**

- **revoke permission from a Pre-installed privileged App to have read and/or write access to an object if the human user have granted this permission themselves, and**
- **grant permission to an App to have read and/or write access to an object, and**
- **select [charge only mode, file transfer mode] when the TOE is connected via charging interface such as USB interface to a computer/laptop, and**
- **grant/revoke permission to/from a Downloaded App or Pre-installed non-privileged App to have access to accessibility service, and**
- **grant/revoke permission to/from a Downloaded App or Pre-installed non-privileged App to have access to device notification**

6.1.4.3 FMT_SMF.1_Privacy Specification of Management Functions

FMT_SMF.1.1/Privacy The TSF shall be capable of performing the following management functions ***by human users*****: [**

- **change the alias provided to a particular App developer to a new random alias**
- **change the alias provided to Advertisers to a new random alias]**

6.1.4.4 FMT_SMF.1_Update Specification of Management Functions

FMT_SMF.1.1/Update The TSF shall be capable of performing the following management functions ***by human users*****:**

- **initiate an update of the TOE software (if available)**
- **display the version number of the TOE software.**
- **uninstall Downloaded Apps**
- **uninstall Pre-installed non-privileged Apps which are not included in the TOE**

6.1.4.5 FMT_MSA.1_Permissions

FMT_MSA.1.1_Permissions The TSF shall enforce the **Permissions Policy** to restrict the ability to **[query, modify]** the security attributes **[the permission of the Downloaded Apps, Pre-installed system permission, non-privileged Apps and Pre-installed non-system permission Apps]** to **[human users]**.

6.1.4.6 FMT_MSA.1_User_data_Asset_Decryption

FMT_MSA.1.1_User_data_Asset_Decryption The TSF shall enforce the **User Data Asset Decryption policy** to restrict the ability to **[modify]** the security attributes **[DE, CE and ECE storage locations]** to **[human users]**.

6.1.4.7 FMT_MSA.3_Permissions

FMT_MSA.3.1_Permissions The TSF shall enforce the **Permissions Policy** to provide **[restrictive]** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2_Permissions The TSF shall allow **[human users]** to specify alternative initial values to override the default values when an object or information is created.

6.1.4.8 FMT_MSA.3_User_data_Asset_Decryption

FMT_MSA.3.1_User_data_Asset_Decryption The TSF shall enforce the **User Data Asset Decryption policy** to provide **[restrictive]** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2_User_data_Asset_Decryption The TSF shall allow **[none]** to specify alternative initial values to override the default values when an object or information is created.

Application note: the storage locations are fixed by the TOE and can by no means be modified by the human user.

6.1.5 Privacy (FPR)

6.1.5.1 FPR_PSE.1_Developers Pseudonymity

FPR_PSE.1.1/Developers The TSF shall ensure that **App developers** are unable to determine the ***Device ID*** bound to **the TSF**, ***unless the App has been permitted access to the Device ID by the user or this permission was granted by the operating system***.

FPR_PSE.1.2/Developers The TSF shall be able to provide **one** alias of the ***Device ID*** to **each App developer**.

FPR_PSE.1.3/Developers The TSF shall **determine an alias for *each App developer*** and verify that it conforms to the **null**.

6.1.5.2 FPR_PSE.1_Advertisers Pseudonymity

FPR_PSE.1.1/Advertisers The TSF shall ensure that **Advertisers in Apps** are unable to determine the **Device ID** bound to **the TSF, *unless the App has been permitted access to the Device ID by the user or this permission was pre-granted.***

FPR_PSE.1.2/Advertisers The TSF shall be able to provide **one** alias of the ***Device ID*** to **Advertisers by the operating system**.

FPR_PSE.1.3/Advertisers The TSF shall **determine an alias for *Advertisers*** and verify that it conforms to the **null**.

6.1.6 Protection of the TSF (FPT)

6.1.6.1 FPT_PHP.3 Resistance to physical attack

FPT_PHP.3.1 The TSF shall resist **[**
- **read or modify the DUK, and**
- **read or modify [Low class key, High class key, Medium class key], and**
- **modify [hash of the boot code signing key] used to verify the integrity of the TSF in FPT_TST.1, and**
- **modify [certificates] used to verify the integrity and authenticity of updates to the TSF in FCS_COP.1_Update, and**
- **read or modify [Null]]**

to the [**hardware based secure environment of the TSF]** by responding automatically such that the SFRs are always enforced.

6.1.6.2 FPT_FLS.1 Failure with preservation of secure state

FPT_FLS.1.1 The TSF shall preserve a secure state when the following types of failures occur: [**Failure of the Update_the_TSF_Software operation in FDP_ACF.1_Update**].

6.1.6.3 FPT_TST.1 TSF testing

FPT_TST.1.1 The TSF shall run a suite of self-tests [**during initial start-up]** to demonstrate the correct operation of **[the TSF]**.

FPT_TST.1.2 The TSF shall provide authorised users with the capability to verify the integrity of **[TSF data]**.

FPT_TST.1.3 The TSF shall provide authorised users with the capability to verify the integrity of **[TSF]**.

6.1.6.4 FPT_RCV.2 Automated recovery

FPT_RCV.2.1 When automated recovery from [**detection of a malevolent persistent presence by FPT_TST.1**] is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

FPT_RCV.2.2 For [**detection of a malevolent persistent presence by FPT_TST.1**], the TSF shall ensure the return of the TOE to a secure state using automated procedures.

6.1.7 Trusted Path/Channels (FTP)

A TOE usually supports many different communication channels, conforming to different standards, and within these standards, using different settings, resulting in different levels of security.

The ST author shall provide FTP_ITC SFRs for each channel that the ST author claims to be secure. At least one of the four SFRs defined in the present clause shall be claimed in the Security Target.

This does not preclude the TOE providing communication channels based on these standards with less secure settings to communicate with devices that are legacy or have limited secure communication capabilities. As part of the vulnerability analysis the impact these lower settings can have on the overall security of the TOE will be assessed.

6.1.7.1 FTP_ITC.1_HTTPs Inter-TSF trusted channel

FTP_ITC.1.1_HTTPs The TSF shall provide a communication channel between itself and another trusted IT product 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_HTTPs The TSF shall permit **the TSF and another trusted IT product** to initiate communication via the trusted channel.

FTP_ITC.1.3_HTTPs The TSF shall initiate communication via the trusted channel for **communicating with the App Distribution Platform and the upgrade server, and accessing a website.**

The HTTPs channel shall as a minimum:

- Conform to RFC 2818, and

- Use TLS v1.2 to implement HTTPs.

The HTTPS channel is used for downloading App installation packages, Update_Package of TOE software and App from the App distribution platform and upgrade server.

6.1.7.2 FTP_ITC.1_TLS Inter-TSF trusted channel

FTP_ITC.1.1_TLS The TSF shall provide a communication channel between itself and another trusted IT product 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_TLS The TSF shall permit **the TSF and another trusted IT product** to initiate communication via the trusted channel.

FTP_ITC.1.3_TLS The TSF shall initiate communication via the trusted channel for**App that needs an encrypted end-to-end communication via published APIs.**

The TLS channel shall as a minimum:

- Implement TLS v1.2, and
- Use X.509v3 certificates for mutual authentication, and
- Determine validity of the peer certificate by certificate path, expiration date and revocation status according to RFC 5280, and
- Notify the TSF and [not establish the connection] if the peer certificate is deemed invalid, and
- Support the following ciphersuites:
  - TLS_AES_128_GCM_SHA256
  - TLS_AES_256_GCM_SHA384
  - TLS_CHACHA20_POLY1305_SHA256
  - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (in RFC 5289)
  - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (in RFC5289)
  - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (RFC 5289)
  - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (RFC 5289)

The TLS channel is used for end-to-end communications security over networks and is widely used for internet communications. The TLS is used to implement HTTPS and EAP-TLS, and the TOE provides access to TLS via published APIs which are accessible to any application that needs an encrypted end-to-end trusted channel.

6.1.7.3 FTP_ITC.1_WLAN Inter-TSF trusted channel

FTP_ITC.1.1_WLAN The TSF shall provide a communication channel between itself and ***a WLAN 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 ***all wireless access point connections except those involved in the trusted channel establishment itself*.**

The WLAN channel shall as a minimum:

- Implement 802.11-2012, 802.1X and EAP-TLS, and
- Generate symmetric keys according to PRF-384 with keylength 128 bit, and
- Use TLS v1.2 (RFC 5246), and
- Support X.509v3 certificates for mutual authentication, and
- Determine validity of the peer certificate by certificate path, expiration date and revocation status according to RFC 5280, and
- Notify the TSF, and not establish the connection or request application authorization to establish the connection if the peer certificate is deemed invalid, and
- Support the following ciphersuites:
  - TLS_AES_128_GCM_SHA256
  - TLS_AES_256_GCM_SHA384
  - TLS_CHACHA20_POLY1305_SHA256
  - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (in RFC 5289)
  - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (in RFC5289)
  - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (RFC 5289)
  - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (RFC 5289)
- Randomly generate a new MAC-address each time it connects to a different access point.

The WLAN channel is used for data exchange between the TOE and the wireless access point. The TOE establishes an 802.11 tunnel between the TOE and the wireless access point using IEEE 802.1X with EAP-TLS for authentication.

6.2 Security Assurance Requirements

The security assurance requirements consist of EAL2 augmented with ALC_FLR.3.

6.3 Security requirements rationale

6.3.1 Rationale for choosing the SARs

EAL2 is chosen as it provides a good balance between effort expected from the developer and assurance gained.

EAL2 provides assurance by a full security target and an analysis of the SFRs in that ST, using a functional and interface specification, guidance documentation and a basic description of the architecture of the TOE, to understand the security behaviour.

The analysis is supported by independent testing of the TSF, evidence of developer testing based on the functional specification, selective independent confirmation of the developer test results, and a vulnerability analysis (based upon the functional specification, TOE design, security architecture description and guidance evidence provided) demonstrating resistance to penetration attackers with a basic attack potential.

EAL2 also provides assurance through use of a configuration management system and evidence of secure delivery procedures.

EAL2 is augmented by ALC_FLR.3 because CMDs are complex devices that are often subject to many hacking attempts and security investigations, that may result in the discovery of security flaws. ALC_FLR.3 provides assurance that the TOE will be maintained and supported in the future, and require the TOE developer to have procedures to:

- accept suspected security flaws in the TOE from third parties and from their own internal processes
- to process these suspected flaws to determine whether they are actual security flaws
- to correct the actual security flaws
- and to distribute these corrections to TOEs in the field automatically in a timely fashion.

6.3.2 The SFRs meet all the security objectives for the TOE

Security Objective	Rationale
O.PROTECT_COMMS	This objective is achieved by FTP_ITC.1_WLAN, FTP_ITC.1_HTTPs and FTP_ITC.1_TLS, each of which sets up a secure channel with authentication and protection from modification and disclosure
O.AUTHENTICATED_UPDATES	These objectives are achieved by: • FDP_ACC.1_Update and FDP_ACF.1_Update specifying the policy for updating • FCS_COP.1_Update specifying the cryptographic mechanism for checking the validity if an update • FMT_SMF.1_Update, specifying that user may initiate an update • FPT_FLS.1, specifying that failure to correctly update will not lead to an insecure state.
O.PROTECT_ASSETS_AT_REST **O.DATA_CLASSIFICATION**	These objectives are achieved by: • FMT_MSA.1_User_Asset_Data_Decryption, FMT_MSA.3_User_Asset_Data_Decryption , FDP_ACC.1_User_Asset_Data_Decryption and FDP_ACF.1_User_Data_Decryption showing the three classes of user data assets, and when each class may be decrypted • FCS_COP.1_User_Data_Assets, specifying how they are encrypted and decrypted • FCS_CKH.1_Low, FCS_CKH.1_Medium/High describing how the cryptographic keys are derived and protected
O.SECURE_WIPE	This objective is achieved by FCS_CKM.4 specifying that keys from the key hierarchy for each class of data can be deleted on request of the user, making the data unreadable, as all user data is encrypted with these keys
O.CRITICAL_STORAGE	This objective is achieved by FPT_PHP.3 which directly implements the objective.
O.ACCESS_CONTROL	This objective is achieved by FMT_MSA.1_Permissions, FMT_MSA.3_Permissions, FDP_ACC.1_Permissions, FDP_ACF.1_Permissions, specifying that user

	permission is needed and FMT_SMF.1_Permissions allowing users to provide and revoke such permission. This objective is supported by FPR_PSE.1_Developers and FPR_PSE.1_Advertisers allowing App developers and Advertisers the ability to track TOEs, but denying their Apps access to a permanent ID of the TOE, and providing an alias instead. FMT_SMF.1_Privacy allows users to reset the alias.
O.SECURE_BOOT **O.PERSISTENT**	These objectives are achieved by FPT_TST.1, testing the integrity of the TSF, and FPT_RCV.2 specifying the actions to be undertaken (either automatic or by the user) when a malevolent persistent presence is found.
O.AUTHENTICATE_USER	This objective is achieved by: • FMT_SMF.1_Authentication specifying that users can register their authentication data and change this later on. • FIA_UAU.1 specifying that each human user can only perform limited actions before being authenticated and is authenticated to gain full access • FIA_UAU.6 listing the conditions under which a human user is to be re-authenticated • FIA_UAU.5 listing the multiple authentication mechanism a TOE has, and the rules for using these. • FIA_SOS.1 listing the minimum quality requirements for authentication (password/PIN length quality) • FIA_AFL.1_Password/PIN/Pattern specifying what happens when authentication fails repeatedly for each mechanism • FIA_UAU.7 specifying that passwords and PINs are not displayed on the screen when entering them, preventing shoulder surfing. • FIA_UID.1 allowing only certain operations without user identification.
O.CRYPTOGRAPHY	This objective is achieved by the requirements in the FCS chapter. These requirements are also required by various other security objectives (see the other entries in this table)
O.RANDOMS	This objective is achieved by FCS_RNG.1 defining a random generator, whose output meets stringent international standards.
O.AUTHENTICATE_PEER_DEVICE	This objective is achieved by: • FIA_UAU.2 specifying how trusted peer devices are authenticated and what they can do after authentication. • FIA_UID.2 requiring trusted peer devices to be identified before allowing any TSF-mediated actions.
O.SELF_PROTECTION	This objective is achieved by: • FPT_TST.1 specifying that the TSF runs self-testing and integrity verification of the TSF or

	part of the TSF to protect the TSF to be tampered, and • FPT_PHP.3.1 to prevent modification of key(s), hashes of key(s), certificate(s) and/or other data used to verify the integrity of the TSF.
O.SEPARATION	This objective is achieved by: • FDP_ACC.1_Permissions and FDP_ACF.1_Permissions defining security attribute based access control for apps, and • FMT_SMF.1_Permissions specifying management functions for apps.

6.3.3 Dependency analysis

SFR	Dependency	Rationale
FCS_RNG.1	-	
FCS_COP.1_Update	FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1 FCS_CKM.4	Not fulfilled, as the key for checking update packages is inserted in the TOE during production Not fulfilled, as the key for checking update packages is never destroyed
FCS_COP.1_User_Data_Assets	FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1 FCS_CKM.4	Fulfilled by FCS_CKH.1. This replaces FCS_CKM.1 Fulfilled by FCS_CKM.4
FCS_CKH.1_Low	FCS_COP.1 FCS_CKM.4	Fulfilled FCS_COP.1_User Data Assets Fulfilled by FCS_CKM.4
FCS_CKH.1_Medium/High	FCS_COP.1 FCS_CKM.4	Fulfilled FCS_COP.1_User Data Assets Fulfilled by FCS_CKM.4
FCS_CKM.4	FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1	Fulfilled by FCS_CKH.1. This replaces FCS_CKM.1 as it generates a key hierarchy rather than an individual key like FCS_CKM.1
FDP_ACC.1_Update	FDP_ACF.1	Fulfilled by FDP_ACF.1_Update
FDP_ACF.1_Update	FDP_ACC.1 FMT_MSA.3	Fulfilled by FDP_ACC.1_Update Not fulfilled, as the only security attribute is the version of the TSF and this is inserted during production and the rules for modifying it (copying the version number of the Update Package is already described in the SFR).
FDP_ACC.1_Permissions	FDP_ACF.1	Fulfilled by FDP_ACF.1_Permissions
FDP_ACF.1_Permissions	FDP_ACC.1 FMT_MSA.3	Fulfilled by FDP_ACC.1_Permissions Not necessary, as the security attributes of the object do not change
FDP_ACC.1_User_Data_Asset_ Decryption	FDP_ACF.1	Fulfilled by FDP_ACF.1_User Data Asset Encryption
FDP_ACF.1_User_Data_Asset_ Decryption	FDP_ACC.1 FMT_MSA.3	Fulfilled by FDP_ACC.1_User Data Asset Encryption Not necessary, as the security attributes Low, Medium and High do not change.
FIA_UID.1	-	

FIA_UID.2	-	
FIA_UAU.1	FIA_UID.1	Fulfilled by FIA_UID.1
FIA_UAU.2	FIA_UID.2	Fulfilled by FIA_UID.2
FIA_UAU.5	-	
FIA_UAU.6	-	
FIA_UAU.7	FIA_UAU.1	Fulfilled by FIA_UAU.1
FIA_SOS.1	-	
FIA_AFL.1_Password/PIN/ Pattern	FIA_UAU.1	Fulfilled by FIA_UAU.1
FMT_SMF.1_Authentication	-	
FMT_SMF.1_Permission	-	
FMT_SMF.1_Privacy	-	
FMT_SMF.1_Update		
FMT_SMR.1		
FMT_MSA.1_Permissions	FMT_SMR.1	Fulfilled by FDP_ACC.1_Permissions & FMT_SMF.1_Permissions There is only one role which is the human user, hence FMT_SMR.1 is not defined.
FMT_MSA.3_Permissions	FMT_MSA.1 FMT_SMR.1	Fulfilled by FMT_MSA.1_Permissions There is only one role which is the human user, hence FMT_SMR.1 is not defined.
FMT_MSA.1_User_data_Decryption	[FDP_ACC.1 or FDP_IFC.1] FMT_SMR.1 FMT_SMF.1	Fulfilled by FDP_ACC.1_User Data Asset Decryption & FMT_SMF.1_User Data Asset Decryption There is only one role which is the human user, hence FMT_SMR.1 is not defined. There are no management functions accessible to TOE users, hence FMT_SMF.1 is not defined
FMT_MSA.3_ User_data_Decryption	FMT_MSA.1 FMT_SMR.1	Fulfilled by FMT_MSA.1_User Data Asset Decryption There is only one role which is the human user, hence FMT_SMR.1 is not defined.
FPR_PSE.1_Developers	-	
FPR_PSE.1_Advertisers	-	
FPT_FLS.1	-	
FPT_PHP.3	-	
FPT_TST.1	-	
FPT_RCV.2	AGD_OPE.1	Fulfilled by EAL2
FTP_ITC.1_HTTPs	-	
FTP_ITC.1_TLS	-	
FTP_ITC.1_WLAN	-	

7 TOE Summary Specification

This chapter describes the security functions:

- Authentication of user and Trusted Peer Devices
- Secure communication
- Secure updating of TOE software and apps
- Self-protection and integrity verification of the TOE
- Protecting user data at different levels of security
- Permission management of apps
- Protection against tracking by App developers and Advertisers
- Protection against physical attacks

7.1 Authentication of user and Trusted Peer Devices

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

FIA_UAU.1: The TOE will allow a user to do the following things before successfully authenticating the user: make an emergency call, receive an incoming phone calls, take screen shots (automatically named and stored internally by the TOE), turn the TOE off, use a flashlight, use a calculator app, use a recorder app, use the world clock, stopwatch and timer functions in the clock app, browse/PIN/remove/like the magazine pictures in the lock-screen, use camera to take photos. Beyond those actions, a user cannot perform any other actions other than observing notifications displayed on the lock screen until after successfully authenticating. The TOE allows a user to configure, on a per application basis, whether notifications will be displayed.

FIA_UAU.2: The TOE can provide authentication services for devices that have the same HUAWEI ID as the TOE. Each device logged in to a HUAWEI ID generates a public-private key pair using elliptic curve cryptography as the authentication credential, and applies for public key attestation from the Huawei Cloud server.

After TOE and another peer device establish connections using WiFi, and TOE and the peer device use the same HUAWEI ID to log in to the HUAWEI Cloud server , device authentication and session key exchange will be performed based on the public and private key pairs of the two devices. After they successfully authenticate each other, TOE builds a security channel between devices, which uses the session key provided by "User authentication" service to encrypt the transmitted data.

"Authentication of Trusted Peer Devices" service also provides the device authentication based on the P2P binding. In this scenario, the user needs to manually create shared secret information between two devices. This can be achieved by, for example, scanning the other device's QR code, entering the random PIN code displayed on the other device "User authentication" service executes the password authenticated key exchange (PAKE) security

protocol based on the shared secret information created by the user, and then establishes a secure communication channel. In addition, the devices generate their own public-private key pairs using elliptic curve cryptography as authentication credentials, exchange the public key credential with the peer device over the secure communication channel, and store the credentials.

FIA_UAU.5: The TOE, allows the user to authenticate to the device's lock screens using a password/PIN/pattern, but does not support hybrid authentication (password + other authentication factor).

FIA_UAU.6: The TOE requires the user to enter their password/PIN/pattern in order to unlock the TOE. Additionally the TOE requires the user to confirm their current password, PIN or pattern when attempting change of password, PIN or pattern. Only after entering their current user password, PIN or pattern can the user then elect to change their password, PIN or pattern.

FIA_UAU.7: The TOE allows the user to enter the user's password, PIN or pattern from the lock screen to unlock the TOE. When the user uses a password or PIN for authentication, the user enters the password or PIN on the lock screen, the TOE will, by default, display the most recently entered character of the password or PIN briefly or until the user enters the next character in the password, at which point the TOE obscures the character by replacing the character with a dot symbol. If the user unlocks the device using pattern, the TOE will display the pattern for a brief time as it is entered by the user.

FIA_SOS.1: The TOE defaults to requiring password or PIN to have a minimum of four characters but no more than 32, and the patterns should consist at least 4 and at most 9 connected points, where each point shall only be used once.

FIA_AFL.1_Password/PIN/Pattern: The TOE maintains, for each user, the number of failed logins since the last successful login. If the password authentication failed login attempt reaches the maximum number (5 times), the TOE will progressively lengthen the time to attempt an authentication. As the number of the password authentication failed login attempt increases, the password authentication will be disabled for a longer time. Turning the device power off does not affect the count of failed login attempts maintained by the TOE, because this count is persistent stored in the non-volatile storage.

FIA_UID.1 & FIA_UID.2: The TOE ensures that human users and trusted peer devices are identified before any TSF-mediated action by assigning a unique identifier to each authorized subject. The human user who unlocks the TOE in order to use the device function is identified by a unique User ID and authenticated by password/PIN/pattern. The TOE will allow a user to do the following things before the user is identified: make an emergency call, receive an incoming phone calls, take screen shots (automatically named and stored internally by the TOE), turn the TOE off, use a flashlight, use a calculator app, use a recorder app, use the world clock, stopwatch and timer functions in the clock app, browse/PIN/remove/like the magazine pictures in the lock-screen, use camera to take photos. In addition, the trusted peer device must first establish the trusted path with the TOE and then be successfully identified and authenticated by logging in the same HUAWEI ID or establish a P2P binding relationship with the TOE before being able to perform any operation.

FMT_SMF.1_Authentication specifying that users can register their authentication data and change this later on. Users can register screen password/PIN/pattern through user interface "Settings->Biometrics & password->Lock screen password", and users can change the screen password/PIN/pattern later on through user interface "Settings->Biometrics & password->Change lock screen password".

7.2 Secure communication

FTP_ITC.1_HTTPS & FTP_ITC.1_TLS & FTP_ITC.1_WLAN: The TOE provides secured (encrypted and mutually authenticated) communication channels between itself and wireless access point through the use of 802.11-2012, 802.1X, and EAP-TLS. The HTTPS is used to establish a secure communication channel between the TOE and the server when downloading an App installation package, updating an application, downloading an operating system update package, or accessing a website, and to protect the confidentiality and integrity of transmitted data. The TOE permits itself and applications to initiate communication via the trusted channel, and the TOE initiates communication via the trusted channel for connection to a wireless access point. The TLS is used to implement HTTPS and EAP-TLS, and the TOE provides access to TLS via published APIs which are accessible to any application that needs an encrypted end-to-end trusted channel.

The TOE establishes an 802.11 tunnel between the TOE and the network infrastructure using IEEE 802.1X with EAP-TLS for authentication, TLS 1.2 is used to support the implementation of EAP-TLS. The TOE's wpa_supplicant provides the PRF384 for WPA2 derivation of 128-bit AES Temporal Key (using the HMAC implementation provided by BoringSSL) and employs its BoringSSL AES-256 DRBG when generating random values used in the EAP-TLS and 802.1X 4-way handshake. To implement the WPA2 security protocol, there is a Wi-Fi module integrated in the TOE's chipset. The module has passed the Wi-Fi Alliance certification.

The TOE provides mobile applications access to TLS via published APIs. These APIs are accessible to any application that needs an encrypted end-to-end trusted channel. The TOE provides mobile applications the use of TLS version 1.2 (compliant with RFC 2818) including support for the selected ciphersuites in the selections in section 6.1.7.3. TLS certificates are required for establishing TLS encryption channels, X.509v3 certificates are used for mutual authentication. In addition, determine validity of the peer certificate by certificate path, expiration date and revocation status according to RFC 5280. If the peer certificate is deemed invalid, the TSF will not establish the connection.

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. TLS v1.2 is used to implement HTTPS. The TOE does not establish the connection if the peer certificate is deemed invalid, and notify to the application that making the HTTPS connection.

7.3 Secure updating of TOE software and App

FDP_ACC.1_Update and FDP_ACF.1_Update:

The TOE's user interface provides a method to query the current version of the TOE software/firmware (OS version, kernel version, and build number) and hardware (model number). Additionally, the TOE provides users the ability to review the currently installed apps (including 3rd party 'built-in' applications) and their version. Users can proactively check whether new update packages are available, or TOE can automatically check whether new update packages are available by period, the conditions for checking the update package include the version number, device name, and cust version number matching. The TOE shall be able to check an update package every 24h and inform the user when a new Update package is available.

Before updating the TOE software, the TOE establishes a secure communication channel with the upgrade server (a Trusted Update Source) and performs mutual authentication. Then, TOE downloads the update package from the upgrade server and use the preconfigured certificate to verify the integrity and source of the update package, and to determine that it is indeed from a Trusted Update Source. At the same time, the TOE submits an update authorization request to the upgrade server. The upgrade server determines whether to allow the TOE to perform the update according to the preset security update policy, e.g. the version number of the Update Package is not lower than the version number of the TOE software, the update is pushed from the Trusted Update Source etc.

The TOE verifies all updates to the TOE software using a public key chaining ultimately to the Huawei root public key. As described above, the hash (SHA-256) of the Huawei root public key is in e-fuses within the application processor. The HarmonyOS on the TOE requires that all applications bear a valid signature before HarmonyOS install the application.

FCS_COP.1_Update: The TOE will check the device name and version number to make sure download the correct package. Then, TOE verifies the signature of the update package using cryptographic algorithm RSA-2048, to checks integrity and authenticity of the update package to determine that it is indeed from a Trusted Update Source.

FMT_SMF.1_Update: Users can initiate an update through TOE's user interface “Settings->System & updates->Software update”.

All applications (apps) downloaded by a human user and pre-installed non-system permission apps which can be uninstalled by the human user. The user can uninstall an app using any of the following methods:

- When using the Standard style home screen, touch and hold the app icon on the home screen, then touch Uninstall and follow the instructions.
- When using the Drawer style home screen, touch and hold the app icon in the drawer, then touch Uninstall and follow the instructions.
- Go to Settings > Apps > Apps, touch the app, then touch Uninstall.

FPT_FLS.1: TOE has a power-fail protection mechanism, if the system restarts due to user press the power key to reset or low battery power, TOE will enter the update mode again. The failures information of a updating will be stored in a partition named misc, and some recovery actions will base on this information.

Before the update, if the battery is low, a call is being made, or the data space is not large enough, TSF will not allow a reboot into recovery mode for the update. Shutting down the TOE is not allowed during a recovery update, and if this is enforced by the user, the TSF has power-failure protection and will continue the update after a reboot until the update is successful.

If an exception occurs during the update, you can use the eRecovery to rectify the fault, or go to an after-sales service center to enter recovery mode and perform the update through the USB port.

7.4 Self-protection and integrity verification of the TOE

These objectives are achieved by FPT_TST.1, testing the integrity of the TSF, and FPT_RCV.2 specifying the actions to be undertaken (either automatic or by the user) when a malevolent persistent presence is found.

FPT_TST.1: The TOE ensures a secure boot process in which the TOE verifies the integrity of the boot chain during initial start-up. The secure boot process consists of two phases: Secboot and verified boot.

Secboot verification is used to verify the integrity of bootrom(PBL), xLoader(XBL), fastboot(ABL), TEE(QSEE), TA(Trust application) and external module firmware image(eg, modem). They use the Huawei root public key whose hash resides in the processor's internal fuses. The PBL is stored in immutable read-only memory; it is literally part of the fabric of each chip. The image cannot be physically altered. The verification process is as follows: the PBL verifies the XBL image. If the signature is valid, the XBL loads ABL. The ABL verifies the TEE image. During the TEE init-up, the TEE verifies the TA image and modem firmware image. In the secboot phase, the system will restart or the subsystem fail to be started if integrity check fails.

Verified boot verification is a native Android integrity check mechanism that checks the integrity of VB images (including boot, ramdisk, dtbo, recovery, system, and vendor). The key used for verifying the verified boot is stored in the VBmeta image instead of the root public key. The key used for verifying the integrity of the next-level image is obtained by reading the VBmeta image data. The next-level image may still be a VBmeta image. Therefore, the key used for verifying the final target image is verified. The key used to verify the first VBmeta image is hardcoded in the fastboot image, and the integrity of the fastboot image is ensured by the secboot process. Each VB image has its own key, which facilitates customization and expansion by operating system developers, mobile phone manufacturers. When verified boot verifies small partitions such as boot and dtbo that are read only once, the entire content is directly loaded to the memory, the hash value is calculated, and the hash value is compared with the hash value prestored in Vbmeta image. If the values are inconsistent, an alarm is reported during the start-up and the system cannot be accessed. For large partitions (such as file system-related partitions) that cannot be installed in the memory, use the hash tree mode and use the built-in dm-verify driver of Android to perform continuous verification during system running. If the calculated hash value at a certain time is not as expected, The system does not use the corresponding data and Android reports an error.

FPT_RCV.2: The TSF would enter recovery mode and use usb update to recovery the system; Or enter eRecovery mode, the user can download new version and recovery the system by them self.

7.5 Protecting user data at different levels of security

FDP_ACC.1_User_Asset_Data Decryption and FDP_ACF.1_User_Data_Decryption:

The TOE supports the following classification of user data assets:

- Low: This data is to be encrypted in such a way that the data can only be decrypted on the TOE itself, when the TOE is successfully powered on, low data is decrypted and can be accessed without user authentication, e.g. alarm;
- Medium: This data is to be encrypted in such a way that the data can only be decrypted on the TOE and when the user is logged in after first successfully authentication, e.g. calendar;
- High: This data is to be encrypted the same as Medium, but additionally it can only be accessed when the screen of the TOE is unlocked, e.g. sensitive health data. High user data assets are allowed to access only when the TOE is successfully powered on and the user has unlocked the device by supplying their credentials (e.g. passcode, PIN, pattern) and the TOE is not locked.

Each user of the device has three storage locations available to applications:
Device Encrypted (DE) storage, which is a storage location available after the user has powered on the device.
Credential Encrypted (CE) storage, which is the default storage location and only available after the user has unlocked the device.
Enhanced Credential Encrypted (ECE) storage is available the same as CE, but additionally it can only be accessed when the screen of the TOE is unlocked.
The storage locations of DE, CE, and ECE are fixed in the TOE, and cannot be changed by user. For third-party applications, the default storage location is CE, and can also call APIs to store data in DE or ECE

FMT_MSA.1_User_Asset_Data Decryption & FMT_MSA.3__User_Asset_Data Decryption:

Low, Medium, and High user data is determined by the data storage location. The TOE supports three types of user data protection:

- Device Encrypted (DE) storage, which is a storage location available after the user has powered on the device.
- Credential Encrypted (CE) storage, which is the default storage location and only available after the user has unlocked the device.
- Enhanced Credential Encrypted (ECE) storage is available the same as CE, but additionally it can only be accessed when the screen of the TOE is unlocked.

The storage locations of DE, CE, and ECE are fixed in the TOE, and cannot be changed by user. For third-party applications, the default storage location is CE, can also call APIs to store data in DE or ECE.

FCS_COP.1_User_Data_Assets specifying how they are encrypted and decrypted.

The TOE provides AES-256 XTS encryption of all data (which includes both user data and TSF data) stored on the TOE through the File Classification Encryption Mechanism (which provide symmetric scheme to protect protected data or sensitive data) on the device. The File Classification Encryption Mechanism generates a unique file encryption key for each individual file, encrypts it with AES-256-XTS and stores it on the internal storage.

The TOE provides mobile applications the ability to mark data (which may be a key) as sensitive or not. An application can determine whether sensitive data should remain encrypted in this manner or if it should be decryypted and re-encrypted (such as by a symmetric key for better performance). Applications can use this to securely receive data while the work environment is locked (such as an email application). By default, the attachment of e-mails is marked as the sensitive data, and other user data are marked as the protected data by system. Applications can specify files as sensitive data by filename or directory name.

FCS_CKH.1_Low & FCS_CKH.1_Medium/High: showing the key hierarchy for the data encryption key(s) for Low, Medium and High user data assets.

The TOE employs a key hierarchy that protects all DEKs and KEKs by encryption with either the DUK or by the DUK and password derived KEK. The TOE supports a Device Unique Key (DUK) stored in a series of 256-bit fuses within the main (application) processor. Requests for encryption or decryption chaining to the DUK are only accessible through the Trusted Execution Environment, or TEE (TrustZone). The TEE does not allow direct access to the DUK but provides services including encryption/decryption of other keys using the DUK and derivation of keys from the DUK through a KDF function. The TEE does not allow trusted applications to use the DUK for encryption or decryption, only the ability to derive a

KEK from the DUK. The TOE includes a TEE application that calls into the TEE in order to derive a KEK from the 256-bit DUK/fuse value and then only permits use of the derived KEK for encryption and decryption as part of the TOE key hierarchy.

All DEKs and KEKs stored in non-volatile memory are encrypted using AES-GCM with 256-bit keys.

The File-Based Encryption (FBE) mechanism is used to protect all user data and enterprise data contained in TOE Each File is encrypted by a unique DEK. Further, the DEKs are derived from the ClassKEK. There are several kinds of ClassKEKs: the ClassKEKs for low, Medium and High user data assets. These ClassKEKs are randomly generated in TEE environment. The ClassKEKs are protected with AES_256 algorithm in GCM mode by the user factor (passsord/PIN) or the device factor (REK) or both. The ClassKEKs can only be decrypted by RIGHT persons in RIGHT devices.

FCS_RNG.1: The implemented physical random number generator together with the associated post processing, and the Random Number Generator can be suitable for generation of signature key pairs, generation of session keys for symmetric encryption mechanisms, random padding bits, generation of seeds for DRNGs.

FCS_CKM.4 specifying that keys from the key hierarchy for each class of data may be deleted on request of the user, making the data unreadable. The TOE destroys cryptographic keys when they are no longer in use by the system. The exceptions to this are public keys (that protect the boot chain and software updates) and the DUK, which are never cleared. DUK never leaves the processor. KEKs and DEKs stored in RAM during use are destroyed by a zero overwrite. Keys stored in Flash (i.e. eMMC) are destroyed by cryptographic erasure through a reset factory setting operation. In this operation, the user data partition is reformatted. Once reset factory setting executed, all keys (of course including the KEKs and DEKs used for FBE) of the TOE are considered cryptographically erased. The TOE provides a TOE Wipe function to erase all encrypted DEKs and KEKs that are stored in the Flash by the reset factory setting operation. In this operation, the TOE will reformat the partition. Upon completion of reformatting the Flash partition holding user data, the TOE will perform a power-cycle.

7.6 Permission management of apps

FDP_ACC.1_Permissions, FDP_ACF.1_Permissions, FMT_SMF.1_Permissions, FMT_MSA.1_Permissions & FMT_MSA.3_Permissions:

TOE uses the sandbox mechanism to manage permissions of applications. By default, applications have only limited access to system resources. In situations where function extension is needed, however, an application may need to access data (including personal data) or capabilities outside its own sandbox. In this case, the system or related applications must share their data or capabilities by providing interfaces explicitly. To prevent misuse or malicious use of the data or capabilities, an access control mechanism, namely, application permission, is therefore introduced.

Objects protected by permissions are classified into data and capabilities. Data mainly includes personal data (such as photos, contacts, calendars, and locations), device data (such as device identifiers, cameras, and microphones), and application data. Capabilities mainly include device capabilities (such as making phone calls, sending SMS messages, and accessing the Internet) and application capabilities (such as displaying a pop-up window and creating a shortcut). Sensitive permissions include those for accessing personal data (such as photos, contacts, calendars, local phone numbers, and SMS messages) and those for

performing sensitive operations (such as using the camera and microphone, making phone calls, and sending SMS messages).

Applications must have certain permissions to call APIs to access data or perform operations. Based on how sensitive the data is and what security threats the operations may pose to the system, HarmonyOS classifies these permissions into different types and defines the available scope and grant mode of each permission to protect data.

The available scope of a permission can be any of the following:

- all: intended for all applications
- signature: intended for signed applications. Signature permission requires consistent signature between the permission definer and the permission user. After the permission is declared in config.json, it can be used if the permission management module verifies that the signatures of the permission definer and the permission user are the same.
- privileged: intended for preset privileged applications. Privileged permissions that can be requested by pre-configured privileged applications in the system
- restricted: intended for applications with restricted certificates. Restricted permissions can be used only after the request for the permissions is approved.

The grant mode of a permission can be either of the following:

- system_grant: This mode applies to non-sensitive permissions. If you declare such a permission in the config.json file, the system automatically grants the permission to your application when users install your application.
- user_grant: This mode applies to sensitive permissions. When your application requests such a permission, the system presents a pop-up window to ask for user consent.

If grant mode of the permission is system_grant, the permission is automatically granted when the application is installed.

If grant mode of the permission is user_grant, the permission can be used only after being authorized by the user (in a dialog box or on the permission setting page). Users can view the reason in the reason field to determine whether to grant the permission.

When an application tries to access a service, the service checks whether the application has required permissions. In case the required permissions are missing, the access will be denied.

Whenever user wants to revoke a permission from an application, he can open permission setting page of this application in system settings, then select the permission to revoke it on this page. Since then, the application will not be allowed to access the data or capability protected by the revoked permission.

When the TOE is connected via USB charging interface to a computer/laptop, the TOE will pop up a function list. The list will provide all available USB functions, including [Charge only mode, file transfer mode]. The TOE user can pick one function within the list and the TOE will perform accordingly.

More detail can be found in this web link:

https://developer.harmonyos.com/en/docs/documentation/doc-guides/security-permissions-overview-0000000000029883.

7.7 Protection against tracking by App developers and Advertisers

FPR_PSE.1_Developers and FPR_PSE.1_Advertisers only being able to see an alias instead of the real device ID, and **FMT_SMF.1_Privacy** allowing users to reset these aliases to a new random alias.

Device identifiers cannot be associated: For downloaded applications, developers and advertisers cannot collect constant device identifiers, such as IMEI and SN. The collected device identity and user account identity of the device cannot be bound to the real identity of the user. For pre-installed system permission applications, if the application has been permitted access to the Device ID by the operating system, the application developer or advertiser in application is able to the constant Device ID.

Device identifier processing: The TOE provides the anonymization and pseudonymization capabilities to process identifiers obtained by developers and advertisers. Device identifiers obtained by developers and advertisers cannot be bound to the real identities of devices or users. During application installation, the TOE generates an alias of the device ID for each application, and the alias is not associated with any existing identifiers. For applications that are running on the same device, each of them has its own device ID alias. In addition, the TOE provides advertisers with a non-permanent alias of the device ID, that is, the Ad ID. The Ad ID is irrelevant to a constant device ID.

Resetting the device identifier: Users can actively reset the Ad ID by going to Settings > Privacy > Ads and privacy or Settings > Security & privacy > More settings > Device identifier, or restoring the device to its factory settings. . After the identifier is reset, the Ad ID obtained by the advertiser is different from the previous identifier. The user can reset the alias of the device ID by uninstalling and reinstalling the application or restoring the device to its factory settings. After the alias of the device ID is reset, the developer obtains a different alias of the device ID.

7.8 Protection against physical attacks

FPT_PHP.3: The HiSilicon Kirin 9000E SoC maintains a “User Key Derivation Key” (UKDK) which is a 256 bit random number that acts as root encryption key (REK). The UKDK is a hardware-protected value that is never allowed to leave the hardware confines. This implies that operations with the UKDK are performed by the hardware. The TOE generates the UKDK/fuse value during manufacturing using the AES-256 CTR DRBG provided by the SoC PRNG that is seeded by a hardware entropy source. The UKDK is never exported or updated after manufacturing. Requests for encryption or decryption chaining to the UKDK are only accessible through the Trusted Execution Environment, or TEE (TrustZone). The TEE does not allow direct access to the UKDK but provides services including encryption/decryption of other keys using the UKDK and derivation of keys from the UKDK through a KDF function.

The High/Medium/Low Class KEK can be decrypted and used only in the TEE. Never leave the TEE, which protects against physical attacks.

The certificate used to verify the integrity and source of the update package is preconfigured in the system image before delivery and protected by the secure boot mechanism. If the certificate is modified, the integrity of system image will be destroyed. And, the TOE will fail to be started and cannot be accessed. The signature algorithm used to verify the integrity and source of the update package is SHA256 with ECDSA.

8 Abbreviations, Terminology and References

8.1 Abbreviations

Abbreviation	Abbreviation from
CMD	Consumer Mobile Device
DEK	Data Encryption Key
DUK	Device Unique Key
GCF	Global Certification Forum
GPS	Global Positioning System
IMEI	International Mobile Equipment Identity
KEK	Key Encryption Key
NFC	Near Field Communication
OEM	Original Equipment Manufacturer
OS	Operating System
PIN	Personal Identification Number
PP	Protection Profile
QR	Quick Response
SoC	System-on-Chip
SN	Serial Number
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functionality
UI	User Interface

8.2 Definition of terms

Best practice cryptography: cryptography that is suitable for the corresponding use case and has no indications of a feasible attack with current readily available techniques.

Consumer mobile device: user customizable device utilizes a mobile operating system, with wireless internet connectivity, high computation power and rich user interface, used for various purposes by the individual owner.

NOTE: Smartphones and tablets are typical consumer mobile devices.

Device ID: unique identity of consumer mobile device, which is not resettable, such as IMEI and SN.

Security problem: statement, which in a formal manner defines the nature and scope of the security that the TOE is intended to address [1].

Security objective: statement of an intent to counter identified threats and/or satisfy identified organization security policies and/or assumptions [1].

Security requirement: requirement, stated in a standardized language, which is meant to contribute to achieving the security objectives for a Target of Evaluation [1].

Target of Evaluation: A set of software, firmware and/or hardware possibly accompanied by guidance [1].

TOE Security Functionality: combined functionality of all hardware, software, and firmware of a TOE that must be relied upon for the correct enforcement of the security requirements.

TOE software: operating system and pre-installed privileged apps which are updated together from a Trusted Update Source

Trusted Peer Device: A device with a special status, for purposes such as screen sharing, file sharing, moving the entire content from an old device to a new device, and using the wireless connection from one device for both.

Trusted Update Source: a central repository from which updates to the TOE software can be downloaded. This repository is typically managed by the TOE developer / OEM and authenticity and integrity of updates are typically guaranteed by digitally signing the updates.

8.3 References

[1] Common Criteria for Information Technology Security Evaluation Part 1: Introduction and General Model version 3.1 revision 5, CCMB-2017-04-01, April 2017.

[2] Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Components, version 3.1 revision 5, CCMB-2017-04-02, April 2017.

[3] Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Components, version 3.1 revision 5, CCMB-2017-04-03, April 2017.

[4] SOG-IS Crypto Evaluation Scheme Agreed Cryptographic Mechanisms, SOG-IS Crypto Working Group, version 1.2, January 2020.

[5] IETF RFC 2818: "HTTP over TLS, IETF".

[6] IETF RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2".

[7] IETF RFC 5280: "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[8] IETF RFC 5288: "AES Galois Counter Mode (GCM) Cipher Suites for TLS".

[9] IETF RFC 5289: "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)".

[10] IETF RFC 8446: "The Transport Layer Security (TLS) Protocol Version 1.3".

[11] Bluetooth® SIG: "Bluetooth Core Specification, v4.0"

[12] Bluetooth® SIG: "Bluetooth Core Specification, v4.2"

[13] Bluetooth® SIG: "Bluetooth Core Specification, v5.2"

[14] IEEE 802.11-2016 - IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications

[15] IEEE 802.1X-2020 - IEEE Standard for Local and Metropolitan Area Networks--Port-Based Network Access Control

[16] IETF RFC 5216: “The EAP-TLS Authentication Protocol”

[17] Common Methodology for Information Technology Security Evaluation, version 3.1 revision 5, CCMB-2017-04-04, April 2017.

[18] Embedded UICC for Consumer Devices Protection Profile, version 1.0, 05-June-2018