Google Pixel Devices on Android 13 – Security Target

Version: 1.0
January 23, 2023

Google LLC

1600 Amphitheatre Parkway
Mountain View, CA 94043
USA

Table of Contents

1 Security Target Introduction 5
1.1 Security Target Reference 6
1.2 TOE Reference 6
1.3 TOE Overview 6
1.4 TOE Description 7
1.4.1 TOE Architecture 8
1.4.2 TOE Documentation 11
2 Conformance Claims 12
2.1 Conformance Rationale 13
3 Security Objectives 14
3.1 Security Objectives for the Operational Environment 14
4 Extended Components Definition 15
5 Security Requirements 18
5.1 TOE Security Functional Requirements 18
5.1.1 Security Audit (FAU) 21
5.1.2 Cryptographic Support (FCS) 26
5.1.3 User Data Protection (FDP) 34
5.1.4 Identification and Authentication (FIA) 36
5.1.5 Security management (FMT) 43
5.1.6 Protection of the TSF (FPT) 51
5.1.7 TOE Access (FTA) 54
5.1.8 Trusted Path/Channels (FTP) 54
5.2 TOE Security Assurance Requirements 56
5.2.1 Development (ADV) 56
5.2.2 Guidance Documents (AGD) 57
5.2.3 Life-cycle support (ALC) 58
5.2.4 Tests (ATE) 59
5.2.5 Vulnerability assessment (AVA) 59
6 TOE Summary Specification 61
6.1 Security audit 61
6.2 Cryptographic support 63
6.3 User data protection 73
6.4 Identification and authentication 78
6.5 Security management 83
6.6 Protection of the TSF 84
6.7 TOE access 90
6.8 Trusted path/channels 90
6.9 Live-cycle support 91

List of Tables

Table 1 - TOE Common Attributes 6
Table 2 - Evaluated Devices 7
Table 3 - Technical Decisions 13
Table 4 - PP_MDF_V3.3 Extended Components 16
Table 5 - MOD_BT_V1.0 Extended Components 16
Table 6 - MOD_WLANC_V1.0 Extended Components 16
Table 7 - MOD_BIO_V1.1 Extended Components 17
Table 8 - PKG_TLS_V1.1 Extended Components 17
Table 9 - PP_MDF_V3.3 Extended Assurance Components 17
Table 10 - TOE Security Functional Components 21
Table 11 - PP_MDF_V3.3 Audit Events 23
Table 12 - MOD_BT_V1.0 Audit Events 24
Table 13 - MOD_WLANC_V1.0 Audit Events 25
Table 14 - Security Management Functions 49
Table 15 - Bluetooth Security Management Functions 49
Table 16 - WLAN Security Management Functions 50
Table 17 - Assurance Components 56
Table 18 - Audit Event Table References 62
Table 19 - Asymmetric Key Generation 63
Table 20 - Wi-Fi Alliance Certificates 64
Table 21 - Salt Nonces 65
Table 22 - BoringSSL Cryptographic Algorithms 66
Table 23 - LockSettings Service KDF Cryptographic Algorithms 66
Table 24 - Titan Security Chipsets 66
Table 25 - Titan M2 Hardware Cryptographic Algorithms 67
Table 26 - Titan M Hardware Cryptographic Algorithms 67
Table 27 - Wi-Fi Chipsets 67
Table 28 - Google Tensor G2 Hardware Cryptographic Algorithms 68
Table 29 - Google Tensor Hardware Cryptographic Algorithms 68
Table 30 - Snapdragon 765 Hardware Cryptographic Algorithms 68
Table 31 - Snapdragon 730 Hardware Cryptographic Algorithms 69
Table 32 - Functional Categories 75
Table 33 - Supported Biometric Modalities 78
Table 34 - Fingerprint False Accept/Reject Rates 80
Table 35 - Power-up Cryptographic Algorithm Known Answer Tests 88
Table 36 - Security Update Period 92

List of Figures

Figure 1 - Password Conditioning 70

1 Security Target Introduction

This section identifies the Security Target (ST) and Target of Evaluation (TOE) identification, ST conventions, ST conformance claims, and the ST organization. The TOE consists of the Pixel Devices on Android 13 provided by Google LLC. The TOE is being evaluated as a Mobile Device.

The Security Target contains the following additional sections:

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

Acronyms and Terminology

AA	Assurance Activity
BAF	Biometric Authentication Factor
BMFPS	Back-Mounted Fingerprint Sensor
CC	Common Criteria
CCEVS	Common Criteria Evaluation and Validation Scheme
NFC	Near Field Communication
PBFPS	Power-Button Fingerprint Sensor
PP	Protection Profile
SAR	Security Assurance Requirement
SEE	Secure Execution Environment
SFR	Security Functional Requirement
ST	Security Target
TEE	Trusted Execution Environment
TOE	Target of Evaluation
UDFPS	Under-Display Fingerprint Sensor
UI	User Interface

Conventions

The following conventions have been applied in this document:

- Security Functional Requirements – Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.

- Iteration: allows a component to be used more than once with varying operations. In the ST, iteration is indicated by a parenthetical number placed at the end of the component. For example FDP_ACC.1(1) and FDP_ACC.1(2) indicate that the ST includes two iterations of the FDP_ACC.1 requirement.
- Assignment: allows the specification of an identified parameter. Assignments are indicated using bold and are surrounded by brackets (e.g., [**assignment**]). Note that an assignment within a selection would be identified in italics and with embedded bold brackets (e.g., [***[selected-assignment]***]).
- Selection: allows the specification of one or more elements from a list. Selections are indicated using bold italics and are surrounded by brackets (e.g., [***selection***]).
- Refinement: allows the addition of details. Refinements are indicated using bold, for additions, and strike-through, for deletions (e.g., "… all objects …" or "… some big things …").

- Other sections of the ST – Other sections of the ST use bolding to highlight text of special interest, such as captions.

1.1 Security Target Reference

ST Title	Google Pixel Devices on Android 13 – Security Target
ST Version	1.0
ST Date	January 23, 2023

1.2 TOE Reference

TOE Identification	Google Pixel Devices on Android 13
TOE Developer	Google LLC
Evaluation Sponsor	Google LLC

1.3 TOE Overview

The Target of Evaluation (TOE) is Google Pixel Phones on Android 13. All the included phones have the following information in common:

Android OS Version	Architecture	Security Patch Level
Android 13	ARMv8	January 2023

Table 1 - TOE Common Attributes

The TOE consists of the following devices:

Product	Model #	SoC	Kernel
Google Pixel 7 Pro	GVU6C, G03Z5, GQML3	Google Tensor G2	5.10
Google Pixel 7	GE2AE, GFE4J, GP4BC	Google Tensor G2	5.10
Google Pixel 6 Pro	GF5KQ, G8V0U, GLU0G	Google Tensor	5.10
Google Pixel 6	GR1YH, GB7N6, G9S9B	Google Tensor	5.10
Google Pixel 6a	GX7AS, GB62Z, G1AZG, GB17L	Google Tensor	5.10
Google Pixel 5a-5G	G4S1M	Qualcomm Snapdragon™ 765	4.19

Product	Model #	SoC	Kernel
Google Pixel 5	GD1YQ, GTT9Q, G5NZ6	Qualcomm Snapdragon™ 765	4.19
Google Pixel 4a-5G	G025E/I/H, G6QU3	Qualcomm Snapdragon™ 765	4.19
Google Pixel 4a	G025J/M/N	Qualcomm Snapdragon™ 730	4.14

Table 2 - Evaluated Devices

Google manufacturers some of the phones in multiple variants, differing in size (the designations vary, from the "base" device not having any, and other models having something like "a" or "Pro") or build materials (entry and premium). The only differences between variants of a given device are build materials, screen type and size, battery capacity, cameras, RAM and Flash storage. The Pro phones are normally physically larger and have larger screen sizes, battery capacity and possibly RAM, while the a phones may have a smaller screen or different build materials. Storage options vary with each release and may be selectable by the customer of the device at purchase.

The TOE allows basic telephony features (make and receive phone calls, send and receive SMS/MMS messages) as well as advanced network connectivity (allowing connections to both 802.11 Wi-Fi and 2G/3G/4G LTE/5G mobile data networks). The TOE supports using client certificates to connect to access points offering WPA2/WPA3 networks with 802.1x/EAP-TLS, or alternatively connecting to cellular base stations when utilizing mobile data.

The TOE offers mobile applications an Application Programming Interface (API) including that provided by the Android framework and supports API calls to the Android Management APIs.

1.4 TOE Description

The TOE is a mobile device to support enterprises and individual users alike.

Some features and settings must be enabled for the TOE to operate in its evaluated configuration. The following features and settings must be enabled:

1. Enable a password screen lock
2. Do not use Smart Lock
3. Enable encryption of Wi-Fi and Bluetooth secrets (NIAP mode DPM API)
4. Do not use USB debugging
5. Do not allow installation of applications from unknown sources
6. Enable security logging
7. Disable 'Usage & Diagnostic' settings
8. Disable Captive Portal Checking
9. Loaded applications must be implemented utilizing the NIAPSEC library

Doing this ensures that the phone complies with the PP_MDF_V3.3 requirements. Please refer to the Admin Guide on how to configure these settings and features.

1.4.1 TOE Architecture

The TOE provides a rich API to mobile applications and provides users installing an application the option to either approve or reject an application based upon the API access that the application requires (or to grant applications access at runtime).

The TOE also provides users with the ability to protect Data-At-Rest with AES encryption, including all user and mobile application data stored in the user's data partition. The TOE uses a key hierarchy that combines a REK with the user's password to provide protection to all user and application cryptographic keys stored in the TOE.

The TOE includes an additional hardware security chip (Titan M or Titan M2, depending on the device, collectively called Titan security chip) that provides dedicated key storage[1]. The TOE makes this secure, hardware key storage available to mobile applications through the StrongBox extensions to the Android Keystore. Currently, the StrongBox extension is not used for any system keys, but remains an option for applications to use should they desire the protections it provides.

Finally, the TOE can interact with a Mobile Device Management (MDM) system (not part of this evaluation) to allow enterprise control of the configuration and operation of the device so as to ensure adherence to enterprise-wide policies (for example, restricting use of a corporate provided device's camera, forced configuration of maximum login attempts, pulling of audit logs off the TOE, etc.) as well as policies governing enterprise applications and data (in a an employee-owned device [BYOD] scenario). An MDM is made up of two parts: the MDM agent and MDM server. The MDM Agent is installed on the phone as an administrator with elevated permissions (allowing it to change the relevant settings on the phone) while the MDM Server is used to issue the commands to the MDM Agent. Neither portion of the MDM process is considered part of the TOE, and therefore not being directly evaluated.

The TOE includes several different levels of execution including (from lowest to highest): hardware, a Trusted Execution Environment, Android's bootloader, and Android's user space, which provides APIs allowing applications to leverage the cryptographic functionality of the device.

1.4.1.1 Physical Boundaries

The TOE's physical boundary is the physical perimeter of its enclosure. The TOE runs Android as its software/OS, executing on the Google Tensor or Qualcomm Snapdragon processors. The TOE does not include the user applications that run on top of the operating system, but does include controls that limit application behavior. Further, the device provides support for downloadable MDM agents to be installed to limit or permit different functionality of the device. There is no built-in MDM agent pre-installed on the device.

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

1.4.1.2 Logical Boundaries

This section summarizes the security functions provided by the Pixel phones:

[1] Does not apply to the Pixel 6 Pro/6/6a

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

1.4.1.2.1 Security audit

The TOE implements the SecurityLog and logcat that are stored in a circular memory buffers. An MDM agent can read/fetch the logs (both the SecurityLog and logcat) and then handle appropriately (potentially storing the log to Flash or transmitting its contents to the MDM server). These log methods meet the logging requirements outlined by FAU_GEN.1 in PP_MDF_V3.3. Please see the Security audit section for further information and specifics.

1.4.1.2.2 Cryptographic support

The TOE includes multiple cryptographic libraries with CAVP certified algorithms for a wide range of cryptographic functions including the following: asymmetric key generation and establishment, symmetric key generation, encryption/decryption, cryptographic hashing and keyed-hash message authentication. These functions are supported with suitable random bit generation, key derivation, salt generation, initialization vector generation, secure key storage, and key and protected data destruction. These primitive cryptographic functions are used to implement security protocols such as TLS, EAP-TLS, and HTTPS and to encrypt the media (including the generation and protection of data and key encryption keys) used by the TOE. Many of these cryptographic functions are also accessible as services to applications running on the TOE allowing application developers to ensure their application meets the required criteria to remain compliant to PP_MDF_V3.3 standards.

1.4.1.2.3 User data protection

The TOE controls access to system services by hosted applications, including protection of the Trust Anchor Database. Additionally, the TOE protects user and other sensitive data using encryption so that even if a device is physically lost, the data remains protected. The TOE's evaluated configuration supports Android Enterprise profiles to provide additional separation between application and application data belonging to the Enterprise profile. Please see the Admin Guide for additional details regarding how to set up and use Enterprise profiles.

1.4.1.2.4 Identification and authentication

The TOE supports a number of features related to identification and authentication. From a user perspective, except for FCC mandated (making phone calls to an emergency number) or non-sensitive functions (e.g., choosing the keyboard input method or taking screen shots), a password (i.e., Password Authentication Factor) must be correctly entered to unlock the TOE. Also, even when unlocked, the TOE

requires the user re-enter the password to change the password. Passwords are obscured when entered so they cannot be read from the TOE's display and the frequency of entering passwords is limited and when a configured number of failures occurs, the TOE will be wiped to protect its contents. Passwords can be constructed using upper and lower cases characters, numbers, and special characters and passwords up to 16 characters are supported. The TOE can also be configured to utilize a biometric authentication factor (fingerprints), to unlock the device (this only works after the password has been entered after the device powers on).

The TOE can also serve as an 802.1X supplicant and can both use and validate X.509v3 certificates for EAP-TLS, TLS, and HTTPS exchanges.

1.4.1.2.5 Security management

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

1.4.1.2.6 Protection of the TSF

The TOE implements a number of features to protect itself to ensure the reliability and integrity of its security features. It protects particularly sensitive data such as cryptographic keys so that they are not accessible or exportable through the use of the application processor’s hardware. The TOE disallows all read access to the Root Encryption Key (REK) and retains all keys derived from the REK within its Trusted Execution Environment (TEE). Application software can only use keys derived from the REK by reference and receive the result.

The TOE also provides its own timing mechanism to ensure that reliable time information is available (e.g., for log accountability). It enforces read, write, and execute memory page protections, uses address space layout randomization, and stack-based buffer overflow protections to minimize the potential to exploit application flaws. It also protects itself from modification by applications as well as to isolate the address spaces of applications from one another to protect those applications.

The TOE includes functions to perform self-tests and software/firmware integrity checking so that it might detect when it is failing or may be corrupt. If any self-tests fail, the TOE will not go into an operational mode. It also includes mechanisms (i.e., verification of the digital signature of each new image) so that the TOE itself can be updated while ensuring that the updates will not introduce malicious or other unexpected changes in the TOE. Digital signature checking also extends to verifying applications prior to their installation as all applications must have signatures (even if self-signed).

1.4.1.2.7 TOE access

The TOE can be locked, obscuring its display, by the user or after a configured interval of inactivity. The TOE also has the capability to display an administrator specified (using the TOE’s MDM API) advisory message (banner) when the user unlocks the TOE for the first use after reboot.

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

1.4.1.2.8 Trusted path/channels

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

1.4.2 TOE Documentation

Google Pixel Phones on Android 13 Administrator Guidance Documentation, Version 1.0, January 20, 2022 **[Admin Guide]**

2 Conformance Claims

This TOE is conformant to the following CC specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017.
  - Part 2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1, Revision 5, April 2017.
  - Part 3 Extended
- PP-Configuration for Mobile Device Fundamentals, Biometric enrollment and verification – for unlocking the device, Bluetooth, and WLAN Clients, Version 1.0, 11 October 2022 (CFG_MDF-BIO-BT-WLANC_V1.0)
  - The PP-Configuration includes the following components:
    - Base-PP: Protection Profile for Mobile Device Fundamentals, Version 3.3, 12 September 2022 (PP_MDF_V3.3)
    - PP-Module: collaborative PP-Module for Biometric enrolment and verification - for unlocking the device - [BIOPP-Module], Version 1.1, September 12, 2022 (MOD_BIO_V1.1)
    - PP-Module: PP-Module for Bluetooth, Version 1.0, 15 April 2021 (MOD_BT_V1.0)
    - PP-Module: PP-Module for WLAN Clients, Version 1.0, 31 March 2022 (MOD_WLANC_V1.0)
- Package Claims:
  - Functional Package for Transport Layer Security (TLS), Version 1.1, 12 February 2019 (PKG_TLS_V1.1)
- Technical Decisions as of January 5, 2023:

TD Number	Applied	Rationale
TD0442 – PKG_TLS_V1.1	Yes	
TD0469 – PKG_TLS_V1.1	No	Product does not have a TLS server
TD0499 – PKG_TLS_V1.1	Yes	
TD0513 – PKG_TLS_V1.1	Yes	
TD0588 – PKG_TLS_V1.1	No	Product does not have a TLS server
TD0600 – MOD_BT_V1.0	No	Using later PP-Configuration
TD0640 – MOD_BT_V1.0	Yes	
TD0650 – MOD_BT_V1.0	Yes	
TD0667 – MOD_WLANC_V1.0	Yes	
TD0671 – MOD_BT_V1.0	Yes	
TD0674 – MOD_WLANC_V1.0	Yes	
TD0677 – PP_MDF_V3.3	Yes	
TD0685 – MOD_BT_V1.0	Yes	
TD0689 – PP_MDF_V3.3	Yes	
TD0700 – MOD_BIO_V1.1	Yes	

TD Number	Applied	Rationale
TD0703 – MOD_WLANC_V1.0	Yes	
TD0704 – PP_MDF_V3.3	Yes	
TD0707 – MOD_BT_V1.0	Yes	
TD0710 – MOD_WLANC_V1.0	Yes	
TD0714 – MOD_BIO_V1.1	Yes	

Table 3 - Technical Decisions

2.1 Conformance Rationale

The ST conforms to the PP_MDF_V3.3/MOD_BT_V1.0/MOD_WLANC_V1.0/MOD_BIO_V1.1/PKG_TLS_V1.1 with Use Case 4 selected for PP_MDF_V3.3. For simplicity, this shall be referenced as MDF/BT/WLANC/BIO/TLS. As explained previously, the security problem definition, security objectives, and security requirements have been drawn from the PP.

3 Security Objectives

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

In general, the MDF/BT/WLANC/BIO/TLS has defined Security Objectives appropriate for mobile device and as such are applicable to the Google Pixel Devices on Android 13 TOE.

3.1 Security Objectives for the Operational Environment

PP_MDF_V3.3
OE.CONFIG TOE administrators will configure the Mobile Device security functions correctly to create the intended security policy.

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

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

OE.DATA_PROPER_USER Administrators take measures to ensure that mobile device users are adequately vetted against malicious intent and are made aware of the expectations for appropriate use of the device.

MOD_WLANC_V1.0
OE.NO_TOE_BYPASS Information cannot flow between external and internal networks located in different enclaves without passing through the TOE.

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

MOD_BIO_V1.1
OE.Protection The TOE environment shall provide the SEE to protect the TOE, the TOE configuration and biometric data during runtime and storage.

4 Extended Components Definition

All of the extended requirements in this ST are drawn from the MDF/BT/WLANC/BIO/TLS. The MDF/BT/WLANC/BIO/TLS defines the following extended requirements and since they are not redefined in this ST, the MDF/BT/WLANC/BIO/TLS should be consulted for more information about those CC extensions.

Extended SFR	Name
FCS_CKM_EXT.1	Cryptographic Key Support
FCS_CKM_EXT.2	Cryptographic Key Random Generation
FCS_CKM_EXT.3	Cryptographic Key Generation
FCS_CKM_EXT.4	Key Destruction
FCS_CKM_EXT.5	TSF Wipe
FCS_CKM_EXT.6	Salt Generation
FCS_HTTPS_EXT.1	HTTPS Protocol
FCS_IV_EXT.1	Initialization Vector Generation
FCS_RBG_EXT.1	Random Bit Generation
FCS_SRV_EXT.1	Cryptographic Algorithm Services
FCS_SRV_EXT.2	Cryptographic Algorithm Services
FCS_STG_EXT.1	Cryptographic Key Storage
FCS_STG_EXT.2	Encrypted Cryptographic Key Storage
FCS_STG_EXT.3	Integrity of Encrypted Key Storage
FDP_ACF_EXT.1	Security Access Control for System Services
FDP_ACF_EXT.2	Security Access Control for System Resources
FDP_DAR_EXT.1	Protected Data Encryption
FDP_DAR_EXT.2	Sensitive Data Encryption
FDP_IFC_EXT.1	Subset Information Flow Control
FDP_STG_EXT.1	User Data Storage
FDP_UPC_EXT.1/APPS	Inter-TSF User Data Transfer Protection (Applications)
FDP_UPC_EXT.1/BLUETOOTH	
FIA_AFL_EXT.1	Authentication Failure Handling
FIA_PMG_EXT.1	Password Management
FIA_TRT_EXT.1	Authentication Throttling
FIA_UAU_EXT.1	Authentication for Cryptographic Operation
FIA_UAU_EXT.2	Timing of Authentication
FIA_X509_EXT.1	Validation of Certificates
FIA_X509_EXT.2	X509 Certificate Authentication
FIA_X509_EXT.3	Request Validation of Certificates
FMT_MOF_EXT.1	Management of Security Functions Behavior
FMT_SMF_EXT.2	Specification of Remediation Actions
FMT_SMF_EXT.3	Current Administrator
FPT_AEX_EXT.1	Application Address Space Layout Randomization
FPT_AEX_EXT.2	Memory Page Permissions
FPT_AEX_EXT.3	Stack Overflow Protection
FPT_AEX_EXT.4	Domain Isolation
FPT_AEX_EXT.5	Kernel Address Space Layout Randomization
FPT_BBD_EXT.1	Application Processor Mediation

Extended SFR	Name
FPT_JTA_EXT.1	JTAG Disablement
FPT_KST_EXT.1	Key Storage
FPT_KST_EXT.2	No Key Transmission
FPT_KST_EXT.3	No Plaintext Key Export
FPT_NOT_EXT.1	Self-Test Notification
FPT_TST_EXT.1	TSF Cryptographic Functionality Testing
FPT_TST_EXT.2/PREKERNEL	TSF Integrity Checking (Pre-Kernel)
FPT_TST_EXT.2/POSTKERNEL	TSF Integrity Checking (Post-Kernel)
FPT_TUD_EXT.1	Trusted Update: TSF Version Query
FPT_TUD_EXT.2	TSF Update Verification
FPT_TUD_EXT.3	Application Signing
FPT_TUD_EXT.6	Trusted Update Verification
FTA_SSL_EXT.1	TSF- and User-initiated Locked State
FTP_ITC_EXT.1	Trusted Channel Communication

Table 4 - PP_MDF_V3.3 Extended Components

Extended SFR	Name
FCS_CKM_EXT.8	Bluetooth Key Generation
FIA_BLT_EXT.1	Bluetooth User Authorization
FIA_BLT_EXT.2	Bluetooth Mutual Authentication
FIA_BLT_EXT.3	Rejection of Duplicate Bluetooth Connections
FIA_BLT_EXT.4	Secure Simple Pairing
FIA_BLT_EXT.6	Trusted Bluetooth User Authorization
FIA_BLT_EXT.7	Untrusted Bluetooth User Authorization
FMT_SMF_EXT.1/BT	Specification of Management Functions
FPT_KST_EXT.1 [modified]	Key Storage
FPT_KST_EXT.2 [modified]	No Key Transmission
FTP_BLT_EXT.1	Bluetooth Encryption
FTP_BLT_EXT.2	Persistence of Bluetooth Encryption
FTP_BLT_EXT.3/BR	Bluetooth Encryption Parameters (BR/EDR)
FTP_BLT_EXT.3/LE	Bluetooth Encryption Parameters (LE)

Table 5 - MOD_BT_V1.0 Extended Components

Extended SFR	Name
FCS_TLSC_EXT.1/WLAN	TLS Client Protocol (EAP-TLS for WLAN)
FCS_TLSC_EXT.2/WLAN	TLS Client Support for Supported Groups Extension (EAP-TLS for WLAN)
FCS_WPA_EXT.1	Supported WPA Versions
FIA_PAE_EXT.1	Port Access Entity Authentication
FIA_X509_EXT.1/WLAN	X.509 Certificate Validation
FIA_X509_EXT.2/WLAN	X.509 Certificate Authentication (EAP-TLS for WLAN)
FIA_X509_EXT.6	X.509 Certificate Storage and Management
FPT_TST_EXT.3/WLAN	TSF Cryptographic Functionality Testing (WLAN Client)
FTA_WSE_EXT.1	Wireless Network Access

Table 6 - MOD_WLANC_V1.0 Extended Components

Extended SFR	Name
FIA_MBE_EXT.1	Biometric enrolment

Extended SFR	Name
FIA_MBE_EXT.2/Fingerprint	Quality of biometric templates for biometric enrolment
FIA_MBV_EXT.1/BMFPS	Biometric verification
FIA_MBV_EXT.1/UDFPS	Biometric verification
FIA_MBV_EXT.2/Fingerprint	Quality of biometric samples for biometric verification
FPT_BDP_EXT.1	Biometric data processing
FPT_PBT_EXT.1	Protection of biometric template

Table 7 - MOD_BIO_V1.1 Extended Components

Extended SFR	Name
FCS_TLS_EXT.1	TLS Protocol
FCS_TLSC_EXT.1	TLS Client Protocol
FCS_TLSC_EXT.2	TLS Client Support for Mutual Authentication
FCS_TLSC_EXT.4	TLS Client Support for Renegotiation
FCS_TLSC_EXT.5	TLS Client Support for Supported Groups Extension

Table 8 - PKG_TLS_V1.1 Extended Components

Extended SAR	Name
ALC_TSU_EXT.1	Timely Security Updates

Table 9 - PP_MDF_V3.3 Extended Assurance Components

5 Security Requirements

This section defines the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) that serve to represent the security functional claims for the Target of Evaluation (TOE) and to scope the evaluation effort.

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

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

5.1 TOE Security Functional Requirements

The following table identifies the SFRs that are satisfied by Google Pixel Devices on Android 13 TOE.

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

Requirement Class	PP	Requirement Component
	PP_MDF_V3.3	FCS_COP.1/CONDITION Cryptographic operation
	PP_MDF_V3.3	FCS_HTTPS_EXT.1 HTTPS Protocol
	PP_MDF_V3.3	FCS_IV_EXT.1 Initialization Vector Generation
	PP_MDF_V3.3	FCS_RBG_EXT.1 Random Bit Generation
	PP_MDF_V3.3	FCS_SRV_EXT.1 Cryptographic Algorithm Services
	PP_MDF_V3.3	FCS_SRV_EXT.2 Cryptographic Algorithm Services
	PP_MDF_V3.3	FCS_STG_EXT.1 Cryptographic Key Storage
	PP_MDF_V3.3	FCS_STG_EXT.2 Encrypted Cryptographic Key Storage
	PP_MDF_V3.3	FCS_STG_EXT.3 Integrity of Encrypted Key Storage
	PKG_TLS_V1.1	FCS_TLS_EXT.1 TLS Protocol
	PKG_TLS_V1.1	FCS_TLSC_EXT.1 TLS Client Protocol
	MOD_WLANC_V1.0	FCS_TLSC_EXT.1/WLAN TLS Client Protocol (EAP-TLS for WLAN)
	MOD_WLANC_V1.0	FCS_TLSC_EXT.2/WLAN TLS Client Support for Supported Groups Extension (EAP-TLS for WLAN)
	PKG_TLS_V1.1	FCS_TLSC_EXT.4 TLS Client Support for Renegotiation
	PKG_TLS_V1.1	FCS_TLSC_EXT.2 TLS Client Support for Mutual Authentication
	PKG_TLS_V1.1	FCS_TLSC_EXT.5 TLS Client Support for Supported Groups Extension (EAP-TLS for WLAN)
	MOD_WLANC_V1.0	FCS_WPA_EXT.1 Supported WPA Versions
FDP: User Data Protection	PP_MDF_V3.3	FDP_ACF_EXT.1 Security Access Control for System Services
	PP_MDF_V3.3	FDP_ACF_EXT.2 Security Access Control for System Resources
	PP_MDF_V3.3	FDP_DAR_EXT.1 Protected Data Encryption
	PP_MDF_V3.3	FDP_DAR_EXT.2 Sensitive Data Encryption
	PP_MDF_V3.3	FDP_IFC_EXT.1 Subset Information Flow Control
	PP_MDF_V3.3	FDP_STG_EXT.1 User Data Storage
	PP_MDF_V3.3	FDP_UPC_EXT.1/APPS Inter-TSF User Data Transfer Protection (Applications)
	PP_MDF_V3.3	FDP_UPC_EXT.1/BLUETOOTH Inter-TSF User Data Transfer Protection (Bluetooth)
FIA: Identification and Authentication	PP_MDF_V3.3	FIA_AFL_EXT.1 Authentication Failure Handling
	MOD_BT_V1.0	FIA_BLT_EXT.1 Bluetooth User Authorization
	MOD_BT_V1.0	FIA_BLT_EXT.2 Bluetooth Mutual Authentication
	MOD_BT_V1.0	FIA_BLT_EXT.3 Rejection of Duplicate Bluetooth Connections
	MOD_BT_V1.0	FIA_BLT_EXT.4 Secure Simple Pairing
	MOD_BT_V1.0	FIA_BLT_EXT.6 Trusted Bluetooth User Authorization
	MOD_BT_V1.0	FIA_BLT_EXT.7 Untrusted Bluetooth User Authorization
	MOD_BIO_V1.1	FIA_MBE_EXT.1 Biometric enrolment
	MOD_BIO_V1.1	FIA_MBE_EXT.2/Fingerprint Quality of biometric templates for biometric enrolment
	MOD_BIO_V1.1	FIA_MBV_EXT.1/BMFPS Biometric verification
	MOD_BIO_V1.1	FIA_MBV_EXT.1/UDFPS Biometric verification
	MOD_BIO_V1.1	FIA_MBV_EXT.2/Fingerprint Quality of biometric samples for biometric verification

Requirement Class	PP	Requirement Component
	MOD_WLANC_V1.0	FIA_PAE_EXT.1 Port Access Entity Authentication
	PP_MDF_V3.3	FIA_PMG_EXT.1 Password Management
	PP_MDF_V3.3	FIA_TRT_EXT.1 Authentication Throttling
	PP_MDF_V3.3	FIA_UAU.5 Multiple Authentication Mechanisms
	PP_MDF_V3.3	FIA_UAU.6/CREDENTIAL Re-Authentication (Credential Change)
	PP_MDF_V3.3	FIA_UAU.6/LOCKED Re-Authentication (TSF Lock)
	PP_MDF_V3.3	FIA_UAU.7 Protected Authentication Feedback
	PP_MDF_V3.3	FIA_UAU_EXT.1 Authentication for Cryptographic Operation
	PP_MDF_V3.3	FIA_UAU_EXT.2 Timing of Authentication
	PP_MDF_V3.3	FIA_X509_EXT.1 Validation of Certificates
	MOD_WLANC_V1.0	FIA_X509_EXT.1/WLAN X.509 Certificate Validation
	PP_MDF_V3.3	FIA_X509_EXT.2 X509 Certificate Authentication
	MOD_WLANC_V1.0	FIA_X509_EXT.2/WLAN X.509 Certificate Authentication (EAP-TLS for WLAN)
	PP_MDF_V3.3	FIA_X509_EXT.3 Request Validation of Certificates
	MOD_WLANC_V1.0	FIA_X509_EXT.6 X.509 Certificate Storage and Management
FMT: Security Management	PP_MDF_V3.3	FMT_MOF_EXT.1 Management of Security Functions Behavior
	PP_MDF_V3.3	FMT_SMF.1 Specification of Management Functions
	MOD_BT_V1.0	FMT_SMF_EXT.1/BT Specification of Management Functions
	MOD_WLANC_V1.0	FMT_SMF.1/WLAN Specification of Management Functions (WLAN Client)
	PP_MDF_V3.3	FMT_SMF_EXT.2 Specification of Remediation Actions
	PP_MDF_V3.3	FMT_SMF_EXT.3 Current Administrator
FPT: Protection of the TSF	PP_MDF_V3.3	FPT_AEX_EXT.1 Application Address Space Layout Randomization
	PP_MDF_V3.3	FPT_AEX_EXT.2 Memory Page Permissions
	PP_MDF_V3.3	FPT_AEX_EXT.3 Stack Overflow Protection
	PP_MDF_V3.3	FPT_AEX_EXT.4 Domain Isolation
	PP_MDF_V3.3	FPT_AEX_EXT.5 Kernel Address Space Layout Randomization
	PP_MDF_V3.3	FPT_BBD_EXT.1 Application Processor Mediation
	MOD_BIO_V1.1	FPT_BDP_EXT.1 Biometric data processing
	PP_MDF_V3.3	FPT_JTA_EXT.1 JTAG Disablement
	PP_MDF_V3.3 & MOD_BIO_V1.1	FPT_KST_EXT.1 Key Storage
	PP_MDF_V3.3 & MOD_BIO_V1.1	FPT_KST_EXT.2 No Key Transmission
	PP_MDF_V3.3	FPT_KST_EXT.3 No Plaintext Key Export
	PP_MDF_V3.3	FPT_NOT_EXT.1 Self-Test Notification
	MOD_BIO_V1.1	FPT_PBT_EXT.1 Protection of biometric template
	PP_MDF_V3.3	FPT_STM.1 Reliable time stamps
	PP_MDF_V3.3	FPT_TST_EXT.1 TSF Cryptographic Functionality Testing

Requirement Class	PP	Requirement Component
	PP_MDF_V3.3	FPT_TST_EXT.2/PREKERNEL TSF Integrity Checking (Pre-Kernel)
	PP_MDF_V3.3	FPT_TST_EXT.2/POSTKERNEL TSF Integrity Checking (Post-Kernel)
	MOD_WLANC_V1.0	FPT_TST_EXT.3/WLAN TSF Cryptographic Functionality Testing (WLAN Client)
	PP_MDF_V3.3	FPT_TUD_EXT.1 Trusted Update: TSF Version Query
	PP_MDF_V3.3	FPT_TUD_EXT.2 TSF Update Verification
	PP_MDF_V3.3	FPT_TUD_EXT.3 Application Signing
	PP_MDF_V3.3	FPT_TUD_EXT.6 Trusted Update Verification
FTA: TOE Access	PP_MDF_V3.3	FTA_SSL_EXT.1 TSF- and User-initiated Locked State
	PP_MDF_V3.3	FTA_TAB.1 Default TOE Access Banners
	MOD_WLANC_V1.0	FTA_WSE_EXT.1 Wireless Network Access
FTP: Trusted Path/Channels	MOD_BT_V1.0	FTP_BLT_EXT.1 Bluetooth Encryption
	MOD_BT_V1.0	FTP_BLT_EXT.2 Persistence of Bluetooth Encryption
	MOD_BT_V1.0	FTP_BLT_EXT.3/BR Bluetooth Encryption Parameters (BR/EDR)
	MOD_BT_V1.0	FTP_BLT_EXT.3/LE Bluetooth Encryption Parameters (LE)
	MOD_WLANC_V1.0	FTP_ITC.1/WLAN Trusted Channel Communication (Wireless LAN)
	PP_MDF_V3.3	FTP_ITC_EXT.1 Trusted Channel Communication

Table 10 - TOE Security Functional Components

5.1.1 Security Audit (FAU)

5.1.1.1 PP_MDF_V3.3:FAU_GEN.1 Audit Data Generation

FAU_GEN.1.1

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

1. Start-up and shutdown of the audit functions
2. All auditable events for the [**not selected**] level of audit
3. All administrative actions
4. Start-up and shutdown of the OS
5. Insertion or removal of removable media
6. Specifically defined auditable events in Table 2 of the PP_MDF_V3.3
7. [***Specifically defined auditable events in Table 11 - PP_MDF_V3.3 Audit Events from Table 3 of the PP_MDF_V3.3 (marked with (ADD)***]

Requirement	Audit Event	Content
FAU_GEN.1	Start-up and shutdown of the audit functions	
	All administrative actions	
	Start-up and shutdown of the Rich OS	
FAU_GEN.1		
FAU_SAR.1		

Requirement	Audit Event	Content
FAU_STG.1		
FAU_STG.4		
FCS_CKM.1	[***None***].	
FCS_CKM.2/UNLOCKED		
FCS_CKM.2/LOCKED		
FCS_CKM_EXT.1	[***None***]	
FCS_CKM_EXT.2		
FCS_CKM_EXT.3		
FCS_CKM_EXT.4		
FCS_CKM_EXT.5	[***None***]	
FCS_CKM_EXT.6		
FCS_COP.1/ENCRYPT		
FCS_COP.1/HASH		
FCS_COP.1/SIGN		
FCS_COP.1/KEYHMAC		
FCS_COP.1/CONDITION		
FCS_IV_EXT.1		
FCS_SRV_EXT.1		
FCS_SRV_EXT.2 (ADD)		
FCS_STG_EXT.1	Import or destruction of key.	Identity of key. Role and identity of requestor.
	[***None***]	
FCS_STG_EXT.2		
FCS_STG_EXT.3	Failure to verify integrity of stored key.	Identity of key being verified.
FDP_ACF_EXT.1		
FDP_ACF_EXT.2 (ADD)		
FDP_DAR_EXT.1	[***None***]	
FDP_DAR_EXT.2	[***Failure to encrypt/decrypt data***]	
FDP_IFC_EXT.1		
FDP_STG_EXT.1	Addition or removal of certificate from Trust Anchor Database.	Subject name of certificate.
FIA_PMG_EXT.1		
FIA_TRT_EXT.1		
FIA_UAU.5		
FIA_UAU.7		
FIA_UAU_EXT.1		
FIA_X509_EXT.1	Failure to validate X.509v3 certificate.	Reason for failure of validation.
FIA_X509_EXT.2		
FMT_MOF_EXT.1		
FPT_AEX_EXT.1		
FPT_AEX_EXT.2		
FPT_AEX_EXT.3		
FPT_AEX_EXT.4 (ADD)		
FPT_AEX_EXT.5 (ADD)		
FPT_BBD_EXT.1 (ADD)		

Requirement	Audit Event	Content
FPT_JTA_EXT.1		
FPT_KST_EXT.1		
FPT_KST_EXT.2		
FPT_KST_EXT.3		
FPT_NOT_EXT.1	[***None***]	[***No additional information***]
FPT_STM.1		
FPT_TST_EXT.1	Initiation of self-test.	
	Failure of self-test.	[***No additional information***]
FPT_TST_EXT.2/PREKERNEL	Start-up of TOE.	
	[***None***]	[***No additional information***]
FPT_TUD_EXT.1		
FTA_SSL_EXT.1		
FTA_TAB.1		

Table 11 - PP_MDF_V3.3 Audit Events

FAU_GEN.1.2

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

1. Date and time of the event
2. Type of event
3. Subject identity
4. The outcome (success or failure) of the event
5. Additional information in Table 11 - PP_MDF_V3.3 Audit Events from Table 2 (of the PP_MDF_V3.3)
6. [***Additional information in Table 3 (of the PP_MDF_V3.3)***]

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

FAU_GEN.1.1/BT

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

a. Start-up and shutdown of the audit functions
b. All auditable events for the [*not specified*] level of audit
c. [*Specifically defined auditable events in the Auditable Events table (of the MOD_BT_V1.0 shown in Table 12 - MOD_BT_V1.0 Audit Events)*]

Requirement	Audit Event	Content
FCS_CKM_EXT.8		
FIA_BLT_EXT.1	Failed user authorization of Bluetooth device.	User authorization decision (e.g., user rejected connection, incorrect pin entry).
	Failed user authorization for local Bluetooth Service.	Bluetooth address and name of device. Bluetooth profile. Identity of local service with [***service ID***].
FIA_BLT_EXT.2	Initiation of Bluetooth connection.	Bluetooth address and name of device.
	Failure of Bluetooth connection.	Reason for failure.
FIA_BLT_EXT.3 (O)	Duplicate connection attempt.	BD_ADDR of connection attempt.

Requirement	Audit Event	Content
FIA_BLT_EXT.4		
FIA_BLT_EXT.6		
FIA_BLT_EXT.7		
FTP_BLT_EXT.1		
FTP_BLT_EXT.2		
FTP_BLT_EXT.3/BR		
FTP_BLT_EXT.3/LE		

Table 12 - MOD_BT_V1.0 Audit Events

FAU_GEN.1.2/BT

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

a. Date and time of the event
b. Type of event
c. Subject identity
d. The outcome (success or failure) of the event
e. For reach audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [*Additional information in the Auditable Events table (of the MOD_BT_V1.0)*]

(TD0707 applied)

5.1.1.3 MOD_WLANC_V1.0:FAU_GEN.1/WLAN Audit Data Generation (Wireless LAN)

FAU_GEN.1.1/WLAN

The TSF shall [***invoke platform-provided functionality***] to generate an audit record of the following auditable events:

a. Startup and shutdown of the audit functions;
b. All auditable events for the [*not specified*] level of audit; and
c. [*all auditable events for mandatory SFRs specified in Table 2 and selected SFRs in Table 5 (of the MOD_WLANC_V1.0 shown in Table 13 - MOD_WLANC_V1.0 Audit Events)*]

Requirement	Audit Event	Content
FAU_GEN.1/WLAN		
FCS_CKM.1/WPA		
FCS_CKM.2/WLAN		
FCS_TLSC_EXT.1/WLAN	Failure to establish an EAP-TLS session.	Reason for failure. Non-TOE endpoint of connection.
	Establishment/termination of an EAP-TLS session.	Non-TOE endpoint of connection.
FCS_TLSC_EXT.2/WLAN		
FCS_WPA_EXT.1		
FIA_PAE_EXT.1		
FIA_X509_EXT.1/WLAN	Failure to validate X.509v3 certificate.	Reason for failure of validation.
FIA_X509_EXT.2/WLAN		
FIA_X509_EXT.6	Attempts to load certificates.	

Requirement	Audit Event	Content
	Attempts to revoke certificates.	
FMT_SMF.1/WLAN		
FPT_TST_EXT.3/WLAN	Execution of this set of TSF self-tests.	(Done as part of FPT_TST_EXT.1)
	[***None***].	[***None***].
FTA_WSE_EXT.1	All attempts to connect to access points.	For each access point record the [***Certificate Check Message and the last [2] octets***] of the MAC Address Success and failures (including reason for failure).
FTP_ITC_EXT.1/WLAN	All attempts to establish a trusted channel.	Identification of the non-TOE endpoint of the channel.

Table 13 - MOD_WLANC_V1.0 Audit Events

FAU_GEN.1.2/WLAN

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

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

b. For each audit event type, based on the auditable event definitions of the functional components included in the PP-Module/ST, [*Additional Audit Record Contents as specified in Table 2 and Table 5 (of the MOD_WLANC_V1.0 shown in Table 13 - MOD_WLANC_V1.0 Audit Events)*]

5.1.1.4 PP_MDF_V3.3:FAU_SAR.1 Audit Review

FAU_SAR.1.1

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

FAU_SAR.1.2

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

5.1.1.5 PP_MDF_V3.3:FAU_STG.1 Audit Storage Protection

FAU_STG.1.1

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

FAU_STG.1.2

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

5.1.1.6 PP_MDF_V3.3:FAU_STG.4 Prevention of Audit Data Loss

FAU_STG.4.1

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

5.1.2 Cryptographic Support (FCS)

5.1.2.1 PP_MDF_V3.3:FCS_CKM.1 Cryptographic Key Generation

FCS_CKM.1.1

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

- ***RSA schemes using cryptographic key sizes of [2048-bit or greater] that meet [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3],***
- ***ECC schemes using ["NIST curves" P-384 and [P-256, P-521] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4]]***

].

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

FCS_CKM.1.1/WPA

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

5.1.2.3 PP_MDF_V3.3:FCS_CKM.2/UNLOCKED Cryptographic Key Establishment

FCS_CKM.2.1/UNLOCKED

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

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

].

5.1.2.4 PP_MDF_V3.3:FCS_CKM.2/LOCKED Cryptographic Key Establishment

FCS_CKM.2.1/LOCKED

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

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

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

5.1.2.5 MOD_WLANC_V1.0:FCS_CKM.2/WLAN Cryptographic Key Distribution (Group Temporal Key for WLAN)

FCS_CKM.2.1/WLAN

The TSF shall decrypt Group Temporal Key in accordance with a specified cryptographic key distribution method [*AES Key Wrap (as defined in RFC 3394) in an EAPOL-Key frame (as defined in IEEE 802.11-2012 for the packet format and timing considerations*] and does not expose the cryptographic keys.

5.1.2.6 PP_MDF_V3.3:FCS_CKM_EXT.1 Cryptographic Key Support

FCS_CKM_EXT.1.1

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

FCS_CKM_EXT.1.2

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

FCS_CKM_EXT.1.3

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

5.1.2.7 PP_MDF_V3.3:FCS_CKM_EXT.2 Cryptographic Key Random Generation

FCS_CKM_EXT.2.1

All DEKs shall be [

- ***randomly generated***

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

5.1.2.8 PP_MDF_V3.3:FCS_CKM_EXT.3 Cryptographic Key Generation

FCS_CKM_EXT.3.1

The TSF shall use [

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

].

FCS_CKM_EXT.3.2

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

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

[

- ***Generate the KEK using an RBG that meets this profile (as specified in FCS_RBG_EXT.1)***
- ***Generate the KEK using a key generation scheme that meets this profile (as specified in FCS_CKM.1)***

- ***Combine the KEK from other KEKs in a way that preserves the effective entropy of each factor by [concatenating the keys and using a KDF (as described in SP 800-108), encrypting one key with another]***

].

5.1.2.9 PP_MDF_V3.3:FCS_CKM_EXT.4 Key Destruction

FCS_CKM_EXT.4.1

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

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

FCS_CKM_EXT.4.2

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

5.1.2.10 PP_MDF_V3.3:FCS_CKM_EXT.5 TSF Wipe

FCS_CKM_EXT.5.1

The TSF shall wipe all protected data by [

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

- ***For non-volatile memory other than EEPROM and flash, the destruction shall be executed by a single direct overwrite with a random pattern that is changed before each write.***].

FCS_CKM_EXT.5.2

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

5.1.2.11 PP_MDF_V3.3:FCS_CKM_EXT.6 Salt Generation

FCS_CKM_EXT.6.1

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

5.1.2.12 MOD_BT_V1.0:FCS_CKM_EXT.8 Bluetooth Key Generation

FCS_CKM_EXT.8.1

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

5.1.2.13 PP_MDF_V3.3:FCS_COP.1/ENCRYPT Cryptographic operation

FCS_COP.1.1/ENCRYPT

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

- AES-CBC (as defined in FIPS PUB 197, and NIST SP 800-38A) mode
- AES-CCMP (as defined in FIPS PUB 197, NIST SP 800-38C and IEEE 802.11-2012), and
- [
  - ***AES Key Wrap (KW) (as defined in NIST SP 800-38F)***
  - ***AES-GCM (as defined in NIST SP 800-38D)***
  - ***AES-XTS (as defined in NIST SP 800-38E) mode***
  - ***AES-GCMP-256 (as defined in NIST SP800-38D and IEEE 802.11ac-2013)***

  ]

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

5.1.2.14 PP_MDF_V3.3:FCS_COP.1/HASH Cryptographic operation

FCS_COP.1.1/HASH

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

5.1.2.15 PP_MDF_V3.3:FCS_COP.1/SIGN Cryptographic operation

FCS_COP.1.1/SIGN

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

- ***[RSA schemes] using cryptographic key sizes of [2048-bit or greater] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 4]***

- ***[ECDSA schemes] using ["NIST curves" P-384 and [P-256, P-521]] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5]***

].

5.1.2.16 PP_MDF_V3.3:FCS_COP.1/KEYHMAC Cryptographic operation

FCS_COP.1.1/KEYHMAC

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

5.1.2.17 PP_MDF_V3.3:FCS_COP.1/CONDITION Cryptographic operation

FCS_COP.1.1/CONDITION

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

5.1.2.18 PP_MDF_V3.3:FCS_HTTPS_EXT.1 HTTPS Protocol

FCS_HTTPS_EXT.1.1

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

FCS_HTTPS_EXT.1.2

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

FCS_HTTPS_EXT.1.3

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

5.1.2.19 PP_MDF_V3.3:FCS_IV_EXT.1 Initialization Vector Generation

FCS_IV_EXT.1.1

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

5.1.2.20 PP_MDF_V3.3:FCS_RBG_EXT.1 Random Bit Generation

FCS_RBG_EXT.1.1

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

FCS_RBG_EXT.1.2

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

FCS_RBG_EXT.1.3

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

5.1.2.21 PP_MDF_V3.3:FCS_SRV_EXT.1 Cryptographic Algorithm Services

FCS_SRV_EXT.1.1

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

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

].

5.1.2.22 PP_MDF_V3.3:FCS_SRV_EXT.2 Cryptographic Algorithm Services

FCS_SRV_EXT.2.1

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

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

] by keys stored in the secure key storage.

5.1.2.23 PP_MDF_V3.3:FCS_STG_EXT.1 Cryptographic Key Storage

FCS_STG_EXT.1.1

The TSF shall provide [***mutable hardware[2], software-based***] secure key storage for asymmetric private keys and [***symmetric keys, persistent secrets***].

FCS_STG_EXT.1.2

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

FCS_STG_EXT.1.3

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

FCS_STG_EXT.1.4

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

FCS_STG_EXT.1.5

[2] Does not apply to the Pixel 6 Pro/6/6a

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

5.1.2.24 PP_MDF_V3.3:FCS_STG_EXT.2 Encrypted Cryptographic Key Storage

FCS_STG_EXT.2.1

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

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

].

FCS_STG_EXT.2.2

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

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

].

5.1.2.25 PP_MDF_V3.3:FCS_STG_EXT.3 Integrity of Encrypted Key Storage

FCS_STG_EXT.3.1

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

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

].

FCS_STG_EXT.3.2

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

5.1.2.26 PKG_TLS_V1.1:FCS_TLS_EXT.1 TLS Protocol

FCS_TLS_EXT.1

The product shall implement [

- ***TLS as a client***

].

5.1.2.27 PKG_TLS_V1.1:FCS_TLSC_EXT.1 TLS Client Protocol

FCS_TLSC_EXT.1.1

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

- ***TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,***
- ***TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***

] and also supports functionality for [

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

].

FCS_TLSC_EXT.1.2

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

FCS_TLSC_EXT.1.3

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

- ***with no exceptions***

].

(TD0442 applied)

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

FCS_TLSC_EXT.2.1

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

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

FCS_TLSC_EXT.1.1/WLAN

The product shall implement TLS 1.2 (RFC 5246) and [***TLS 1.1 (RFC 4346)***] in support of the EAP-TLS protocol as specified in RFC 5216 supporting the following cipher suites: [

- ***TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246,***
- ***TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,***
- ***TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,***
- ***TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***

].

FCS_TLSC_EXT.1.2/WLAN

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

FCS_TLSC_EXT.1.3/WLAN

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

FCS_TLSC_EXT.1.4/WLAN

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

FCS_TLSC_EXT.1.5/WLAN

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

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

FCS_TLSC_EXT.2.1/WLAN

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

5.1.2.31 PKG_TLS_V1.1:FCS_TLSC_EXT.4 TLS Client Support for Renegotiation

FCS_TLSC_EXT.4.1

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

5.1.2.32 PKG_TLS_V1.1:FCS_TLSC_EXT.5 TLS Client Support for Supported Groups Extension

FCS_TLSC_EXT.5.1

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

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

].

5.1.2.33 MOD_WLANC_V1.0:FCS_WPA_EXT.1 Supported WPA Versions

FCS_WPA_EXT.1.1

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

5.1.3 User Data Protection (FDP)

5.1.3.1 PP_MDF_V3.3:FDP_ACF_EXT.1 Security Access Control for System Services

FDP_ACF_EXT.1.1

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

FDP_ACF_EXT.1.2

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

common application developer (for sharing between applications), no one (for sharing between personal and enterprise profiles)].

5.1.3.2 PP_MDF_V3.3:FDP_ACF_EXT.2 Security Access Control for System Resources

FDP_ACF_EXT.2.1

The TSF shall provide a separate [***address book, calendar, [keychain]***] for each application group and only allow applications within that process group to access the resource. Exceptions may only be explicitly authorized for such sharing by [***the administrator (for address book), no one (for calendar, keychain)***].

5.1.3.3 PP_MDF_V3.3:FDP_DAR_EXT.1 Protected Data Encryption

FDP_DAR_EXT.1.1

Encryption shall cover all protected data.

FDP_DAR_EXT.1.2

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

5.1.3.4 PP_MDF_V3.3:FDP_DAR_EXT.2 Sensitive Data Encryption

FDP_DAR_EXT.2.1

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

FDP_DAR_EXT.2.2

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

FDP_DAR_EXT.2.3

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

FDP_DAR_EXT.2.4

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

5.1.3.5 PP_MDF_V3.3:FDP_IFC_EXT.1 Subset Information Flow Control

FDP_IFC_EXT.1.1

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

5.1.3.6 PP_MDF_V3.3:FDP_STG_EXT.1 User Data Storage

FDP_STG_EXT.1.1

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

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

FDP_UPC_EXT.1.1/APPS

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

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

and [

- ***no other protocol***

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

FDP_UPC_EXT.1.2/APPS

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

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

FDP_UPC_EXT.1.1/BLUETOOTH

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

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

and [

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

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

FDP_UPC_EXT.1.2/BLUETOOTH

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

5.1.4 Identification and Authentication (FIA)

5.1.4.1 PP_MDF_V3.3:FIA_AFL_EXT.1 Authentication Failure Handling

FIA_AFL_EXT.1.1

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

FIA_AFL_EXT.1.2

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

FIA_AFL_EXT.1.3

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

FIA_AFL_EXT.1.4

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

FIA_AFL_EXT.1.5

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

FIA_AFL_EXT.1.6

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

5.1.4.2 MOD_BT_V1.0:FIA_BLT_EXT.1 Bluetooth User Authorization

FIA_BLT_EXT.1.1

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

5.1.4.3 MOD_BT_V1.0:FIA_BLT_EXT.2 Bluetooth Mutual Authentication

FIA_BLT_EXT.2.1

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

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

FIA_BLT_EXT.3.1

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

5.1.4.5 MOD_BT_V1.0:FIA_BLT_EXT.4 Secure Simple Pairing

FIA_BLT_EXT.4.1

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

FIA_BLT_EXT.4.2

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

5.1.4.6 MOD_BT_V1.0:FIA_BLT_EXT.6 Trusted Bluetooth User Authorization

FIA_BLT_EXT.6.1

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

5.1.4.7 MOD_BT_V1.0:FIA_BLT_EXT.7 Untrusted Bluetooth User Authorization

FIA_BLT_EXT.7.1

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

5.1.4.8 MOD_BIO_V1.1:FIA_MBE_EXT.1 Biometric enrolment

FIA_MBE_EXT.1.1

The TSF shall provide a mechanism to enrol an authenticated user to the biometric system.

(TD0714 applied)

5.1.4.9 MOD_BIO_V1.1:FIA_MBE_EXT.2/Fingerprint Quality of biometric templates for biometric enrolment

FIA_MBE_EXT.2.1/Fingerprint

The TSF shall only use biometric samples of sufficient quality for enrolment. Sufficiency of sample data shall be determined by measuring sample with [***developer defined quality assessment method***].

5.1.4.10 MOD_BIO_V1.1:FIA_MBV_EXT.1/BMFPS Biometric verification

This applies to the Pixel 5/5a-5G, 4a-5G/4a devices.

FIA_MBV_EXT.1.1/BMFPS

The TSF shall provide a biometric verification mechanism using [***fingerprint***].

FIA_MBV_EXT.1.2/BMFPS

The TSF shall provide a biometric verification mechanism with the [***FAR***] not exceeding [**1:100,000**] for the upper bound of [**95%**] confidence interval and, [***FRR***] not exceeding [**2%**] for the upper bound of [**95%**] confidence interval.

5.1.4.11 MOD_BIO_V1.1:FIA_MBV_EXT.1/UDFPS Biometric verification

This applies to the Pixel 7 Pro/7, 6 Pro/6/6a devices.

FIA_MBV_EXT.1.1/UDFPS

The TSF shall provide a biometric verification mechanism using [***fingerprint***].

FIA_MBV_EXT.1.2/UDFPS

The TSF shall provide a biometric verification mechanism with the [***FAR***] not exceeding [**1:50,000**] for the upper bound of [**95%**] confidence interval and, [***FRR***] not exceeding [**3%**] for the upper bound of [**95%**] confidence interval.

5.1.4.12 MOD_BIO_V1.1:FIA_MBV_EXT.2/Fingerprint Quality of biometric samples for biometric verification

FIA_MBV_EXT.2.1/Fingerprint

The TSF shall only use biometric samples of sufficient quality for verification. Sufficiency of sample data shall be determined by measuring sample with [***developer defined quality assessment method***].

5.1.4.13 MOD_WLANC_V1.0:FIA_PAE_EXT.1 Port Access Entity Authentication

FIA_PAE_EXT.1.1

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

5.1.4.14 PP_MDF_V3.3:FIA_PMG_EXT.1 Password Management

FIA_PMG_EXT.1.1

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

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

5.1.4.15 PP_MDF_V3.3:FIA_TRT_EXT.1 Authentication Throttling

FIA_TRT_EXT.1.1

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

5.1.4.16 PP_MDF_V3.3:FIA_UAU.5 Multiple Authentication Mechanisms

FIA_UAU.5.1

The TSF shall provide password and [***biometric in accordance with the Biometric Enrollment and Verification, version 1.1***] to support user authentication.

FIA_UAU.5.2

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

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

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

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

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

To change protected settings or issue certain commands:

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

].

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

FIA_UAU.6.1/CREDENTIAL

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

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

FIA_UAU.6./LOCKED

The TSF shall re-authenticate the user via an authentication factor defined in FIA_UAU.5.1 under the conditions TSF-initiated lock, user-initiated lock, [**no other conditions**].

5.1.4.19 PP_MDF_V3.3:FIA_UAU.7 Protected Authentication Feedback

FIA_UAU.7.1

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

5.1.4.20 PP_MDF_V3.3:FIA_UAU_EXT.1 Authentication for Cryptographic Operation

FIA_UAU_EXT.1.1

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

5.1.4.21 PP_MDF_V3.3:FIA_UAU_EXT.2 Timing of Authentication

FIA_UAU_EXT.2.1

The TSF shall allow **[***[***

- ***Take screen shots (stored internally)***
- ***Make emergency calls***
- ***Receive calls***
- ***Take pictures (stored internally) - unless the camera was disabled***
- ***Turn the TOE off***
- ***Restart the TOE***
- ***Place TOE into lockdown mode***
- ***Enable Airplane mode***
- ***Change the state of Wi-Fi, Bluetooth, Mobile Data (cellular data)***
- ***Change Battery Saver mode***
- ***Adjust screen brightness***
- ***See notifications (note that some notifications identify actions, for example to view a screenshot; however, selecting those notifications highlights the password prompt and require the password to access that data)***
- ***Configure sound, vibrate, or mute***
- ***Set the volume (up and down) for ringtone***

- ***Change keyboard input method***
- ***Change live captions***
- ***Access notification widgets (without authentication):***
  - ***Flashlight toggle***
  - ***Do not disturb toggle***
  - ***Auto rotate toggle***
  - ***Sound (on, mute, vibrate)***
  - ***Night light filter toggle***

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

FIA_UAU_EXT.2.2

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

5.1.4.22 PP_MDF_V3.3:FIA_X509_EXT.1 Validation of Certificates

FIA_X509_EXT.1.1

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

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a certificate in the Trust Anchor Database.
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.
- The TSF shall validate that any CA certificate includes caSigning purpose in the key usage field.
- The TSF shall validate the revocation status of the certificate using [***OCSP as specified in RFC 6960***].
- The TSF shall validate the extendedKeyUsage field according to the following rules:
  - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field
  - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
  - Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) in the extendedKeyUsage field. [conditional]
  - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the extendedKeyUsage field.
  - OCSP certificates presented for OCSP responses shall have the OCSP Signing purpose (id-dp 9 with OID 1.3.6.1.5.5.7.3.9) in the extendedKeyUsage field. [conditional]

FIA_X509_EXT.1.2

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

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

FIA_X509_EXT.1.1/WLAN

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

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

FIA_X509_EXT.1.2/WLAN

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

5.1.4.24 PP_MDF_V3.3:FIA_X509_EXT.2 X509 Certificate Authentication

FIA_X509_EXT.2.1

The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [mutually authenticated TLS as defined in the Package for Transport Layer Security, HTTPS, [***no other protocol***]] and [***no additional uses***].

FIA_X509_EXT.2.2

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

5.1.4.25 MOD_WLANC_V1.0:FIA_X509_EXT.2/WLAN X.509 Certificate Authentication (EAP-TLS for WLAN)

FIA_X509_EXT.2.1/WLAN

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

5.1.4.26 PP_MDF_V3.3:FIA_X509_EXT.3 Request Validation of Certificates

FIA_X509_EXT.3.1

The TSF shall provide a certificate validation service to applications.

FIA_X509_EXT.3.2

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

5.1.4.27 MOD_WLANC_V1.0:FIA_X509_EXT.6 Certificate Storage and Management

FIA_X509_EXT.6.1

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

FIA_X509_EXT.6.2

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

5.1.5 Security management (FMT)

5.1.5.1 PP_MDF_V3.3:FMT_MOF_EXT.1 Management of Security Functions Behavior

FMT_MOF_EXT.1.1

The TSF shall restrict the ability to perform the functions in [column 4 of Table 14 - Security Management Functions] to the user.

FMT_MOF_EXT.1.2

The TSF shall restrict the ability to perform the functions [in column 6 of Table 14 - Security Management Functions] to the administrator when the device is enrolled and according to the administrator-configured policy.

5.1.5.2 PP_MDF_V3.3:FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1

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

(In the last four columns, M = Mandatory and I = Implemented, to denote which options are available for any management function.)

#	Management Function	Implemented	User Only	Admin	Admin Only
1.	configure password policy: • Minimum password length • Minimum password complexity • Maximum password lifetime The administrator can configure the required password characteristics (minimum length, complexity, and lifetime) using the Android MDM APIs. Length: an integer value of characters Complexity: Unspecified, Something, Numeric, Alphabetic, Alphanumeric, Complex. Lifetime: an integer value of seconds (0 = no maximum).	M		M	M

#	Management Function	Implemented	User Only	Admin	Admin Only
2.	configure session locking policy: • Screen-lock enabled/disabled • Screen lock timeout • Number of authentication failures The administrator can configure the session locking policy using the Android MDM APIs. Screen lock timeout: an integer number of milliseconds before the TOE locks. An integer number (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 [negative integers and zero means no lockout]). Authentication failures: an integer number (-2,147,483,648 to 2,147,483,648 [negative integers and zero means no limit]) of failures before a wipe action is initiated. This only applies to password authentication, biometric attempts do not increment this counter.	M		M	M
3.	enable/disable the VPN protection: • Across device • [***on a per-group of applications processes basis***] Both users (using the TOE's settings UI) and administrator (using the TOE's MDM APIs) can configure a third-party VPN client and then enable the VPN client to protect traffic. The User can set up VPN protection, but if an admin enables VPN protection, the user cannot disable it. The administrator (using the TOE's MDM APIs) can configure a VPN client for a group of applications through the creation of a work profile. The VPN client for the work profile will be associated with all the applications included in the work profile.	M		I	I
4.	enable/disable [**Bluetooth**] enable/disable [**NFC, Wi-Fi, cellular**] The administrator (using the TOE's MDM APIs) can manage Bluetooth radio. The user cannot override the administrator setting. The NFC, Wi-Fi and cellular radios can be disabled by the administrator (using the TOE's MDM APIs), but the user (using the TOE's settings UI) is able to override this and turn the radios back on. The TOE's radios operate at frequencies of 2.4 GHz (NFC/Bluetooth), 2.4/5 GHz (Wi-Fi), and 850, 900, 1800, 1900 MHz (4G/LTE). The radios are initialized during the initial power-up sequence. If the radio is supposed to be off (by setting), it will be turned off after the initial check.	M M	I	I	I

#	Management Function	Implemented	User Only	Admin	Admin Only
5.	enable/disable [**microphone, camera**]: • Across device, • [***on a per-app basis***] An administrator can enable/disable the device's microphone via an MDM API. Once the microphone has been disabled, the user cannot re-enable it until the administrator enables it. In the user's settings, a user can view a permission by type (i.e. camera, microphone). The user can access this by going to the settings UI (`Settings -> Privacy -> Permission manager -> <camera/microphone>`) and revoking any applications.	M M		I	I
6.	transition to the locked state Both users (using the TOE's settings UI) and administrators (using the TOE's MDM APIs) can transition the TOE into a locked state.	M		M	
7.	TSF wipe of protected data Both users (using the TOE's settings UI) and administrators (using the TOE's MDM APIs) can force the TOE to perform a full wipe (factory reset) of data.	M		M	
8.	configure application installation policy by: [• ***restricting the sources of applications,*** • ***denying installation of applications***] The administrator (using the TOE's MDM APIs) can configure the TOE so that applications cannot be installed and can also block the use of the Google Market Place.	M		M	M
9.	import keys or secrets into the secure key storage Both users (using the TOE's settings UI) and administrators (using the TOE's MDM APIs) can import secret keys into the secure key storage.	M		I	
10.	destroy imported keys or secrets and [***no other keys or secrets***] in the secure key storage Both users and administrators (using the TOE's MDM APIs) can destroy secret keys in the secure key storage.	M		I	
11.	import X.509v3 certificates into the Trust Anchor Database Both users (using the TOE's settings UI) and administrators (using the TOE's MDM APIs) can import X.509v3 certificates into the Trust Anchor Database.	M		M	

#	Management Function	Implemented	User Only	Admin	Admin Only
12.	remove imported X.509v3 certificates and [***no other X.509v3 certificates***] in the Trust Anchor Database Both users (using the TOE's settings UI) and administrators (using the TOE's MDM APIs) can remove imported X.509v3 certificates from the Trust Anchor Database as well as disable any of the TOE's default Root CA certificates (in the latter case, the CA certificate still resides in the TOE's read-only system partition; however, the TOE will treat that Root CA certificate and any certificate chaining to it as untrusted).	M		I	
13.	enroll the TOE in management TOE users can enroll the TOE in management according to the instructions specific to a given MDM. Presumably any enrollment would involve at least some user functions (e.g., install an MDM agent application) on the TOE prior to enrollment.	M			
14.	remove applications Both users (using the TOE's settings UI) and administrators (using the TOE's MDM APIs) can uninstall user and administrator installed applications on the TOE.	M		M	
15.	update system software Users can check for updates and cause the device to update if an update is available. An administrator can use MDM APIs to query the version of the TOE and query the installed applications and an MDM agent on the TOE could issue pop-ups, initiate updates, block communication, etc. until any necessary updates are completed.	M		M	
16.	install applications Both users and administrators (using the TOE's MDM APIs) can install applications on the TOE.	M		M	
17.	remove Enterprise applications An administrator (using the TOE's MDM APIs) can uninstall Enterprise installed applications on the TOE.	M		M	
18.	enable/disable display notification in the locked state of: [• ***all notifications***] Notifications can be configured to display in the following formats: • Users & administrators: show all notification content • Users: hide sensitive content • Users & administrators: hide notifications entirely If the administrator sets any of the above settings, the user cannot change it.	M		I	I
19.	enable data-at rest protection The TOE always encrypts its user data storage.	M			
20.	enable removable media's data-at-rest protection The device does not support removable media.				

#	Management Function	Implemented	User Only	Admin	Admin Only
21.	enable/disable location services: • Across device • [***no other method***] The administrator (using the TOE's MDM APIs) can enable or disable location services. An additional MDM API can prohibit TOE users' ability to enable and disable location services.	M		I	I
22.	enable/disable the use of [***Biometric Authentication Factor***]	I		I	I
23.	configure whether to allow or disallow establishment of [**assignment**: *configurable trusted channel in FTP_ITC_EXT.1.1 or FDP_UPC_EXT.1.1/APPS*] if the peer or server certificate is deemed invalid.				
24.	enable/disable all data signaling over [**assignment:** *list of externally accessible hardware ports*]				
25.	enable/disable [**Bluetooth tethering**] The administrator (using the TOE's MDM APIs) can enable/disable all tethering methods (i.e. all or none disabled). The TOE acts as a server (acting as an access point, a USB Ethernet adapter, and as a Bluetooth Ethernet adapter respectively) in order to share its network connection with another device.	I		I	I
26.	enable/disable developer modes The administrator (using the TOE's MDM APIs) can disable Developer Mode. Unless disabled by the administrator, TOE users can enable and disable Developer Mode.	I		I	I
27.	enable/disable bypass of local user authentication N/A – It is not possible to bypass local user auth for this TOE				
28.	wipe Enterprise data An administrator (using the TOE's MDM APIs) can remove Enterprise applications and their data.	I		I	
29.	approve [**selection**: *import, removal*] by applications of X.509v3 certificates in the Trust Anchor Database				
30.	configure whether to allow or disallow establishment of a trusted channel if the TSF cannot establish a connection to determine the validity of a certificate				
31.	enable/disable the cellular protocols used to connect to cellular network base stations				
32.	read audit logs kept by the TSF	I		I	
33.	configure [**selection**: *certificate, public-key*] used to validate digital signature on applications				
34.	approve exceptions for shared use of keys or secrets by multiple applications				

#	Management Function	Implemented	User Only	Admin	Admin Only
35.	approve exceptions for destruction of keys or secrets by applications that did not import the key or secret				
36.	configure the unlock banner The administrator (using the TOE's MDM APIs) can specify text to always be shown on the lock screen.	I		I	
37.	configure the auditable items				
38.	retrieve TSF-software integrity verification values				
39.	enable/disable [• ***USB mass storage mode***] The administrator (using the TOE's MDM APIs) can specify whether the device can have its storage mounted as USB storage available for read/write (when the device is unlocked) to another device (such as a computer).	I		I	
40.	enable/disable backup to [***all applications***] to [***remote system***] The administrator (using the TOE's MDM APIs) can specify whether applications can back up their data to a remote host based on the device user account. This is a global setting using the Google accounts on the device and does not apply to individual applications that may implement internal backup capabilities.	I		I	I
41.	enable/disable [• ***Hotspot functionality authenticated by [pre-shared key],*** • ***USB tethering authenticated by [no authentication]]*** The administrator (using the TOE's MDM APIs) can disable the Wi-Fi hotspot and USB tethering. Unless disabled by the administrator, TOE users can configure the Wi-Fi hotspot with a pre-shared key and can configure USB tethering (with no authentication, though the device must be unlocked to establish the initial tethering connection).	I		I	I
42.	approve exceptions for sharing data between [***groups of application***] The administrator (using the TOE's MDM APIs) can specify grouping of applications to restrict sharing data between the groups.	I		I	I
43.	place applications into application process groups based on [**assignment**: *enterprise configuration settings*]				
44.	unenroll the TOE from management The administrator (using the TOE's MDM APIs) or the user (using the TOE's settings UI) can choose to remove the TOE from management.	I		I	I

#	Management Function	Implemented	User Only	Admin	Admin Only
45.	enable/disable the Always On VPN protection • Across device • [***no other method***] The administrator (using the TOE's MDM APIs) can specify whether a VPN connection is required for the device to access any network services. The configuration would specify the VPN connection(s) required.	I		I	I
46.	revoke Biometric template				
47.	[**assignment**: *list of other management functions to be provided by the TSF*]				

Table 14 - Security Management Functions

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

FMT_SMF_EXT.1.1/BT

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

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

Table 15 - Bluetooth Security Management Functions

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

FMT_SMF_EXT.1.1/WLAN

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

#	Management Function	Implemented	Admin	User
WL-1.	configure security policy for each wireless network: • [***specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s)***] • security type • authentication protocol • client credentials to be used for authentication	M	M	
WL-2.	specify wireless networks (SSIDs) to which the TSF may connect; An administrator can specify a list of wireless networks to which the TOE may connect and can restrict the TOE to only allow a connection to the specified networks.	M	M	
WL-3.	enable/disable disable wireless network bridging capability (for example, bridging a connection between the WLAN and cellular radios to function as a hotspot) authenticated by [***pre-shared key***]	M	M	
WL-4.	enable/disable certificate revocation list checking;			
WL-5.	disable ad hoc wireless client-to-client connection capability			
WL-6.	disable roaming capability;			
WL-7.	enable/disable IEEE 802.1X pre-authentication;			
WL-8.	loading X.509 certificates into the TOE			
WL-9.	revoke X.509 certificates loaded into the TOE			
WL-10.	enable/disable and configure PMK caching: • set the amount of time (in minutes) PMK entries are cached; • set the maximum number of PMK entries that can be cached.			
WL-11.	configure security policy for each wireless network: set wireless frequency band to [**selection**: *2.4 GHz, 5 GHz, 6 GHz*]			

Table 16 - WLAN Security Management Functions

(TD0667 applied)

5.1.5.5 PP_MDF_V3.3:FMT_SMF_EXT.2 Specification of Remediation Actions

FMT_SMF_EXT.2.1

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

5.1.5.6 PP_MDF_V3.3:FMT_SMF_EXT.3 Current Administrator

FMT_SMF_EXT.3.1

The TSF shall provide a mechanism that allows users to view a list of currently authorized administrators and the management functions that each administrator is authorized to perform.

5.1.6 Protection of the TSF (FPT)

5.1.6.1 PP_MDF_V3.3:FPT_AEX_EXT.1 Application Address Space Layout Randomization

FPT_AEX_EXT.1.1

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

FPT_AEX_EXT.1.2

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

5.1.6.2 PP_MDF_V3.3:FPT_AEX_EXT.2 Memory Page Permissions

FPT_AEX_EXT.2.1

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

5.1.6.3 PP_MDF_V3.3:FPT_AEX_EXT.3 Stack Overflow Protection

FPT_AEX_EXT.3.1

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

5.1.6.4 PP_MDF_V3.3:FPT_AEX_EXT.4 Domain Isolation

FPT_AEX_EXT.4.1

The TSF shall protect itself from modification by untrusted subjects.

FPT_AEX_EXT.4.2

The TSF shall enforce isolation of address space between applications.

5.1.6.5 PP_MDF_V3.3:FPT_AEX_EXT.5 Kernel Address Space Layout Randomization

FPT_AEX_EXT.5.1

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

FPT_AEX_EXT.5.2

The base address of any kernel-space memory mapping will consist of [**13-25**] unpredictable bits.

5.1.6.6 PP_MDF_V3.3:FPT_BBD_EXT.1 Application Processor Mediation

FPT_BBD_EXT.1.1

The TSF shall prevent code executing on any baseband processor (BP) from accessing application processor (AP) resources except when mediated by the AP.

5.1.6.7 MOD_BIO_V1.1:FPT_BDP_EXT.1 Biometric data processing

FPT_BDP_EXT.1.1

Processing of plaintext biometric data shall be inside the SEE in runtime.

FPT_BDP_EXT.1.2

Transmission of plaintext biometric data between the capture sensor and the SEE shall be isolated from the main computer operating system on the TSF in runtime.

5.1.6.8 PP_MDF_V3.3:FPT_JTA_EXT.1 JTAG Disablement

FPT_JTA_EXT.1.1

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

5.1.6.9 PP_MDF_V3.3 & MOD_BIO_V1.1:FPT_KST_EXT.1 Key Storage

FPT_KST_EXT.1.1

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

5.1.6.10 PP_MDF_V3.3 & MOD_BIO_V1.1:FPT_KST_EXT.2 No Key Transmission

FPT_KST_EXT.2.1

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

5.1.6.11 PP_MDF_V3.3:FPT_KST_EXT.3 No Plaintext Key Export

FPT_KST_EXT.3.1

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

5.1.6.12 PP_MDF_V3.3:FPT_NOT_EXT.1 Self-Test Notification

FPT_NOT_EXT.1.1

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

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

5.1.6.13 MOD_BIO_V1.1:FPT_PBT_EXT.1 Protection of biometric template

FPT_PBT_EXT.1.1

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

(TD0714 applied)

5.1.6.14 PP_MDF_V3.3:FPT_STM.1 Reliable time stamps

FPT_STM.1.1

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

5.1.6.15 PP_MDF_V3.3:FPT_TST_EXT.1 TSF Cryptographic Functionality Testing

FPT_TST_EXT.1.1

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

5.1.6.16 PP_MDF_V3.3:FPT_TST_EXT.2/PREKERNEL TSF Integrity Checking (Pre-Kernel)

FPT_TST_EXT.2.1/PREKERNEL

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

5.1.6.17 PP_MDF_V3.3:FPT_TST_EXT.2/POSTKERNEL TSF Integrity Checking (Post-Kernel)

FPT_TST_EXT.2.1/POSTKERNEL

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

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

FPT_TST_EXT.3.1/WLAN

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

FPT_TST_EXT.3.2/WLAN

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

5.1.6.19 PP_MDF_V3.3:FPT_TUD_EXT.1 Trusted Update: TSF Version Query

FPT_TUD_EXT.1.1

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

FPT_TUD_EXT.1.2

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

FPT_TUD_EXT.1.3

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

5.1.6.20 PP_MDF_V3.3:FPT_TUD_EXT.2 TSF Update Verification

FPT_TUD_EXT.2.1

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

FPT_TUD_EXT.2.2

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

FPT_TUD_EXT.2.3

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

5.1.6.21 PP_MDF_V3.3:FPT_TUD_EXT.3 Application Signing

FPT_TUD_EXT.3.1

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

5.1.6.22 PP_MDF_V3.3:FPT_TUD_EXT.6 Trusted Update Verification

FPT_TUD_EXT.6.1

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

5.1.7 TOE Access (FTA)

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

FTA_SSL_EXT.1.1

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

FTA_SSL_EXT.1.2

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

FTA_SSL_EXT.1.3

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

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

5.1.7.2 PP_MDF_V3.3:FTA_TAB.1 Default TOE Access Banners

FTA_TAB.1.1

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

5.1.7.3 MOD_WLANC_V1.0:FTA_WSE_EXT.1 Wireless Network Access

FTA_WSE_EXT.1.1

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

5.1.8 Trusted Path/Channels (FTP)

5.1.8.1 MOD_BT_V1.0:FTP_BLT_EXT.1 Bluetooth Encryption

FTP_BLT_EXT.1.1

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

FTP_BLT_EXT.1.2

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

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

FTP_BLT_EXT.2.1

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

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

FTP_BLT_EXT.3.1/BR

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

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

FTP_BLT_EXT.3.1/LE

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

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

FTP_ITC.1.1/WLAN

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

FTP_ITC.1.2/WLAN

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

FTP_ITC.1.3/WLAN

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

5.1.8.6 PP_MDF_V3.3:FTP_ITC_EXT.1 Trusted Channel Communication

FTP_ITC_EXT.1.1

The TSF shall use

- 802.11-2012 in accordance with the [PP-Module for Wireless LAN Clients, version 1.0],
- 802.1X in accordance with the [PP-Module for Wireless LAN Clients, version 1.0],
- EAP-TLS in accordance with the [PP-Module for Wireless LAN Clients, version 1.0],
- mutually authenticated TLS in accordance with [the Functional Package for Transport Layer Security (TLS), version 1.1]

and [

- ***HTTPS***

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

FTP_ITC_EXT.1.2

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

FTP_ITC_EXT.1.3

The TSF shall initiate communication via the trusted channel for wireless access point connections, administrative communication, configured enterprise connections, and [***OTA updates***].

5.2 TOE Security Assurance Requirements

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

Requirement Class	Requirement Component
ADV: Development	ADV_FSP.1: Basic Functional Specification
AGD: Guidance documents	AGD_OPE.1: Operational User Guidance
	AGD_PRE.1: Preparative Procedures
ALC: Life-cycle support	ALC_CMC.1: Labelling of the TOE
	ALC_CMS.1: TOE CM Coverage
	ALC_TSU_EXT.1: Timely Security Updates
ATE: Tests	ATE_IND.1: Independent Testing - Conformance
AVA: Vulnerability assessment	AVA_VAN.1: Vulnerability Survey

Table 17 - Assurance Components

5.2.1 Development (ADV)

5.2.1.1 ADV_FSP.1 Basic Functional Specification

ADV_FSP.1.1D

The developer shall provide a functional specification.

ADV_FSP.1.2D

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

ADV_FSP.1.1C

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

ADV_FSP.1.2C

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

ADV_FSP.1.3C

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

ADV_FSP.1.4C

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

ADV_FSP.1.1E

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

ADV_FSP.1.2E

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

5.2.2 Guidance Documents (AGD)

5.2.2.1 AGD_OPE.1 Operational User Guidance

AGD_OPE.1.1D

The developer shall provide operational user guidance.

AGD_OPE.1.1C

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

AGD_OPE.1.2C

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

AGD_OPE.1.3C

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

AGD_OPE.1.4C

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

AGD_OPE.1.5C

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

AGD_OPE.1.6C

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

AGD_OPE.1.7C

The operational user guidance shall be clear and reasonable.

AGD_OPE.1.1E

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

5.2.2.2 AGD_PRE.1 Preparative Procedures

AGD_PRE.1.1D

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

AGD_PRE.1.1C

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

AGD_PRE.1.2C

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

AGD_PRE.1.1E

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

AGD_PRE.1.2E

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

5.2.3 Life-cycle support (ALC)

5.2.3.1 ALC_CMC.1 Labeling of the TOE

ALC_CMC.1.1D

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

ALC_CMC.1.1C

The TOE shall be labelled with its unique reference.

ALC_CMC.1.1E

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

5.2.3.2 ALC_CMS.1 TOE CM Coverage

ALC_CMS.1.1D

The developer shall provide a configuration list for the TOE.

ALC_CMS.1.1C

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

ALC_CMS.1.2C

The configuration list shall uniquely identify the configuration items.

ALC_CMS.1.1E

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

5.2.3.3 ALC_TSU_EXT.1 Timely Security Updates

ALC_TSU_EXT.1.1D

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

ALC_TSU_EXT.1.1C

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

ALC_TSU_EXT.1.2C

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

ALC_TSU_EXT.1.3C

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

ALC_TSU_EXT.1.4C

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

ALC_TSU_EXT.1.1E

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

5.2.4 Tests (ATE)

5.2.4.1 ATE_IND.1 Independent Testing - Conformance

ATE_IND.1.1D

The developer shall provide the TOE for testing.

ATE_IND.1.1C

The TOE shall be suitable for testing.

ATE_IND.1.1E

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

ATE_IND.1.2E

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

5.2.5 Vulnerability assessment (AVA)

5.2.5.1 AVA_VAN.1 Vulnerability Survey

AVA_VAN.1.1D

The developer shall provide the TOE for testing.

AVA_VAN.1.1C

The TOE shall be suitable for testing.

AVA_VAN.1.1E

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

AVA_VAN.1.2E

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

AVA_VAN.1.3E

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

6 TOE Summary Specification

This chapter describes the security functions:

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

6.1 Security audit

PP_MDF_V3.3:FAU_GEN.1:
MOD_BT_V1.0:FAU_GEN.1/BT:
MOD_WLANC_V1.0:FAU_GEN.1/WLAN:
The TOE uses different forms of logs to meet all the required management logging events specified in Table 2 and Table 3 of the PP_MDF_V3.3, Table 2 of the MOD_BT_V1.0 and Table 2 of the MOD_WLANC_V1.0:

1. SecurityLog events
2. Logcat events

Each of the above logging methods are described below.

- *SecurityLog events*: A full list of all auditable events can be found here: https://developer.android.com/reference/android/app/admin/SecurityLog#constants_1. Values found in this list represent SecurityLog keywords used in this logging function along with a description of what the log means and any additional information/variables present in the audit record. Additionally, the following link provides the additional information that can be grabbed when an MDM requests a copy of the logs: https://developer.android.com/reference/android/app/admin/SecurityLog.SecurityEvent. Each log contains a keyword or phrase describing the event, the date and time of the event, and further event-specific values that provide success, failure, and other information relevant to the event.
- *Logcat events:* Similar to SecurityLog events, logcat events contain date, time, and further even-specific values within the logs. In addition, logcat events provide a value that maps to a user ID to identify which user caused the event that generated the log. Finally, logcat events are descriptive and do not require the administrator to know the template of the log to understand its values. Logcat events cannot be exported but can be viewed by an administrator via an MDM agent.

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

The following tables enumerate the events that the TOE audits:

Protection Profile	Table
PP_MDF_V3.3	Mandatory - Table 11 - PP_MDF_V3.3 Audit Events
	Additional (marked with (ADD)) - Table 11 - PP_MDF_V3.3 Audit Events
MOD_BT_V1.0	Table 12 - MOD_BT_V1.0 Audit Events
MOD_WLANC_V1.0	Table 13 - MOD_WLANC_V1.0 Audit Events

Table 18 - Audit Event Table References

The details of the events audited are included in section 9 of the Admin Guide.

Some audit records, while logged, are unavailable to the administrator due to a number of reasons. Such audits and their explanations are identified below:

- (ALL) FAU_GEN.1 – Shutdown of the audit functions: Upon log shutdown, the security log buffer is deallocated and no longer available to be read, rendering the viewing of such an audit unavailable for the administrator to view.
- PP_MDF_V3.3:FAU_GEN.1 – Shutdown of the Rich OS: Since security logs are stored in memory, a shutdown of the system clears the audit record that is generated stating that the system is shutting down.
- PP_MDF_V3.3:FPT_TST_EXT.1 – Failure of self-test: Self-tests take place prior to the initialization of audit records. While the self-test success/failure audit is queued up to be logged upon security logs being initialized, when a self-test failure occurs the boot process is halted prior to security logs being initialized.

PP_MDF_V3.3:FAU_SAR.1:
The TOE provides an MDM API to allow a Device-Owner MDM agent to read the security logs.

PP_MDF_V3.3:FAU_STG.1:
For security logs, the TOE stores all audit records in memory, making it only accessible to the logd daemon, and only device owner applications can call the MDM API to retrieve a copy of the logs. Additionally, only new logs can be added. There is no designated method allowing for the deletion or modification of logs already present in memory, but reading the security logs clears the buffer at the time of the read.

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

PP_MDF_V3.3:FAU_STG.4:
The SecurityLog and logcat are stored in memory in a circular log buffer of 10KB/64KB, respectively. Logcat storage is configurable, able to be set by an MDM API. There is no limit to the size that the logcat buffer can be configured to and it is limited to the size of the system's memory. Once the log is full, it begins overwriting the oldest message in its respective buffer and continues overwriting the oldest message with each new auditable event. These logs persist until either they are overwritten or the device is restarted.

6.2 Cryptographic support

PP_MDF_V3.3:FCS_CKM.1:
The TOE provides generation of asymmetric keys including:

Algorithm	Key/Curve Sizes	Usage
RSA, FIPS 186-4	2048/3072/4096	API/Application & Sensitive Data Protection (FDP_DAR_EXT.2)
ECDSA, FIPS 186-4	P-256/384/521	API/Application
ECDHE keys (not domain parameters)	P-256/384	TLS KeyEx (WPA2/WPA3 w/ EAP-TLS & HTTPS)

Table 19 - Asymmetric Key Generation

The TOE's cryptographic algorithm implementations have received NIST algorithm certificates (see the tables in FCS_COP.1 for all of the TOE'S algorithm certificates). The TOE itself does not generate any RSA/ECDSA authentication key pairs for TOE functionality (the user or administrator must load certificates for use with WPA2/WPA3 with EAP-TLS authentication); however, the TOE provides key generation APIs to mobile applications to allow them to generate RSA/ECDSA key pairs. The TOE generates only ECDH key pairs (as BoringSSL does not support DH/DHE cipher suites) and does not generate domain parameters (curves) for use in TLS Key Exchange.

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

MOD_WLANC_V1.0:FCS_CKM.1/WPA:
The TOE adheres to IEEE 802.11-2012 for key generation. The TOE's wpa_supplicant provides PRF384, PRF512 and PRF704 for derivation of 128-bit, 192-bit and 256-bit AES Temporal Keys (using the HMAC implementation provided by BoringSSL) and employs its BoringSSL AES-256 DRBG when generating random values used in the EAP-TLS and 802.11 4-way handshake. The TOE supports the AES-128 CCMP and AES-192/AES-256 GCMP encryption modes. The TOE has successfully completed certification (including WPA2/WPA3 Enterprise) and received Wi-Fi CERTIFIED Interoperability Certificates from the Wi-Fi Alliance. The Wi-Fi Alliance maintains a website providing further information about the testing program: http://www.wi-fi.org/certification.

Device Name	Model Number	Wi-Fi Alliance Certificate Numbers
Pixel 7 Pro	GVU6C, G03Z5, GQML3	WFA119877, WFA119869
Pixel 7	GE2AE, GFE4J, GP4BC	WFA119878, WFA119753
Pixel 6 Pro	GF5KQ, G8V0U, GLU0G	WFA113888, WFA113887
Pixel 6	GR1YH, GB7N6, G9S9B	WFA113889, WFA111718
Pixel 6a	GX7AS, GB62Z, G1AZG, GB17L	WFA117809, WFA117592
Pixel 5	GD1YQ(NA), GTT9Q(ROW), G5NZ6(JP)	WFA100151
Pixel 5a-5G	G1F8F, G4S1M	WFA102288, WFA102271
Pixel 4a-5G	G024a-5G(NA), G025I(ROW), G025H(JP), G6QU3	WFA99858, WFA100153
Pixel 4a	G025J(NA), G025N(ROW), G025M(JP)	WFA96515

Table 20 - Wi-Fi Alliance Certificates

PP_MDF_V3.3:FCS_CKM.2/UNLOCKED:
The TOE performs key establishment as part of EAP-TLS and TLS session establishment. Table 19 - Asymmetric Key Generation enumerates the TOE's supported key establishment implementations (RSA/ECDH for TLS/EAP-TLS).

PP_MDF_V3.3:FCS_CKM.2.1/LOCKED:
The TOE provides an SDP library for applications that uses a hybrid crypto scheme based on 4096-bit RSA based key establishment. Applications can utilize this library to implement SDP that encrypts incoming data received while the phone is locked in a manner compliant with this requirement.

MOD_WLANC_V1.0:FCS_CKM.2/WLAN:
The TOE adheres to RFC 3394 and 802.11-2012 standards and unwraps the GTK (sent encrypted with the WPA2/WPA3 KEK using AES Key Wrap in an EAPOL-Key frame). The TOE, upon receiving an EAPOL frame, will subject the frame to a number of checks (frame length, EAPOL version, frame payload size, EAPOL-Key type, key data length, EAPOL-Key CCMP descriptor version, and replay counter) to ensure a proper EAPOL message and then decrypt the GTK using the KEK, thus ensuring that it does not expose the Group Temporal Key (GTK).

PP_MDF_V3.3:FCS_CKM_EXT.1:
The TOE includes a Root Encryption Key (REK) stored in a 256-bit fuse bank within the application processor. The TOE generates the REK/fuse value during manufacturing using its hardware DRBG. The application processor protects the REK by preventing any direct observation of the value and prohibiting any ability to modify or update the value. The application processor loads the fuse value into an internal hardware crypto register and the Trusted Execution Environment (TEE) provides trusted applications the ability to derive KEKs from the REK (using an SP 800-108 KDF to combine the REK with a salt). Additionally, the when the REK is loaded, the fuses for the REK become locked, preventing any further changing or loading of the REK value. The TEE does not allow trusted applications to use the REK for encryption or decryption, only the ability to derive a KEK from the REK. The TOE includes a TEE application that calls into the TEE in order to derive a KEK from the 256-bit REK/fuse value and then only permits use of the derived KEK for encryption and decryption as part of the TOE key hierarchy. More information regarding Trusted Execution Environments may be found here: http://www.globalplatform.org/mediaguidetee.asp.

PP_MDF_V3.3:FCS_CKM_EXT.2:
The TOE utilizes its approved RBGs to generate DEKs. When generating AES keys for itself (for example, the TOE's sensitive data encryption keys or for the Secure Key Storage), the TOE utilizes the RAND_bytes() API call from its BoringSSL AES-256 CTR_DRBG to generate a 256-bit AES key. The TOE also utilizes that same DRBG when servicing API requests from mobile applications wishing to generate AES keys (either 128 or 256-bit).

In all cases, the TOE generates DEKs using a compliant RBG seeded with sufficient entropy so as to ensure that the generated key cannot be recovered with less work than a full exhaustive search of the key space.

PP_MDF_V3.3:FCS_CKM_EXT.3:
The TOE takes the user-entered password and conditions/stretches this value before combining the factor with other KEK.

The TOE generates all non-derived KEKs using the RAND_bytes() API call from its BoringSSL AES-256 CTR_DRBG to ensure a full 128/256-bits of strength for asymmetric/symmetric keys, respectively. And the TOE combines KEKs by encrypting one KEK with the other so as to preserve entropy.

PP_MDF_V3.3:FCS_CKM_EXT.4:
The TOE clears sensitive cryptographic material (plaintext keys, authentication and biometric data, and other security parameters) from memory when no longer needed or when transitioning to the device's locked state (in the case of the Sensitive Data Protection keys). Public keys (such as the one used for Sensitive Data Protection) can remain in memory when the phone is locked, but all crypto-related private keys are evicted from memory upon device lock. No plaintext cryptographic material resides in the TOE's Flash as the TOE encrypts all keys stored in Flash. When performing a full wipe of protected data, the TOE cryptographically erases the protected data by clearing the Data-At-Rest DEK. Because the Android Keystore of the TOE resides within the user data partition, the TOE effectively cryptographically erases those keys when clearing the Data-At-Rest DEK. In turn, the TOE clears the Data-At-Rest DEK and Secure Key Storage SEK through a secure direct overwrite (BLKSECDISCARD ioctl) of the wear-leveled Flash memory containing the key followed by a read-verify.

PP_MDF_V3.3:FCS_CKM_EXT.5:
The TOE stores all protected data in encrypted form within the user data partition (either protected data or sensitive data). Upon request, the TOE cryptographically erases the Data-At-Rest DEK protecting the user data partition and the SDP Master KEK protecting sensitive data files in the user data partition, clears those keys from memory, reformats the partition, and then reboots. The TOE's clearing of the keys follows the requirements of FCS_CKM_EXT.4.

PP_MDF_V3.3:FCS_CKM_EXT.6:
The TOE generates salt nonces (which are just salt values used in WPA2/WPA3) using its /dev/urandom.

Salt value and size	RBG origin	Salt storage location
User password salt (128-bit)	BoringSSL's AES-256 CTR_DRBG	Flash filesystem
TLS client_random (256-bit)	BoringSSL's AES-256 CTR_DRBG	N/A (ephemeral)
TLS pre_master_secret (384-bit)	BoringSSL's AES-256 CTR_DRBG	N/A (ephemeral)
WPA2/WPA3 4-way handshake supplicant nonce (SNonce)	BoringSSL's AES-256 CTR_DRBG	N/A (ephemeral)

Table 21 - Salt Nonces

MOD_BT_V1.0:FCS_CKM_EXT.8
The TOE generates new ECDH key pairs every time a connection with a Bluetooth device is established.

PP_MDF_V3.3:FCS_COP.1/ENCRYPT:
PP_MDF_V3.3:FCS_COP.1/HASH:
PP_MDF_V3.3:FCS_COP.1/SIGN:
PP_MDF_V3.3:FCS_COP.1/KEYHMAC:
PP_MDF_V3.3:FCS_COP.1/CONDITION:
The TOE implements cryptographic algorithms in accordance with the following NIST standards and has received the following CAVP algorithm certificates. These algorithms are in software and hardware, depending on the implementation.

The TOE's BoringSSL Library (version 2022061300 with both Processor Algorithm Accelerators (PAA) and without PAA) provides the following algorithms as validated on Android 13:

SFR	Algorithm	Keys	NIST Standard	Cert#
FCS_CKM.1	RSA IFC Key Generation	2048, 3072, 4096	FIPS 186-4, RSA	A2811
	ECDSA ECC Key Generation	P256, P384, P521	FIPS 186-4, ECDSA	
FCS_CKM.2	RSA-based Key Exchange		Vendor affirm 800-56B	
	ECC-based Key Exchange	P256, P384, P521	SP 800-56A, CVL KAS ECC	
FCS_COP.1/ENCRYPT	AES CBC, GCM, KW	128/256	FIPS 197, SP 800-38A/D/F	
FCS_COP.1/HASH	SHA Hashing	1/256/ 384/512	FIPS 180-4	
FCS_COP.1/SIGN	RSA Sign/Verify	2048, 3072, 4096	FIPS 186-4, RSA	
	ECDSA Sign/Verify	P256, P384, P521	FIPS 186-4, ECDSA	
FCS_COP.1/KEYHMAC	HMAC-SHA 1/256/384/512	1/256/ 384/512	FIPS 198-1 & 180-4	
FCS_RBG_EXT.1	DRBG Bit Generation	256	SP 800-90A (Counter)	

Table 22 - BoringSSL Cryptographic Algorithms

Android's LockSettings service (version 77561fc30db9aedc1f50f5b07504aa65b4268b88) as validated on Android 13 provides the TOE's SP 800-108 key based key derivation function for deriving KEKs.

SFR	Algorithm	Keys	NIST Standard	Cert#
FCS_CKM_EXT.3	LockSettings service KBKDF	256	SP 800-108	A2168

Table 23 - LockSettings Service KDF Cryptographic Algorithms

The following algorithms used in the TOE are provided by hardware components of the device. As these algorithms are implemented solely in hardware, they do not utilize Android 13 as their operating environment, but provide lower-level services upon which some of the security functionality rests.

All Pixel devices include a Titan security chip, which provides cryptographic algorithm implementations within a secure microprocessor supporting the Android Keystore StrongBox HAL. Table 24 provides a list of the supported chips in the devices (devices not listed do not support the StrongBox HAL). Titan security chips support the Android Keystore StrongBox hardware abstraction layer, and as such, provides secure key generation, digital signatures, and other cryptographic functions in a mutual hardware Keystore.

Device	Chip	Hardware	Firmware
Pixel 7 Pro/7	Titan M2	H1D3M	MP-SC-05a
Pixel 5/5a-5G	Titan M	H1C2M	57042c8aa
Pixel 4a-5G/4a	Titan M	H1C2M	57042c8aa

Table 24 - Titan Security Chipsets

SFR	Algorithm	Keys	NIST Standard	Cert#
FCS_CKM.1	RSA IFC Key Generation	2048	FIPS 186-4, RSA	A2951
	ECDSA ECC Key Generation	P256	FIPS 186-4, ECDSA	
FCS_COP.1/ENCRYPT	AES CBC, GCM	128/256	FIPS 197, SP 800-38A/D	
FCS_COP.1/HASH	SHA Hashing	256	FIPS 180-4	
FCS_COP.1/SIGN	RSA Sign/Verify	2048	FIPS 186-4, RSA	
	ECDSA Sign/Verify	P256	FIPS 186-4, ECDSA	
FCS_COP.1/KEYHMAC	HMAC-SHA	256	FIPS 198-1 & 180-4	
FCS_RBG_EXT.1	DRBG Bit Generation	256	SP 800-90A (Counter)	
FCS_CKM_EXT.3	KBKDF	256	SP 800-108	

Table 25 - Titan M2 Hardware Cryptographic Algorithms

SFR	Algorithm	Keys	NIST Standard	Cert#
FCS_CKM.1	RSA IFC Key Generation	2048	FIPS 186-4, RSA	C1969
	ECDSA ECC Key Generation	P256	FIPS 186-4, ECDSA	
FCS_COP.1/ENCRYPT	AES 128/256 CBC, GCM	128/256	FIPS 197, SP 800-38A/D	
FCS_COP.1/HASH	SHA Hashing	256	FIPS 180-4	
FCS_COP.1/SIGN	RSA Sign/Verify	2048	FIPS 186-4, RSA	
	ECDSA Sign/Verify	P256	FIPS 186-4, ECDSA	
FCS_COP.1/KEYHMAC	HMAC-SHA 256	256	FIPS 198-1 & 180-4	
FCS_RBG_EXT.1	DRBG Bit Generation	256	SP 800-90A (Counter)	
FCS_CKM_EXT.3	KBKDF	256	SP 800-108	

Table 26 - Titan M Hardware Cryptographic Algorithms

The devices have unique Wi-Fi chipsets. All Wi-Fi chipsets provide AES-CCMP 128-bit encryption to meet FCS_COP.1/ENCRYPT that meet the NIST Standards FIPS 197, SP 800-38C.

Device	Wi-Fi Chipset	Cert#
Pixel 7 Pro/7	BCM4389	5926 C1025
Pixel 6 Pro/6/6a		
Pixel 5/5a-5G	WCN3998-1	4748
Pixel 4a-5G/4a	WCN3998	

Table 27 - Wi-Fi Chipsets

The Pixel 7 Pro/7 application processor (Google Tensor G2) provides cryptographic algorithms (marked as SoC). The Google Tensor UFS Inline Storage Engine is version 1.0.0 with hardware sf_crypt_fmp_fx8_v4.10. The Google Trusty TEE is version 9004426 (marked as TEE).

SFR	Component	Algorithm	Keys	NIST Standard	Cert#
FCS_COP.1/ENCRYPT	Storage	AES XTS	128/256	FIPS 197, SP 800-38E	A2937
FCS_COP.1/HASH	Storage	SHA Hashing	256	FIPS 180-4	A2938
FCS_COP.1/KEYHMAC	Storage	HMAC-SHA	256	FIPS 198-1 & 180-4	
FCS_COP.1/ENCRYPT	SoC	AES CBC	128/256	FIPS 197, SP 800-38A	A2923
FCS_COP.1/HASH	SoC	SHA Hashing	256	FIPS 180-4	
FCS_COP.1/KEYHMAC	SoC	HMAC-SHA	256	FIPS 198-1 & 180-4	
FCS_CKM_EXT.3	TEE	KBKDF	256	SP 800-108	A2928
FCS_COP.1/ENCRYPT	TEE	AES GCM	128/256	FIPS 197, SP 800-38D	
FCS_COP.1/HASH	TEE	SHA Hashing	256	FIPS 180-4	
FCS_COP.1/KEYHMAC	TEE	HMAC-SHA	256	FIPS 198-1 & 180-4	

Table 28 - Google Tensor G2 Hardware Cryptographic Algorithms

The Pixel 6 Pro/6/6a application processor (Google Tensor) provides cryptographic algorithms (marked as SoC). The Google Tensor UFS Inline Storage Engine is version de8b6c8621; the Hash and Keyed Hash functions are implemented in software, not hardware (marked as Storage). The Google Trusty TEE is version 7623683 (marked as TEE).

SFR	Component	Algorithm	Keys	NIST Standard	Cert#
FCS_COP.1/ENCRYPT	Storage	AES XTS	128/256	FIPS 197, SP 800-38E	A1981
FCS_COP.1/HASH	Storage	SHA Hashing	256	FIPS 180-4	
FCS_COP.1/KEYHMAC	Storage	HMAC-SHA	256	FIPS 198-1 & 180-4	
FCS_COP.1/ENCRYPT	SoC	AES CBC	128/256	FIPS 197, SP 800-38A	A1980
FCS_COP.1/HASH	SoC	SHA Hashing	256	FIPS 180-4	
FCS_COP.1/KEYHMAC	SoC	HMAC-SHA	256	FIPS 198-1 & 180-4	
FCS_RBG_EXT.1	SoC	DRBG Bit Generation	256	SP 800-90A (Counter)	
FCS_CKM_EXT.3	TEE	KBKDF	256	SP 800-108	A1982
FCS_COP.1/ENCRYPT	TEE	AES GCM	128/256	FIPS 197, SP 800-38D	
FCS_COP.1/HASH	TEE	SHA Hashing	256	FIPS 180-4	
FCS_COP.1/KEYHMAC	TEE	HMAC-SHA	256	FIPS 198-1 & 180-4	

Table 29 - Google Tensor Hardware Cryptographic Algorithms

The Pixel 5/5a-5G/4a-5G application processor (Snapdragon 765) provides cryptographic algorithms using the Qualcomm Technologies, Inc. Crypto Engine Core v5.5.1 (marked as SoC). The Qualcomm Technologies, Inc. Inline Crypto Engine (UFS) v3.1.0 provides storage encryption (marked as Storage).

SFR	Component	Algorithm	Keys	NIST Standard	Cert#
FCS_COP.1/ENCRYPT	SoC	AES CBC	128/256	FIPS 197, SP 800-38A	A242
FCS_COP.1/ENCRYPT	Storage	AES XTS	128/256	FIPS 197, SP 800-38E	C1553 C1552
FCS_COP.1/HASH	SoC	SHA Hashing	256	FIPS 180-4	A896
FCS_COP.1/HASH	SoC (DRBG)	SHA Hashing	256	FIPS 180-4	A214
FCS_COP.1/KEYHMAC	SoC	HMAC-SHA	256	FIPS 198-1 & 180-4	A896
FCS_RBG_EXT.1	SoC (DRBG)	DRBG Bit Generation	256	SP 800-90A (Hash-256)	A50
FCS_CKM_EXT.3	SoC	KBKDF	128	SP 800-108	A244

Table 30 - Snapdragon 765 Hardware Cryptographic Algorithms

The Pixel 4a application processor (Snapdragon 730) provides cryptographic algorithms using the Qualcomm Technologies, Inc. Crypto Engine Core v5.4.1 (marked as SoC). The Qualcomm Technologies, Inc. Inline Crypto Engine (UFS) v3.1.0 provides storage encryption (marked as Storage). The Qualcomm Snapdragon 730 processor uses the same Crypto/PRNG core as the Snapdragon 845 listed in the algorithm certificate, but remains valid for the implementation within the Snapdragon 730.

SFR	Component	Algorithm	Keys	NIST Standard	Cert#
FCS_COP.1/ENCRYPT	SoC	AES CBC	128/256	FIPS 197, SP 800-38A	4959
FCS_COP.1/ENCRYPT	Storage	AES XTS	128/256	FIPS 197, SP 800-38E	4958 4957
FCS_COP.1/HASH	SoC	SHA Hashing	256	FIPS 180-4	4049
FCS_COP.1/HASH	SoC (DRBG)	SHA Hashing	256	FIPS 180-4	4048 4047
FCS_COP.1/KEYHMAC	SoC	HMAC-SHA	256	FIPS 198-1 & 180-4	3305

SFR	Component	Algorithm	Keys	NIST Standard	Cert#
FCS_RBG_EXT.1	SoC (DRBG)	DRBG Bit Generation	256	SP 800-90A (Hash-256)	1788
FCS_CKM_EXT.3	SoC	KBKDF	256	SP 800-108	KDF171

Table 31 - Snapdragon 730 Hardware Cryptographic Algorithms

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

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

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

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

The password-derived key is combined with the hardware REK in storage, preventing the ability to perform offline attacks, and online attacks are limited due to the TOE's configuration of the maximum no more than 50 incorrect password attempts (with at least 4 character passwords). The use of the password derivation function in combination with the device configuration forces the attacker to only be able to perform an exhaustive key search to unlock the device without access to the password, and then subject to the configured limit of 50 or less attempts before the device is wiped.

The following scrypt diagram shows how the password and salt are used with PBKDFv2 and ROMix to fulfil the requirements for password conditioning.

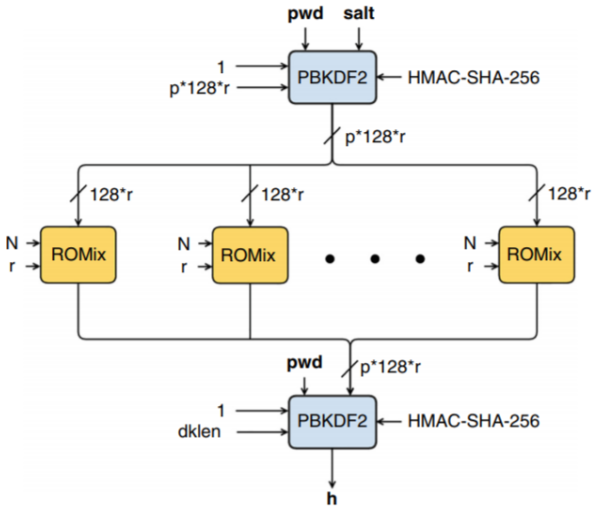

Figure 1 - Password Conditioning

The resulting derived key from this operation is used to decrypt the FBE and to derive the User Keystore Daemon Value.

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

PP_MDF_V3.3:FCS_HTTPS_EXT.1:
The TOE supports the HTTPS protocol (compliant with RFC 2818) so that (mobile and system) applications executing on the TOE can act as HTTPS clients and securely connect to external servers using HTTPS. Administrators have no credentials and cannot use HTTPS or TLS to establish administrative sessions with the TOE as the TOE does not provide any such capabilities.

PP_MDF_V3.3:FCS_IV_EXT.1:
The TOE generates IVs by reading from /dev/urandom for use with all keys. In all cases, the TOE uses /dev/urandom and generates the IVs in compliance with the requirements of table 11 of PP_MDF_V3.3.

PP_MDF_V3.3:FCS_RBG_EXT.1:
The TOE provides a number of different RBGs including:

1. An Application Processor DRBG in hardware:
    a. An AES-256 CTR_DRBG in Google Tensor processors
    b. A SHA-256 Hash_DRBG in Qualcomm Snapdragon processors
2. An AES-256 CTR_DRBG provided by BoringSSL. This is the only accredited and supported DRBG present in the system and available to independently developed applications. As such, the TOE

provides mobile applications access (through an Android Java API) to random data drawn from its AES-256 CTR_DRBG

3. An SHA-256 HMAC_DRBG provided by the Titan security chip

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

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

The TOE seeds the Titan security chip SHA-256 HMAC_DRBG with entropy from its hardware noise and then uses the DRBG when generating keys and cryptographic random values.

PP_MDF_V3.3:FCS_SRV_EXT.1:
The TOE provides applications access to the cryptographic operations including encryption (AES), hashing (SHA), signing and verification (RSA & ECDSA), key hashing (HMAC), keyed message digests (HMAC-SHA-256), generation of asymmetric keys for key establishment (RSA and ECDH), and generation of asymmetric keys for signature generation and verification (RSA, ECDSA). The TOE provides access through the Android operating system's Java API, through the native BoringSSL API, and through the application processor module (user and kernel) APIs.

PP_MDF_V3.3:FCS_SRV_EXT.2:
The TOE provides applications with APIs to perform the functions referenced in FCS_COP.1/ENCRYPT and FCS_COP.1/SIGN.

PP_MDF_V3.3:FCS_STG_EXT.1:
The TOE provides the user, administrator and mobile applications the ability to import and use asymmetric public and private keys into the TOE's software-based Secure Key Storage. Certificates are stored in files using UID-based permissions and an API virtualizes the access. Additionally, the user and administrator can request the TOE to destroy the keys stored in the Secure Key Storage. While normally mobile applications cannot use or destroy the keys of another application, applications that share a common application developer (and are thus signed by the same developer key) may do so. In other words, applications with a common developer (and which explicitly declare a shared UUID in their application manifest) may use and destroy each other's keys located within the Secure Key Storage.

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

The TOE also provides an extension to Android Keystore, StrongBox, which allows mobile applications to specify that keys be stored in the Pixel's hardware-based key storage (provided by the Titan M security chip[3]).

PP_MDF_V3.3:FCS_STG_EXT.2:

[3] StrongBox is not available on the Pixel 6 Pro/6/6a

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

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

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

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

PP_MDF_V3.3:FCS_STG_EXT.3:
The TOE protects the integrity of all DEKs and KEKs (including LTTCKM keys) stored in Flash by using authenticated encryption/decryption methods (CCM, GCM).

PKG_TLS_V1.1:FCS_TLS_EXT.1:
PKG_TLS_V1.1:FCS_TLSC_EXT.1:
PKG_TLS_V1.1:FCS_TLSC_EXT.2:
The TOE provides mobile applications (through its Android API) the use of TLS version 1.2 as a client, including support for the selections chosen in section 5 for FCS_TLSC_EXT.1 (and the TOE requires no configuration other than using the appropriate library APIs as described in the Admin Guidance).

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

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

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

PKG_TLS_V1.1:FCS_TLSC_EXT. 4:
The TOE includes the ‘renegotiation_info’ TLS extension in its TLS client hello message.

PKG_TLS_V1.1:FCS_TLSC_EXT.5:
The TOE in its evaluated configuration and, by design, supports elliptic curves for TLS (P-256 and P-384) and has a fixed set of supported curves (thus the admin cannot and need not configure any curves).

MOD_WLANC_V1.0:FCS_TLSC_EXT.1/WLAN:
MOD_WLANC_V1.0:FCS_TLSC_EXT.2/WLAN:

The TSF supports TLS versions 1.1, and 1.2 and also supports the selected ciphersuites utilizing SHA-1, SHA-256, and SHA-384 (see the selections in section 5 for FCS_TLSC_EXT.1/WLAN) for use with EAP-TLS as part of WPA2/WPA3. The TOE in its evaluated configuration and, by design, supports only evaluated elliptic curves (P-256 & P-384 and no others) and has a fixed set of supported curves (thus the admin cannot and need not configure any curves).

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

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

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

MOD_WLANC_V1.0:FCS_WPA_EXT.1:
The TSF support WPA2 and WPA3 security types for Wi-Fi networks.

6.3 User data protection

PP_MDF_V3.3:FDP_ACF_EXT.1:
The TOE provides a mechanism based on the use of assigned permissions to specify the level of access any application may have to any system service. A system service may have multiple permissions associated with it, depending on the functionality of the service (for example read and write access may be separate controls on one service while both may be combined into a single control on another service). When an application wants to access the system service in question, the calling application must be granted access to the permission by the user.

Some permissions are granted automatically for applications that are installed by Google (these are only for Google applications and are not provided for any third party applications) while all the user of the device must authorize other permissions. Applications using API Level 23 (Android 6.0) or higher (the current API Level is 33) will prompt the user to grant the permission the first time the permission is requested by the application. Applications written to older API Levels will prompt the user for all permissions the first time the application runs. If the user has approved the permission persistently, it will be allowed every time the application runs, but if the user only approved the permission for one time use, the user will be prompted to approve access every time the permission is requested by the application.

Permissions in API Level 33 are assigned a protectionLevel based on the implied potential risk to accessing data protected by the permission. The protectionLevel is divided into two types: base permissions and protection flags. Base permissions are associated with the level of risk while the flags are modifiers that may provide context or refinement of the base permission.

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

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

2. Dangerous - A higher-risk permission that would give a requesting application access to private user data or control over the device that can negatively impact the user. Because this type of permission introduces potential risk, the system cannot automatically grant it to the requesting application. For example, any dangerous permissions requested by an application will be displayed to the user and require confirmation before proceeding or some other approach can be taken to avoid the user automatically allowing the use of such facilities.
3. Signature - A permission that the system is to grant only if the requesting application is signed with the same certificate as the application that declared the permission. If the certificates match, the system automatically grants the permission without notifying the user or asking for the user's explicit approval.
4. Internal - a permission that is managed internally by the system and only granted according to the protection flags.

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

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

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

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

Additionally, Android includes the following flags that layer atop the base categories:

1. privileged - this permission can also be granted to any applications installed as privileged apps on the system image. Please avoid using this option, as the signature protection level should be sufficient for most needs and works regardless of exactly where applications are installed. This permission flag is used for certain special situations where multiple vendors have applications built in to a system image which need to share specific features explicitly because they are being built together.
2. system - Old synonym for 'privileged'.
3. development - this permission can also (optionally) be granted to development applications (e.g., to allow additional location reporting during beta testing).

4. appop - this permission is closely associated with an app op for controlling access.
5. pre23 - this permission can be automatically granted to apps that target API levels below API level 23 (Android 6.0).
6. installer - this permission can be automatically granted to system apps that install packages.
7. verifier - this permission can be automatically granted to system apps that verify packages.
8. preinstalled - this permission can be automatically granted to any application pre-installed on the system image (not just privileged apps) (the TOE does not prompt the user to approve the permission).

The Android 13 (Level 33) API (details found here https://developer.android.com/reference/packages) provides services to mobile applications.

While Android provides a large number of individual permissions, they are grouped into categories or features that provide similar functionality for the simplicity of the user interaction. These groupings do not affect the permissions themselves; it is only a way to group them together for the user presentation. Table 32 shows a series of functional categories centered on common functionality.

Service Features	Description
Sensitive I/O Devices & Sensors	Location services, Audio & Video capture, Body sensors
User Personal Information & Credentials	Contacts, Calendar, Call logs, SMS
Metadata & Device ID Information	IMEI, Phone Number
Data Storage Protection	App data, App cache
System Settings & Application Management	Date time, Reboot/Shutdown, Sleep, Force-close application, Administrator Enrollment
Wi-Fi, Bluetooth, USB Access	Wi-Fi, Bluetooth, USB tethering, debugging and file transfer
Mobile Device Management & Administration	MDM APIs
Peripheral Hardware	NFC, Camera, Headphones
Security & Encryption	Certificate/Key Management, Password, Revocation rules

Table 32 - Functional Categories

PP_MDF_V3.3:FDP_ACF_EXT.1.2:
Applications with a common developer have the ability to allow sharing of data between their applications. A common application developer can sign their generated APK with a common certificate or key and set the permissions of their application to allow data sharing. When the different applications' signatures match and the proper permissions are enabled, information can then be shared as needed.

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

with the personal profile; the enterprise profile MDM must allow for sharing of contacts before any information can be shared.

PP_MDF_V3.3:FDP_ACF_EXT.2:
The TOE allows an administrator to allow sharing of the enterprise profile address book with the normal profile. Each application group (profile) has its own calendar as well as keychain (keychain is the collection of user [not application] keys, and only the user can grant the user's applications access to use a given key in the user's keychain), thus Android's personal and work profiles do not share calendar appointments nor keys.

PP_MDF_V3.3:FDP_DAR_EXT.1:
The TOE provides Data-At-Rest AES-256 XTS hardware encryption for all data stored on the TOE in the user data partition (which includes both user data and TSF data). The TOE also has TSF data relating to key storage for TSF keys not stored in the system's Android Key Store. The TOE separately encrypts those TSF keys and data. Additionally, the TOE includes a read-only file system in which the TOE's system executables, libraries, and their configuration data reside. For its Data-At-Rest encryption of the data partition on the internal Flash (where the TOE stores all user data and all application data), the TOE uses an AES-256 bit DEK with XTS feedback mode to encrypt each file in the data partition using dedicated application processor hardware.

PP_MDF_V3.3:FDP_DAR_EXT.2:
The vendor provides the NIAPSEC library for Sensitive Data Protection (SDP) that application developers must use to opt-in for sensitive data protection. When developers opt-in for SDP, all data that is received on the device destined for that application is treated as sensitive. This library calls into the TOE to generate an RSA key that acts as a master KEK for the SDP encryption process. When an application that has opted-in for SDP receives incoming data while the device is locked, an AES symmetric DEK is generated to encrypt that data. The public key from the master RSA KEK above is then used to encrypt the AES DEK. Once the device is unlocked, the RSA KEK private key is re-derived and can be used to decrypt the AES DEK for each piece of information that was stored while the device was locked. The TOE then takes that decrypted data and re-encrypts it following FDP_DAR_EXT.1.

PP_MDF_V3.3:FDP_IFC_EXT.1:
The TOE will route all traffic other than traffic necessary to establish the VPN connection to the VPN gateway (when the gateway's configuration specifies so) when the Always-On-VPN is enabled. The TOE includes an interceptor kernel module that controls inbound and output packets. When a VPN is active, the interceptor will route all incoming packets to the VPN and conversely route all outbound packets to the VPN before they are output.

Note that when the TOE tries to connect to a Wi-Fi network, it performs a standard captive portal check which sends traffic that bypasses the full tunnel VPN configuration in order to detect whether the Wi-Fi network restricts Internet access until one has authenticated or agreed to usage terms through a captive portal. If the administrator wishes to deactivate the captive portal check (in order to prevent the plaintext traffic), they may do this by following the instructions in the Admin Guide.

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

PP_MDF_V3.3:FDP_STG_EXT.1:
The TOE's Trusted Anchor Database consists of the built-in certs and any additional user or admin/MDM loaded certificates. The built-in certs are individually stored in the device's read-only system image in the /system/etc/security/cacerts directory, and the user can individually disable certs through the Android user interface:

```
Settings -> Security -> Advanced settings -> Encryption &
credentials -> Trusted Credentials
```

Because the built-in CA certificates reside on the read-only system partition, the TOE places a copy of any disabled built-in certificate into the /data/misc/user/X/cacerts-removed/ directory, where 'X' represents the user's number (which starts at 0). The TOE stores added CA certificates in the corresponding /data/misc/user/X/cacerts-added/ directory and also stores a copy of the CA certificate in the user's Secure Key Storage (residing in the /data/misc/keystore/user_X/ directory). The TOE uses Linux file permissions that prevent any mobile application or entity other than the TSF from modifying these files. Only applications registered as an administrator (such as an MDM Agent Application) have the ability to access these files, staying in accordance to the permissions established in FMT_SMF.1 and FMT_MOF_EXT.1.

PP_MDF_V3.3:FDP_UPC_EXT.1/APPS:
PP_MDF_V3.3:FDP_UPC_EXT.1/BLUETOOTH:
The TOE provides APIs allowing non-TSF applications (mobile applications) the ability to establish a secure channel using TLS, HTTPS, and Bluetooth DR/EDR and LE. Additionally, the vendor provides the NIAPSEC library for application developers to use for Hostname Checking, Revocation Checking, and TLS Ciphersuite restriction. Application developers must utilize this library to ensure the device behaves in the evaluated configuration. Mobile applications can use the following Android APIs for TLS, HTTPS, and Bluetooth respectively:

SSL:

javax.net.ssl.SSLContext:

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

Developers then need to swap SocketFactory for SecureSocketFactory, part of a private library provided by Google.

Developers can request this library by emailing: niapsec@google.com

HTTPS:

javax.net.ssl.HttpsURLConnection:

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

Developers then need to swap HTTPSUrlConnections for SecureUrl part of a private library provided by Google.

Developers can request this library by emailing: niapsec@google.com

Bluetooth:

android.bluetooth:

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

6.4 Identification and authentication

PP_MDF_V3.3:FIA_AFL_EXT.1:
The TOE maintains in persistent storage, for each user, the number of failed password logins since the last successful login and upon reaching the maximum number of incorrect logins, the TOE performs a full wipe of all protected data (and in fact, wipes all user data). The maximum number of failed attempts is limited to only counting password attempts, as biometric attempts are not considered critical attempts that can trigger a wipe.

The administrator can adjust the number of failed login attempts that are allowed for the password unlock screen through an MDM. The possible values range from the default of ten failed logins to a value between 0 (deactivate wiping) and 50. When an authentication attempt occurs, the TOE first increments the failed login counter, and then checks the validity of the password by providing it to Android's Gatekeeper (which runs in the Trusted Execution Environment).

Any visual error to the user about a failed entry is displayed after the validation check. Android's Gatekeeper keeps this password counter in persistent secure storage and increments the counter before validating the password. Upon successful validation of the password, this counter is reset back to zero. If the login attempt is a failure and the counter is equal or greater than the specified value the device will be wiped. By storing the counter persistently, and by incrementing the counter prior to validating it, the TOE ensures a correct tally of failed attempts even if it loses power.

Table 33 lists the supported biometric fingerprint sensors for each device.

Device:	Back Mount	Under Display
Pixel 7 Pro/7		X
Pixel 6 Pro/6/6a		X
Pixel 5/5a-5G	X	
Pixel 4a-5G/4a	X	

Table 33 - Supported Biometric Modalities

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

MOD_BT_V1.0:FIA_BLT_EXT.1:
The TOE requires explicit user authorization before it will pair with a remote Bluetooth device. When pairing with another device, the TOE requires that the user either confirm that a displayed numeric passcode matches between the two devices or that the user enter (or choose) a numeric passcode that the peer device generates (or must enter). The TOE requires this authorization (via manual input) for

mobile application use of the Bluetooth trusted channel and in situations where temporary (non-bonded) connections are formed.

MOD_BT_V1.0:FIA_BLT_EXT.2:
The TOE does not allow any data transfers with remote devices that have not been paired or authorized by the user of the TOE. All Bluetooth connections require initial approval by the user in the user interface and cannot be done programmatically. Bluetooth pairing (RFCOMM connections) is completed by confirming/entering a displayed passcode in the user interface. TOE support for OBEX (OBject EXchange) through L2CAP (Logical Link Control and Adaptation Protocol) requires the user to explicitly authorize the transfer via a popup that will be displayed to the user.

MOD_BT_V1.0:FIA_BLT_EXT.3:
The TOE rejects duplicate Bluetooth connections by only allowing a single session per paired device. This ensures that when the TOE receives a duplicate session attempt while the TOE already has an active session with that device, then the TOE ignores the duplicate session.

MOD_BT_V1.0:FIA_BLT_EXT.4:
The TOE's Bluetooth host and controller supports Bluetooth Secure Simple Pairing and the TOE utilizes this pairing method when the remote host also supports it.

MOD_BT_V1.0:FIA_BLT_EXT.6:
The TOE requires explicit user authorization before granting trusted (paired) remote devices access to services associated with the OPP and MAP Bluetooth profiles.

MOD_BT_V1.0:FIA_BLT_EXT.7:
The TOE requires explicit user authorization before granting untrusted (unpaired) remote devices access to services associated with all Bluetooth profiles.

MOD_BIO_V1.1:FIA_MBE_EXT.1:
The TOE provides a mechanism to enroll user biometrics for authentication. The enrollment process requires the user to have a password set and to be authenticated successfully prior to being able to start the enrollment process. The user is able to enroll multiple fingerprints individually for use on supported devices.

MOD_BIO_V1.1:FIA_MBE_EXT.2/Fingerprint:
MOD_BIO_V1.1:FIA_MBV_EXT.2/Fingerprint:
The TSF determines the quality of a sample before using it to enroll or verify a user. The fingerprint systems utilize different capture sensors, but the data analysis of the quality is common. Fingerprint sample quality is determined using three measures.

The first measure is the completeness of the sample. For example, if the finger is offset from the sensor and only half the sensor acquires an image of the fingerprint, this partial print would be considered insufficient quality as there is not enough data to make a match. This can also be triggered by events like a too light or too hard touch on the sensor (either of which can cause parts of the sensor to not sense the minutia of the fingerprint).

The second measure is the clarity of the sample. Clarity is determined primarily by how clear the fingerprint minutia are in the sample image. Areas with no minutia (that is not a partial image), such as when a finger is dirty, or if the sensor is blocked, will be considered as insufficient quality. Repeated

attempts with similar results will trigger a user notification that the sensor or finger may be dirty so the user can consider remediation (such as wiping the sensor or washing the finger).

The third measure is a sufficient number of minutia within the sample. If the sample is complete and clear, then it is checked for a sufficient number of minutia to be used. The system looks for a large number of data points from which to build the template for use. While there is not a specific number, the check looks not only at the number but the distribution across the sample. For example if 50 points are needed but 40 points are found in only half the image this would be rejected, the distribution of the minutia would still cause the sample to fail.

Each of these measures is used independently to determine whether the sample has sufficient quality.

These measures are used on any sample that is collected, regardless of the purpose of the sample (enrollment or verification).

MOD_BIO_V1.1:FIA_MBV_EXT.1/BMFPS:
MOD_BIO_V1.1:FIA_MBV_EXT.1/UDFPS:
The TOE's fingerprint sensor provides FAR and FRR rates as shown in Table 34. Each phone provides a FRR of shown in the table below, along with a rounded up (for the worse) and mapped ratio. Prior to the rounded rate, the FRR meets the requirements for FIA_BMG_EXT in all cases.

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

Device	Sensor Type	False Accept Rate (FAR)	False Reject Rate (FRR)
Pixel 7 Pro/7	UDFPS	1:50,000	2.5%
Pixel 6 Pro/6/6a	UDFPS	1:50,000	2.5%
Pixel 5/5a-5G	BMFPS	1:100,000	1.2%
Pixel 4a-5G/4a	BMFPS	1:100,000	1.2%

Table 34 - Fingerprint False Accept/Reject Rates

MOD_WLANC_V1.0:FIA_PAE_EXT.1:
The TOE can join WPA2-802.1X (802.11i) and WPA3-Enterprise wireless networks requiring EAP-TLS authentication, acting as a client/supplicant (and in that role connect to the 802.11 access point and communicate with the 802.1X authentication server).

PP_MDF_V3.3:FIA_PMG_EXT.1:
The TOE authenticates the user through a password consisting of basic Latin characters (upper and lower case, numbers, and the special characters noted in the selection (see the selections in section 5 for FIA_PMG_EXT.1)). The TOE defaults to requiring passwords to have a minimum of four characters but no more than sixteen, contain at least one letter; however, an MDM application can change these defaults. The Smart Lock feature is not allowed in the evaluated configuration as this feature circumvents the requirements for FIA_PMG_EXT.1 and many others.

PP_MDF_V3.3:FIA_TRT_EXT.1:
Android's GateKeeper throttling is used to prevent brute-force attacks. After a user enters an incorrect password or a failed biometric, GateKeeper APIs return a value in milliseconds (500ms default) in which

the caller must wait before attempting to validate another attempt. Any attempts before the defined amount of time has passed will be ignored by GateKeeper. Gatekeeper also keeps a count of the number of failed validation attempts since the last successful attempt. These two values together are used to prevent brute-force attacks of the TOE.

PP_MDF_V3.3:FIA_UAU.5:
The TOE, in its evaluated configuration, allows the user to authenticate using either a password or biometric (see Table 33). Upon boot, the first unlock screen presented requires the user to enter their password to unlock the device. The biometric sensors are disabled until the user enters their password for the first time.

Upon device lock during normal use of the device, the user has the ability to unlock the phone either by entering their password or by using a biometric authentication. Throttling of these inputs can be read about in the FIA_AFL_EXT.1 section. The entered password is compared to a value derived as described in the key hierarchy and key table above (FCS_STG_EXT.2 and FCS_CKM_EXT.3, respectively). FIA_MBV_EXT.1 describes the biometric authentication process and its security measures.

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

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

PP_MDF_V3.3:FIA_UAU.6/CREDENTIAL:
PP_MDF_V3.3:FIA_UAU.6/LOCKED:
The TOE requires the user to enter their password or supply their biometric in order to unlock the TOE. Additionally the TOE requires the user to confirm their current password when accessing the

```
Settings -> Security -> Screen lock
```

menu in the TOE's user interface. The TOE can disable Smart Lock through management controls. Only after entering their current user password can the user then elect to change their password.

PP_MDF_V3.3:FIA_UAU.7:
The TOE allows the user to enter the user's password from the lock screen. The TOE will, by default, display the most recently entered character of the password briefly or until the user enters the next character in the password, at which point the TOE obscures the character by replacing the character with a dot symbol. Further, the TOE provides no feedback other than whether the fingerprint unlock attempt succeeded or failed.

PP_MDF_V3.3:FIA_UAU_EXT.1:
As described before, the TOE's key hierarchy requires the user's password in order to derive the KEK_* keys in order to decrypt other KEKs and DEKs. Thus, until it has the user's password, the TOE cannot decrypt the DEK utilized for Data-At-Rest encryption, and thus cannot decrypt the user's protected data.

PP_MDF_V3.3:FIA_UAU_EXT.2:
The TOE, when configured to require a user password, allows a user to perform the actions assigned in FIA_UAU_EXT.2.1 (see selections in section 5 for FIA_UAU_EXT.2) without first successfully authenticating. Choosing the input method allows the user to select between different keyboard devices

(say, for example, if the user has installed additional keyboards). Note that the TOE automatically names and saves (to the internal Flash) any screen shots or photos taken from the lock screen, and the TOE provides the user no opportunity to name them or change where they are stored.

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

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

PP_MDF_V3.3:FIA_X509_EXT.1:
MOD_WLANC_V1.0:FIA_X509_EXT.1/WLAN:
The TOE checks the validity of all imported CA certificates by checking for the presence of the basicConstraints extension and that the CA flag is set to TRUE as the TOE imports the certificate. Additionally, the TOE verifies the extendedKeyUsage Server Authentication purpose during WPA2/EAP-TLS negotiation. The TOE's certificate validation algorithm examines each certificate in the path (starting with the peer's certificate) and first checks for validity of that certificate (e.g., has the certificate expired; or if not yet valid, whether the certificate contains the appropriate X.509 extensions [e.g., the CA flag in the basic constraints extension for a CA certificate, or that a server certificate contains the Server Authentication purpose in the extendedKeyUsage field]), then verifies each certificate in the chain (applying the same rules as above, but also ensuring that the Issuer of each certificate matches the Subject in the next rung "up" in the chain and that the chain ends in a self-signed certificate present in either the TOE's trusted anchor database or matches a specified Root CA), and finally the TOE performs revocation checking for all certificates in the chain.

PP_MDF_V3.3:FIA_X509_EXT.2:
MOD_WLANC_V1.0:FIA_X509_EXT.2/WLAN:
MOD_WLANC_V1.0:FIA_X509_EXT.6:
The TOE uses X.509v3 certificates during EAP-TLS, TLS, and HTTPS. The TOE comes with a built-in set of default Trusted Credentials (Android's set of trusted CA certificates), and while the user cannot remove any of the built-in default CA certificates, the user can disable any of those certificates through the user interface so that certificates issued by disabled CA's cannot validate successfully. In addition, a user and an administrator/MDM can import a new trusted CA certificate into the Trust Anchor Database (the TOE stores the new CA certificate in the Security Key Store).

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

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

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

PP_MDF_V3.3:FIA_X509_EXT.3:
Applications needing compliant revocation checking must utilize the NIAPSEC library. The NIAPSEC library created by the vendor provides the following functions to allow for certificate path validation and revocation checking:

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

The first function allows for validation and revocation checking against a list of certificates, while the second checks a singular certificate. Revocation checking is completed using OCSP. Please see the FIA_X509_EXT.2/WLAN section for a further explanation on how the TOE handles revocation checking.

6.5 Security management

PP_MDF_V3.3:FMT_MOF_EXT.1:
PP_MDF_V3.3:FMT_SMF.1:
The TOE provides the management functions described in Table 14 - Security Management Functions in section 5. The table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted. It is worth noting that the TOE's ability to specify authorized application repositories takes the form of allowing enterprise applications (i.e., restricting applications to only those applications installed by an MDM Agent).

MOD_BT_V1.0:FMT_SMF_EXT.1/BT:
The TOE provides the management functions described in Table 15 - Bluetooth Security Management Functions in section 5. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted.

MOD_WLANC_V1.0:FMT_SMF_EXT.1/WLAN:
The TOE provides the management functions described in Table 16 - WLAN Security Management Functions in section 5. As with Table 14 - Security Management Functions, the table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted.

PP_MDF_V3.3:FMT_SMF_EXT.2:
The TOE offers MDM agents the ability to wipe protected data, wipe sensitive data, remove Enterprise applications, and remove all device stored Enterprise resource data upon un-enrollment. The TOE offers

MDM agents the ability to wipe protected data (effectively wiping the device) at any time. Similarly, the TOE also offers the ability to remove Enterprise applications and a full wipe of managed profile data of the TOE's Enterprise data/applications at any time.

PP_MDF_V3.3:FMT_SMF_EXT.3:
The TOE offers MDM agents and the user the

```
Settings -> Security -> Advanced settings -> Device admin apps
```

menu to view each application that has been granted admin rights, and further to see what operations each admin app has been granted.

6.6 Protection of the TSF

PP_MDF_V3.3:FPT_AEX_EXT.1:
The Linux kernel of the TOE's Android operating system provides address space layout randomization utilizing the get_random_int(void) kernel random function to provide eight unpredictable bits to the base address of any user-space memory mapping. The random function, though not cryptographic, ensures that one cannot predict the value of the bits.

PP_MDF_V3.3:FPT_AEX_EXT.2:
The TOE utilizes 5.10, 4.19 and 4.14 Linux kernels (https://source.android.com/devices/architecture/kernel/modular-kernels#core-kernel-requirements), whose memory management unit (MMU) enforces read, write, and execute permissions on all pages of virtual memory and ensures that write and execute permissions are not simultaneously granted on all memory. The Android operating system sets the ARM No eXecute (XN) bit on memory pages and the TOE's ARMv8 Application Processor's Memory Management Unit (MMU) circuitry enforces the XN bits. From Android's documentation (https://source.android.com/devices/tech/security/index.html), Android supports 'Hardware-based No eXecute (NX) to prevent code execution on the stack and heap. Section D.5 of the ARMv8 Architecture Reference Manual contains additional details about the MMU of ARM-based processors: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0487a.f/index.html.

PP_MDF_V3.3:FPT_AEX_EXT.3:
The TOE's Android operating system provides explicit mechanisms to prevent stack buffer overruns in addition to taking advantage of hardware-based No eXecute to prevent code execution on the stack and heap. Specifically, the vendor builds the TOE (Android and support libraries) using gcc-fstack-protector compile option to enable stack overflow protection and Android takes advantage of hardware-based eXecute-Never to make the stack and heap non-executable. The vendor applies these protections to all TSF executable binaries and libraries.

PP_MDF_V3.3:FPT_AEX_EXT.4:
The TOE protects itself from modification by untrusted subjects using a variety of methods. The first protection employed by the TOE is a Secure Boot process that uses cryptographic signatures to ensure the authenticity and integrity of the bootloader and kernels using data fused into the device processor.

The TOE protects its REK by limiting access to only trusted applications within the TEE (Trusted Execution Environment). The TOE key manager includes a TEE module that utilizes the REK to protect all other keys in the key hierarchy. All TEE applications are cryptographically signed, and when invoked at runtime (at

the behest of an untrusted application), the TEE will only load the trusted application after successfully verifying its cryptographic signature.

The TOE protects biometric data by separating it from the Android operating system. The biometric sensor is tied to the TEE such that it cannot be accessed directly from Android but can only be done through the biometric software inside the TEE. All biometric data is maintained within the TEE such that Android is only able to know the result of a biometric process (such as enrollment or verification), and not any of the data used in that process itself.

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

While the TOE supports USSD and MMI codes, they are only available once the user has authenticated to the TOE through the dialer. Attempting to access these codes through the emergency dialer will be rejected as a non-emergency number.

The TOE, in its evaluated configuration has its bootloader in the locked state. This prevents a user from installing a new software image via another method than Google's proscribed OTA methods. The TOE allows an operator to download and install an OTA update through the system settings

```
Settings -> System -> System update -> Check for update
```

while the phone is running, or by separately downloading an OTA image, and then "sideloading" the OTA update from Android's recovery mode. In both cases, the TOE will verify the digital signature of the new OTA before applying the new firmware.

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

PP_MDF_V3.3:FPT_AEX_EXT.5:
The TOE models provide Kernel Address Space Layout Randomization (KASLR) as a hardening feature to randomize the location of kernel data structures at each boot, including the core kernel as a random physical address, mapping the core kernel at a random virtual address in the vmalloc area, loading kernel modules at a random virtual address in the vmalloc area, and mapping system memory at a random virtual address in the linear area. The entropy used to dictate the randomization is based on the hardware present within the phone. For ARM devices, such as the TOE, 13-25 bits of entropy are generated on boot from the DRBG in the Application Processor, from which the starting memory address is generated.

PP_MDF_V3.3:FPT_BBD_EXT.1:
The TOE's hardware and software architecture ensures separation of the application processor (AP) from the baseband or communications processor (CP) through internal controls of the TOE's SoC, which contains both the AP and the CP. The AP restricts hardware access control through a protection unit that

restricts software access from the baseband processor through a dedicated 'modem interface'. The protection unit combines the functionality of the Memory Protection Unit (MPU), the Register Protection Unit (RPU), and the Address Protection Unit (APU) into a single function that conditionally grants access by a master to a software defined area of memory, to registers, or to a pre-decoded address region, respectively. The modem interface provides a set of APIs (grouped into five categories) to enable a high-level OS to send messages to a service defined on the modem/baseband processor. The combination of hardware and software restrictions ensures that the TOE's AP prevents software executing on the modem or baseband processor from accessing the resources of the application processor (outside of the defined methods, mediated by the application processor).

MOD_BIO_V1.1:FPT_BDP_EXT.1:
The complete biometric authentication process happens inside the TEE (including image capture, all processing and match determination). All software in the biometric system is inside the TEE boundary, while the sensors are accessible from within Android. The TEE handles calls for authentication made from Android with only the success or failure of the match provided back to Android (and when applicable, to the calling app). The image taken by the capture sensor is processed by the biometric service to check the enrolled templates for a match to the captured image.

PP_MDF_V3.3:FPT_JTA_EXT.1:
The TOE prevents access to its processor's JTAG interface by requiring use of a signing key to authenticate prior to gaining JTAG access. Only a JTAG image with the accompanying device serial number (which is different for each mobile device) that has been signed by Google's private key can be used to access a device's JTAG interface. The Google private key corresponds to the Google RSA 2048-bit public key (a SHA-256 hash of which is fused into the TOE's application processor).

PP_MDF_V3.3 & MOD_BIO_V1.1:FPT_KST_EXT.1:
The TOE does not store any plaintext key or biometrics material in its internal Flash; the TOE encrypts all keys and biometric data before storing them. This ensures that irrespective of how the TOE powers down (e.g., a user commands the TOE to power down, the TOE reboots itself, or battery depletes or is removed), all keys and biometric data stored in the internal Flash are wrapped with a KEK. Please refer to section 6.2 of the TSS for further information (including the KEK used) regarding the encryption of keys stored in the internal Flash. As the TOE encrypts all keys stored in Flash, upon boot-up, the TOE must first decrypt any keys in order to utilize them.

PP_MDF_V3.3 & MOD_BIO_V1.1:FPT_KST_EXT.2:
The TOE itself (i.e., the mobile device) comprises a cryptographic module that utilizes cryptographic libraries including BoringSSL, application processor cryptography (which leverages AP hardware), and the following system-level executables that utilize KEKs: vold, wpa_supplicant, and the Android Key Store.

1. vold and application processor hardware provides Data-At-Rest encryption of the user data partition in Flash
    a. Qualcomm Inline Cryptographic Engine (ICE) on Snapdragon processors
    b. Google Tensor Inline Storage Encryption (ISE) on Tensor processors
2. wpa_supplicant provides WPA2/WPA3 services
3. the Android Key Store application provides key generation, storage, deletion services to mobile applications and to user through the UI

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

The TOE also ensures that biometric data used for enrolling and authenticating users can not be exported. During authentication or enrollment, the calling program (the TSF or an app) is able to request biometric actions, but the data resulting from that action is not provided back to the calling program. The calling program only receives a notice of success of failure about the process.

PP_MDF_V3.3:FPT_KST_EXT.3:
The TOE does not provide any way to export plaintext DEKs or KEKs (including all keys stored in the Android Key Store) as the TOE chains or directly encrypts all KEKs to the REK.

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

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

PP_MDF_V3.3:FPT_NOT_EXT.1:
When the TOE encounters a critical failure (either a self-test failure or TOE software integrity verification failure), a failure is message is displayed to the screen, and the TOE attempts to reboot. If the failure persists between boots, the user may attempt to boot to the recovery mode/kernel to wipe data and perform a factory reset in order to recover the device.

MOD_BIO_V1.1:FPT_PBT_EXT.1:
The TOE requires the user to enter their password to enroll, re-enroll or unenroll any biometric templates. When the user attempts biometric authentication to the TOE, the biometric sensor takes an

image of the presented biometric for comparison to the enrolled templates. The biometric system compares the captured image to all the stored templates on the device to determine if there is a match.

PP_MDF_V3.3:FPT_STM.1:
The TOE requires time for the Package Manager (which installs and verifies APK signatures and certificates), image verifier, wpa_supplicant, and Android Key Store applications. These TOE components obtain time from the TOE using system API calls [e.g., time() or gettimeofday()]. An application (unless a system application is residing in /system/priv-app or signed by the vendor) cannot modify the system time as mobile applications need the Android 'SET_TIME' permission to do so. Likewise, only a process with root privileges can directly modify the system time using system-level APIs. Further, this stored time is used both for the time/date tags in audit logs and is used to track inactivity timeouts that force the TOE into a locked state.

By default, the TOE uses the Cellular Carrier time (obtained through the Carrier's network time server) as the trusted time source. The admin can decide to not use cellular time as the trusted source but instead use a NTP server to set the trusted time. The default NTP server is a Google-hosted server source, but this can be changed by the admin to point to another trusted server. It is also possible to let the user set the date and time through the TOE's user interface and use the internal clock to maintain a local (as opposed to externally checked) trusted time.

PP_MDF_V3.3:FPT_TST_EXT.1:
The TOE automatically performs known answer power on self-tests (POST) on its cryptographic algorithms to ensure that they are functioning correctly. Each component providing cryptography (application processor, and BoringSSL) performs known answer tests on their cryptographic algorithms to ensure they are working correctly. Should any of the tests fail, the TOE displays an error message stating "Boot Failure" and halts the boot process, displays an error to the screen, and forces a reboot of the device.

Algorithm	Implemented in	Description
AES encryption/decryption	BoringSSL	Comparison of known answer to calculated value
ECDH key agreement	BoringSSL	Comparison of known answer to calculated value
DRBG random bit generation	BoringSSL	Comparison of known answer to calculated value
ECDSA sign/verify	BoringSSL	Comparison of known answer to calculated value
HMAC-SHA	BoringSSL	Comparison of known answer to calculated value
RSA sign/verify	BoringSSL	Comparison of known answer to calculated value
SHA hashing	BoringSSL	Comparison of known answer to calculated value
AES encryption/decryption	Application Processor	Comparison of known answer to calculated value
HMAC-SHA	Application Processor	Comparison of known answer to calculated value
DRBG random bit generation	Application Processor	Comparison of known answer to calculated value
SHA hashing	Application Processor	Comparison of known answer to calculated value
AES-XTS encrypt/decrypt	Application Processor	Comparison of known answer to calculated value

Table 35 - Power-up Cryptographic Algorithm Known Answer Tests

PP_MDF_V3.3:FPT_TST_EXT.2/PREKERNEL:
PP_MDF_V3.3:FPT_TST_EXT.2/POSTKERNEL:
MOD_WLANC_V1.0:FPT_TST_EXT.3/WLAN:
The TOE ensures a secure boot process in which the TOE verifies the digital signature of the bootloader software for the Application Processor (using a public key whose hash resides in the processor's internal

fuses) before transferring control. The bootloader, in turn, verifies the signature of the Linux kernel it loads. This series of checks occur for all boot modes (normal, recovery and fastboot). The recovery and fastboot modes utilize the same alternative boot mode but expose different software to the user once the boot is complete.

For any boot mode, the TOE performs checking of the entire /system and /vendor partitions through use of Android's dm-verity mechanism (and while the TOE will still operate, it will log any blocks/executables that have been modified). Some limited failures (changes under a block size, depending on the location of the failure) can be automatically self-corrected as part of the check process.

dm-verity is a hash table of the block device used for storage (in this case the /system and /vendor partitions) where every 4k block has a SHA256. These hashes are then concatenated and every 4k of that hash is again hashed. This is repeated until a 4k root hash is generated (this is normally 4 total layers of hashes), and this root hash is signed with the keys used to verify the signature of the Linux kernel. The Wi-Fi components are included in the /system partition and are verified as part of the dm-verity check of that partition as part of the platform checks.

PP_MDF_V3.3:FPT_TUD_EXT.1:
The TOE's user interface provides a method to query the current version of the TOE software/firmware (Android version, baseband version, kernel version, build number, and software version) and hardware (model and version). Additionally, the TOE provides users the ability to review the currently installed apps (including 3rd party 'built-in' applications) and their version.

PP_MDF_V3.3:FPT_TUD_EXT.2:
The TOE verifies all OTA (over-the-air) updates to the TOE software (which includes baseband processor updates) using a public key chaining ultimately to the Root Public Key, a hardware protected key whose SHA-256 hash resides inside the application processor. Should this verification fail, the software update will fail and the update will not be installed.

The application processor verifies the bootloader's authenticity and integrity (thus tying the bootloader and subsequent stages to a hardware root of trust: the SHA-256 hash of the Root Public Key, which cannot be reprogrammed after the "write-enable" fuse has been blown).

PP_MDF_V3.3:FPT_TUD_EXT.3:
The Android OS on the TOE requires that all applications bear a valid signature before Android will install the application.

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

PP_MDF_V3.3:FPT_TUD_EXT.6:
The TOE maintains a monotonic anti-rollback counter used to set a minimum version for the TOE software. Before a new update can be installed, the version of the new software is compared to the counter version. The update is allowed only if the version of the new software is equal or greater than the counter.

6.7 TOE access

PP_MDF_V3.3:FTA_SSL_EXT.1:
The TOE transitions to its locked state either immediately after a User initiates a lock by pressing the power button (if configured) or after a (also configurable) period of inactivity, and as part of that transition, the TOE will display a lock screen (the KeyGuard lock screen) to obscure the previous contents and play a "lock sound" to indicate the phone's transition; however, the TOE's lock screen still displays email notifications, calendar appointments, user configured widgets, text message notifications, the time, date, call notifications, battery life, signal strength, and carrier network. But without authenticating first, a user cannot perform any related actions based upon these notifications (they cannot respond to emails, calendar appointments, or text messages) other than the actions assigned in Timing of Authentication (PP_MDF_V3.3:FIA_UAU_EXT.2).

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

Note that during power up, the TOE presents the user with an unlock screen stating "unlock for all features and data". While at this screen, the TOE has already decrypted Device Encrypted (DE) files within the userdata partition, but cannot yet decrypt the user's Credential Encrypted (CE) files. The user can only access a subset of device functionality before authenticating (e.g. the user can making an emergency call, receive incoming calls, receiving alarms, and any other "direct boot" functionality). After the user enters their password, the TOE decrypts the user's CE files within the user data partition and the user has unlocked the full functionality of the phone. After this initial authentication, upon (re)locking the phone, the TOE presents the user with the previously mentioned KeyGuard lock screen. While locked, the actions described in FIA_UAU_EXT.2.1 are available for the user to utilize.

PP_MDF_V3.3:FTA_TAB.1:
The TOE can be configured to display a user-specified message on the Lock screen, and additionally an administrator can also set a Lock screen message using an MDM.

MOD_WLANC_V1.0:FTA_WSE_EXT.1:
The TOE allows an administrator to specify (through the use of an MDM) a list of wireless networks (SSIDs) to which the user may direct the TOE to connect to, the security type, authentication protocol, and the client credentials to be used for authentication. When not enrolled with an MDM, the TOE allows the user to control to which wireless networks the TOE should connect, but does not provide an explicit list of such networks, rather the user may scan for available wireless network (or directly enter a specific wireless network), and then connect. Once a user has connected to a wireless network, the TOE will automatically reconnect to that network when in range and the user has enabled the TOE's Wi-Fi radio.

6.8 Trusted path/channels

MOD_BT_V1.0:FTP_BLT_EXT.1:
MOD_BT_V1.0:FTP_BLT_EXT.3/BR:
MOD_BT_V1.0:FTP_BLT_EXT.3/LE:
The TSF enforces the use of encryption by default, over Bluetooth BD/EDR and LE connections using at least 128-bit AES encryption keys and does not allow the key length to be renegotiated below the length set at the pairing (the request to change the size will be rejected, and the connection terminated if this

is not accepted). ECDH is used to generate key pairs for the devices to exchange symmetric keys. The admin cannot configure key sizes.

MOD_BT_V1.0:FTP_BLT_EXT.2:
The TSF will terminate a connection with a remote device if the remote device requests to terminate encryption.

PP_MDF_V3.3:FTP_ITC_EXT.1:
The TOE provides secured (encrypted and mutually authenticated) communication channels between itself and other trusted IT products through the use of TLS and HTTPS. The TOE provides mobile applications and MDM agent applications access to HTTPS and TLS via published APIs, thus facilitating administrative communication and configured enterprise connections. These APIs are accessible to any application that needs an encrypted end-to-end trusted channel.

The TOE also uses TLS connections to download OTA updates for the device.

MOD_WLANC_V1.0:FTP_ITC_EXT.1/WLAN:
The TOE provides secured (encrypted and mutually authenticated) communication channels between itself and other trusted IT products through the use of IEEE 802.11-2012, 802.1X, and EAP-TLS. The TOE permits itself and applications to initiate communicate via the trusted channel, and the TOE initiates communications via the WPA2/WPA3 (IEEE 802.11-2012, 802.1X with EAP-TLS) trusted channel for connection to a wireless access point.

6.9 Live-cycle support

PP_MDF_V3.3:ALC_TSU_EXT.1:
Google supports a bug filing system for the Android OS outlined here: https://source.android.com/setup/contribute/report-bugs. This allows developers or users to search for, file, and vote on bugs that need to be fixed. This helps to ensure that all bugs that affect large numbers of people get pushed up in priority to be fixed. The method outlined above requires the user to submit their bug to Android's website. As such, the user of the device needs to establish a trusted channel web connection to securely file the bug by following the set-up steps to establish a secure HTTPS/TLS/EAP-TLS connection from the TOE, then visiting the above web portal to submit the report.

Google also commits to pushing out monthly security updates for the Android operating system (including the Java layer and kernel, not including applications). Google provides security updates for at least three years from the device launch. The latest information about this can be found at https://support.google.com/nexus/answer/4457705?hl=en#zippy=%2Cpixel-xl-a-a-g-a-g%2Cpixel-later (summarized in

Device	Android updates to	Security patched to
Pixel 7 Pro/7	Oct 2025	Oct 2027
Pixel 6 Pro/6	Oct 2024	Oct 2026
Pixel 6a	Jul 2025	Jul 2027
Pixel 5a-5G	Aug 2024	Aug 2024
Pixel 5	Oct 2023	Oct 2023
Pixel 4a-5G	Nov 2023	Nov 2023
Pixel 4a	Aug 2023	Aug 2023

Table 36 - Security Update Period

These systematic updates are designed to address the highest issue problems as quickly as possible and allows Google to ensure their Pixel products remain as safe as possible and any issues are addressed promptly. Google posts Android Security Bulletins with each release showing the patches that are included https://source.android.com/docs/security/bulletin.

Google creates updates and patches to resolve reported issues as quickly as possible. The delivery time for resolving an issue depends on the severity, and can be as rapid as a few days before the update can be deployed for high priority cases. Google maintains a security blog (https://android-developers.googleblog.com/) to disseminate information directly to the public.