

# Google Pixel Phones on Android 16 Administrator Guidance Documentation

Version 1.0  
May 4, 2026

|           |                                                     |           |
|-----------|-----------------------------------------------------|-----------|
| <b>1.</b> | <b>DOCUMENT INTRODUCTION.....</b>                   | <b>4</b>  |
| 1.1.      | EVALUATED DEVICES.....                              | 4         |
| 1.2.      | ACRONYMS.....                                       | 5         |
| <b>2.</b> | <b>EVALUATED CAPABILITIES.....</b>                  | <b>6</b>  |
| 2.1.      | DATA PROTECTION.....                                | 6         |
| 2.1.1.    | <i>File-Based Encryption</i> .....                  | 6         |
| 2.2.      | LOCK SCREEN.....                                    | 7         |
| 2.2.1.    | <i>Biometric Authentication – General</i> .....     | 7         |
| 2.2.2.    | <i>Biometric Authentication - Fingerprint</i> ..... | 7         |
| 2.2.3.    | <i>Biometric Authentication – Face</i> .....        | 8         |
| 2.2.4.    | <i>Authentication Rate Limiting</i> .....           | 8         |
| 2.3.      | KEY MANAGEMENT.....                                 | 9         |
| 2.3.1.    | <i>KeyStore</i> .....                               | 9         |
| 2.3.2.    | <i>KeyChain</i> .....                               | 9         |
| 2.4.      | DEVICE INTEGRITY.....                               | 10        |
| 2.4.1.    | <i>Verified Boot</i> .....                          | 10        |
| 2.5.      | DEVICE MANAGEMENT.....                              | 10        |
| 2.5.1.    | <i>EMM/MDM console</i> .....                        | 11        |
| 2.5.2.    | <i>DPC (MDM Agent)</i> .....                        | 11        |
| 2.6.      | WORK PROFILE SEPARATION.....                        | 11        |
| 2.7.      | VPN CONNECTIVITY.....                               | 12        |
| 2.8.      | AUDIT LOGGING.....                                  | 12        |
| <b>3.</b> | <b>SECURITY CONFIGURATION.....</b>                  | <b>13</b> |
| 3.1.      | MANAGEMENT ENROLLMENT.....                          | 13        |
| 3.1.1.    | <i>Android Device Policy</i> .....                  | 13        |
| 3.2.      | COMMON CRITERIA MODE.....                           | 13        |
| 3.3.      | CRYPTOGRAPHIC MODULE IDENTIFICATION.....            | 14        |
| 3.4.      | PERMISSIONS MODEL.....                              | 15        |
| 3.5.      | COMMON CRITERIA RELATED SETTINGS.....               | 16        |
| 3.6.      | PASSWORD RECOMMENDATIONS.....                       | 25        |
| 3.7.      | BUG REPORTING PROCESS.....                          | 25        |
| <b>4.</b> | <b>BLUETOOTH CONFIGURATION.....</b>                 | <b>26</b> |
| <b>5.</b> | <b>WI-FI CONFIGURATION.....</b>                     | <b>28</b> |
| <b>6.</b> | <b>VPN CONFIGURATION.....</b>                       | <b>29</b> |
| <b>7.</b> | <b>WORK PROFILE SEPARATION.....</b>                 | <b>30</b> |
| <b>8.</b> | <b>SECURE UPDATE PROCESS.....</b>                   | <b>31</b> |
| 8.1.      | GOOGLE PLAY SYSTEM UPDATES.....                     | 31        |
| <b>9.</b> | <b>AUDIT LOGGING.....</b>                           | <b>33</b> |

|                                                                                                                         |           |
|-------------------------------------------------------------------------------------------------------------------------|-----------|
| <b>10. FDP_DAR_EXT.2 &amp; FCS_CKM.2(2) – SENSITIVE DATA PROTECTION OVERVIEW .....</b>                                  | <b>41</b> |
| 10.1. ARCHITECTURE OVERVIEW .....                                                                                       | 41        |
| 10.2. DYNAMIC STATE TRANSITION AND RE-WRAPPING (FDP_DAR_EXT.2.4).....                                                   | 41        |
| 10.3. IMPLEMENTATION GUIDE AND SAMPLE CODE.....                                                                         | 42        |
| <b>11. API SPECIFICATION.....</b>                                                                                       | <b>44</b> |
| 11.1. CRYPTOGRAPHIC APIs.....                                                                                           | 44        |
| 11.1.1. <i>SecureCipher</i> .....                                                                                       | 45        |
| 11.1.2. <i>FCS_CKM.2/UNLOCKED – Key Establishment (RSA)</i> .....                                                       | 47        |
| 11.1.3. <i>FCS_CKM.2/UNLOCKED – Key Establishment (ECDSA) &amp; FCS_COP.1/SIGN – Signature Algorithms (ECDSA)</i> ..... | 47        |
| 11.1.4. <i>FCS_CKM.1 – Key Generation (ECDSA)</i> .....                                                                 | 48        |
| 11.1.5. <i>FCS_COP.1/ENCRYPT – Encryption/Decryption (AES)</i> .....                                                    | 48        |
| 11.1.6. <i>FCS_COP.1/HASH – Hashing (SHA)</i> .....                                                                     | 49        |
| 11.1.7. <i>FCS_COP.1/SIGN – RSA (Signature Algorithms)</i> .....                                                        | 49        |
| 11.1.8. <i>FCS_CKM.1 –Key Generation (RSA)</i> .....                                                                    | 50        |
| 11.1.9. <i>FCS_COP.1/KEYHMAC - HMAC</i> .....                                                                           | 50        |
| 11.2. KEY MANAGEMENT .....                                                                                              | 51        |
| 11.2.1. <i>SecureKeyGenerator</i> .....                                                                                 | 51        |
| 11.3. FCS_TLSC_EXT.1 - CERTIFICATE VALIDATION, TLS, HTTPS .....                                                         | 52        |
| 11.3.1. <i>Cipher Suites</i> .....                                                                                      | 54        |
| 11.3.2. <i>Guidance for Bluetooth Low Energy APIs</i> .....                                                             | 54        |

## 1. Document Introduction

This guide includes procedures for configuring Pixel phones running Android 16 into a Common Criteria evaluated configuration and additionally includes guidance to application developers wishing to write applications that leverage the Pixel phone's Common Criteria compliant APIs and features.

### 1.1. Evaluated Devices

The evaluated devices include the following models and versions:

| Google Product    | Model #                    | Kernel | Android OS Version | Security Patch Level |
|-------------------|----------------------------|--------|--------------------|----------------------|
| Pixel 10 Pro XL   | G45RY, GUL82, GYPW4        | 6.6    | Android 16         | December 2025        |
| Pixel 10 Pro      | GEHN3, G4QUR, GN4F5        | 6.6    | Android 16         | December 2025        |
| Pixel 10          | GK2MP, GL066, GLBW0        | 6.6    | Android 16         | December 2025        |
| Pixel 10 Pro Fold | GU0NP, GM66V               | 6.6    | Android 16         | December 2025        |
| Pixel 10a         | G4H7L, GV0BP, GE1GQ        | 6.6    | Android 16         | December 2025        |
| Pixel 9 Pro XL    | GZC4K, GQ57S, GGX8B        | 6.1    | Android 16         | December 2025        |
| Pixel 9 Pro       | GEC77, GWVK6, GR83Y        | 6.1    | Android 16         | December 2025        |
| Pixel 9           | GUR25, G1B60, G2YBB        | 6.1    | Android 16         | December 2025        |
| Pixel 9 Pro Fold  | GGH2X, GC15S               | 6.1    | Android 16         | December 2025        |
| Pixel 9a          | GXQ96, GTF7P, G3Y12        | 6.1    | Android 16         | December 2025        |
| Pixel 8 Pro       | G1NMW, GC3VE               | 6.1    | Android 16         | December 2025        |
| Pixel 8           | GKWS6, G9BQD               | 6.1    | Android 16         | December 2025        |
| Pixel 8a          | G5760D, G6GPR, G8HHN       | 6.1    | Android 16         | December 2025        |
| Pixel Tablet      | GTU8P                      | 6.1    | Android 16         | December 2025        |
| Pixel Fold        | G9FPL, G0B96               | 6.1    | Android 16         | December 2025        |
| Pixel 7 Pro       | GVU6C, G03Z5, GQML3        | 6.1    | Android 16         | December 2025        |
| Pixel 7           | GE2AE, GFE4J, GP4BC        | 6.1    | Android 16         | December 2025        |
| Pixel 7a          | GWKK3, GHL1X, G82U8, GODZQ | 6.1    | Android 16         | December 2025        |
| Pixel 6 Pro       | GF5KQ, G8V0U, GLU0G        | 6.1    | Android 16         | December 2025        |
| Pixel 6           | GR1YH, GB7N6, G9S9B        | 6.1    | Android 16         | December 2025        |
| Pixel 6a          | GX7AS, GB62Z, G1AZG, GB17L | 6.1    | Android 16         | December 2025        |

To verify the Device, OS Version and Security Patch Level on your device:

1. Tap on Settings
2. Tap on About phone
  - a. Scroll down to Android version and tap on it
  - b. Scroll down to Model and tap on it

The Device Policy application, when using the built-in DPC (MDM Agent) is version 144.

To verify the Device Policy version on your device:

1. Tap on Settings
2. Tap on Apps

3. Tap on See all apps
  - a. Scroll down to Device Policy and tap on it
  - b. Scroll down to the bottom of the page where the version is specified

## 1.2. Acronyms

- AE – Android Enterprise
- AES – Advanced Encryption Standard
- API – Application Programming Interface
- BYOD – Bring Your Own Device
- CA – Certificate Authority
- DO – Device Owner
- DPC – Device Policy Controller
- EMM – Enterprise Mobility Management
- MDM – Mobile Device Management
- PKI – Public Key Infrastructure
- TOE – Target of Evaluation

## 2. Evaluated Capabilities

The Common Criteria configuration adds support for many security capabilities. Some of those capabilities include the following:

- Data Protection
- Lock Screen
- Key Management
- Device Integrity
- Device Management
- Work Profile Separation
- VPN Connectivity
- Audit Logging

### 2.1. Data Protection

Android uses industry-leading security features to protect user data. The platform creates an application environment that protects the confidentiality, integrity, and availability of user data.

#### 2.1.1. File-Based Encryption

Encryption is the process of encoding user data on an Android device using an encryption key. With encryption, even if an unauthorized party tries to access the data, they won't be able to read it. The device utilizes File-based encryption (FBE) which allows different files to be encrypted with different keys that can be unlocked independently.

[Direct Boot](#) allows encrypted devices to boot straight to the lock screen and allows alarms to operate, accessibility services to be available and phones to receive calls before a user has provided their credential.

With file-based encryption and APIs to make apps aware of encryption, it's possible for these apps to operate within a limited context before users have provided their credentials while still protecting private user information.

On a file-based encryption-enabled device, each device user has two storage locations available to apps:

1. Credential Encrypted (CE) storage, which is the default storage location and only available after the user has unlocked the device. CE keys are derived from a combination of user credentials and a hardware secret. It is available after the user has successfully unlocked the device the first time after boot and remains available for active users until the device shuts down, regardless of whether the screen is subsequently locked or not.
2. Device Encrypted (DE) storage, which is a storage location available both before the user has unlocked the device (Direct Boot) and after the user has unlocked the device. DE keys are derived from a hardware secret that's only available after the device has performed a successful Verified Boot.

By default, apps do not run during Direct Boot mode. If an app needs to take action during Direct Boot mode, such as an accessibility service like Talkback or an alarm clock app, the app can register components to run during this mode.

DE and CE keys are unique and distinct - no user's CE or DE key will match another. File-based encryption allows files to be encrypted with different keys, which can be unlocked independently. All encryption is based on AES-256 in XTS mode. Due to the way XTS is defined, it needs two 256-bit keys. In effect, both CE and DE keys are 512-bit keys.

By taking advantage of CE, file-based encryption ensures that a user cannot decrypt another user's data. This is an improvement on full-disk encryption where there's only one encryption key, so all users must know the primary user's passcode to decrypt data. Once decrypted, all data is decrypted.

---

## 2.2. Lock screen

---

---

### 2.2.1. Biometric Authentication – General

---

Biometric authentication is available on all devices. The specific approved (tested) methods varies. Fingerprints can be used on all devices, Face can be used on Pixel 10 and Pixel 10 Fold devices. While the position of the sensor varies by device (such as the different fingerprint sensor placements vs the front-facing camera), the user interactions with the system are otherwise identical. To enroll any biometric, the user must first set a password for the device.

The biometric cannot be used after a power event (such as a power-on or a reboot), and so the password must be configured to act as the primary authentication method. Once the password has been entered once, the user will be able to authenticate using the biometric. Periodically (at least once per day) the user will be required to enter the password.

For supported devices, the user can enroll both fingerprint and face biometrics and use them independently. In these cases, after the initial password authentication, either the fingerprint or face biometric can be used.

---

### 2.2.2. Biometric Authentication - Fingerprint

---

Up to four separate fingerprints can be enrolled into the device at one time. This provides both the ability to use different hands as well as options for things like cuts to a finger that could prevent a successful match.

The user can manage their fingerprints by going to Settings>Security & privacy>Device unlock and tapping on the Fingerprint. The user will be prompted to enter their password and then will be able to manage their fingerprints (fingerprints cannot be managed without entering the password). Tapping Fingerprint will let the user add new fingerprints or delete existing ones from the device. Once the maximum number is added the Add fingerprint option will be unavailable.

When enrolling a fingerprint, the device will prompt the user to provide good samples to build the template. The user will be asked to move the part of the finger touching the sensor around (to cover a wider area). If the user moves the finger too far (such as off the sensor) or not enough, the user will be prompted to ensure a proper range of locations of the finger as well as enough quality

samples. Different messages will guide the user to provide the proper samples and will continue until the device confirms enough quality data has been acquired.

When using an enrolled fingerprint for authentication verification, the device will prompt the user if there is a failure in using the fingerprint during authentication. Failures can be due to any number of issues, from incorrect placement on the sensor (such as only covering half the sensor), dirt or other obstructions (cuts, water, etc) or just a poor capture (such as not leaving the finger on the sensor long enough). Some minimal guidance may be provided where the attempt looks more accidental (such as the finger not being left on the sensor long enough to read the print), while otherwise it is just presented as a failure, regardless of the failure determination. A dirty finger, and incorrect finger (one not registered) or a different attempt (such as a spoof attempt) will just provide a warning of a failure, not a detail of what the failure is. This prevents leaking too much information about the attempt in case of attacks.

---

### 2.2.3. Biometric Authentication – Face

---

Only a single face can be enrolled into the device at one time.

The user can manage their face template by going to Settings>Security & privacy>Device unlock and tapping on the Face. The user will be prompted to enter their password and then will be able to manage their face (the face template and associated settings cannot be managed without entering the password). Tapping Face will let the user delete the existing face template or add one to the device.

When enrolling a face, the device will prompt the user to provide good samples to build the template. The user will be prompted for several different pictures around their face, specifying the positioning of the device in relation to the user while providing a silhouette for centering the user's face to ensure a good capture. The different messages will guide the user to provide the proper samples (images) and will continue until the device confirms enough quality data has been acquired.

When using an enrolled face for authentication verification, the device will prompt the user if there is a failure in using the face during authentication. Failures can be due to any number of issues, from not looking at the camera at an appropriate angle to poor lighting conditions. Some minimal guidance may be provided where the attempt looks more accidental (such as raising the device while turned away from it), while otherwise it is just presented as a failure, regardless of the failure determination. Many failures will just provide a warning of a failure, not a detail of what the failure is. This prevents leaking too much information about the attempt in case of attacks.

---

### 2.2.4. Authentication Rate Limiting

---

Both biometric template matching and passcode verification can only take place on secure hardware with rate limiting (exponentially increasing timeouts) enforced. Android's GateKeeper throttling is also used to prevent brute-force attacks. After a user enters an incorrect password, GateKeeper APIs return a value in milliseconds in which the caller must wait before attempting to validate another password. 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's password.

For biometric fingerprint authentication, the user can attempt 5 failed fingerprint unlocks before fingerprint is locked. The device prohibits use of fingerprint until the password is entered.

Android offers [APIs](#) that allow apps to use biometrics for authentication, and allows users to authenticate by using their fingerprint scans on supported devices. These APIs are used in conjunction with the [Android Keystore system](#).

---

## 2.3. Key Management

---

---

### 2.3.1. KeyStore

---

The Android [KeyStore](#) class lets you manage private keys in secure hardware to make them more difficult to extract from the device. The KeyStore enables apps to generate and store credentials used for authentication, encryption, or signing purposes.

Keystore supports [symmetric cryptographic primitives](#) such as AES (Advanced Encryption Standard) and HMAC (Keyed-Hash Message Authentication Code) and asymmetric cryptographic algorithms such as RSA and EC. Access controls are specified during key generation and enforced for the lifetime of the key. Keys can be restricted to be usable only after the user has authenticated, and only for specified purposes or with specified cryptographic parameters. For more information, see the [Authorization Tags](#) and [Functions](#) pages.

Additionally, [version binding](#) binds keys to an operating system and patch level version. This ensures that an attacker who discovers a weakness in an old version of system or TEE software cannot roll a device back to the vulnerable version and use keys created with the newer version.

On Pixel phones, the KeyStore is implemented in secure hardware. This guarantees that even in the event of a kernel compromise, KeyStore keys are not extractable from the secure hardware. Pixel devices also include StrongBox Keymaster, an implementation of the Keymaster HAL that resides in a Titan M. This module contains its own CPU, secure storage, a true random-number generator and additional mechanisms to resist package tampering and unauthorized sideloading of apps. When checking keys stored in the StrongBox Keymaster, the system corroborates a key's integrity with the Trusted Execution Environment (TEE).

---

### 2.3.2. KeyChain

---

The [KeyChain](#) class allows apps to use the system credential storage for private keys and certificate chains. KeyChain is often used by Chrome, Virtual Private Network (VPN) apps, and many enterprise apps to access keys imported by the user or by the mobile device management app.

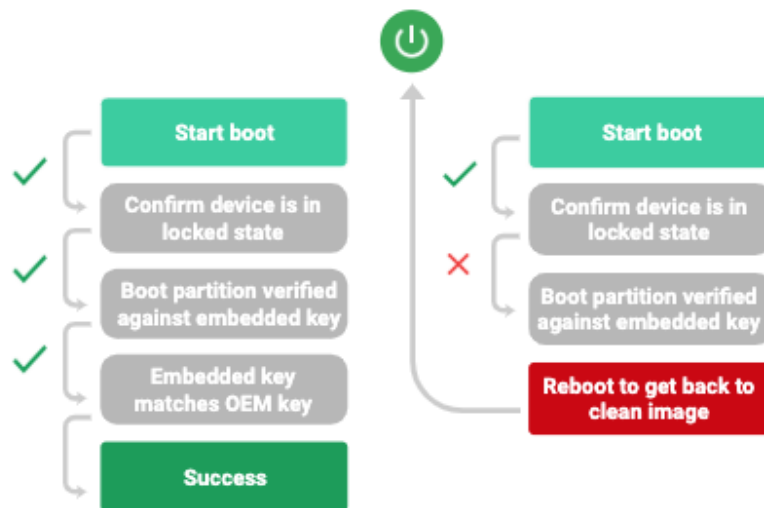
Whereas the KeyStore is for non-shareable app-specific keys, KeyChain is for keys that are meant to be shared across profiles. For example, your mobile device management agent can import a key that Chrome will use for an enterprise website.

## 2.4. Device Integrity

Device integrity features protect the mobile device from running a tampered operating system. With companies using mobile devices for essential communication and core productivity tasks, keeping the OS secure is essential. Without device integrity, very few security properties can be assured. Android adopts several measures to guarantee device integrity at all times.

### 2.4.1. Verified Boot

Verified Boot is Android's secure boot process that verifies system software before running it. This makes it more difficult for software attacks to persist across reboots, and provides users with a safe state at boot time. Each Verified Boot stage is cryptographically signed. Each phase of the boot process verifies the integrity of the subsequent phase, prior to executing that code. Full boot of a compatible device with a locked bootloader proceeds only if the OS satisfies integrity checks. Verification algorithms used must be as strong as current recommendations from NIST for hashing algorithms (SHA-256) and public key sizes (RSA-2048).

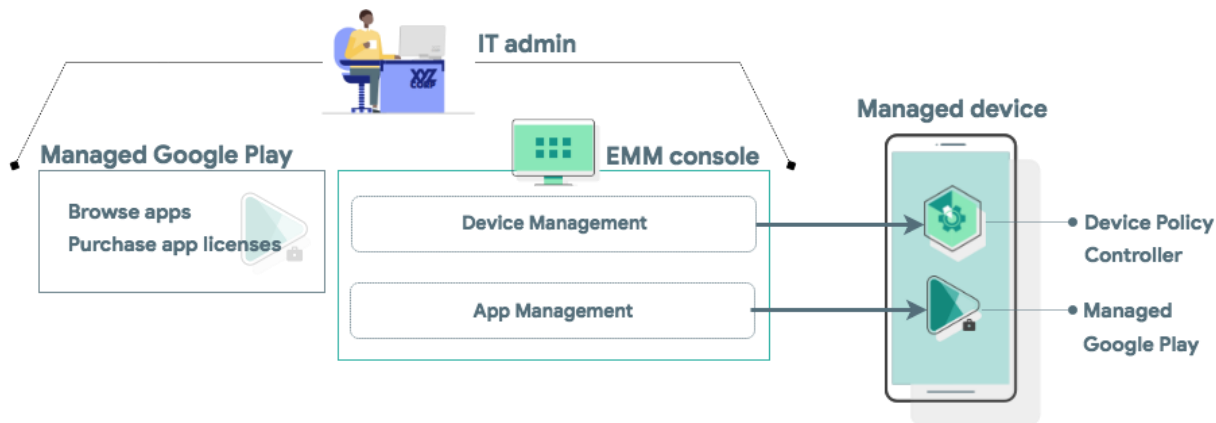


The Verified Boot state is used as an input in the process to derive disk encryption keys. If the Verified Boot state changes (e.g. the user unlocks the bootloader), then the secure hardware prevents access to data used to derive the disk encryption keys that were used when the bootloader was locked.

Find out more about Verified Boot [here](#).

## 2.5. Device Management

The TOE leverages the device management capabilities that are provided through Android Enterprise which is a combination of three components: your EMM/MDM console, a device policy controller (DPC) which is your MDM Agent, and a EMM/MDM Application Catalog.



Components of an Android Enterprise solution.

### 2.5.1. EMM/MDM console

EMM solutions typically take the form of an EMM console—a web application you develop that allows IT admins to manage their organization, devices, and apps. To support these functions for Android, you integrate your console with the APIs and UI components provided by Android Enterprise.

### 2.5.2. DPC (MDM Agent)

All Android devices that an organization manages through your EMM console must install a DPC app during setup. A DPC is an agent that applies the management policies set in your EMM console to devices. Depending on which [development option you choose](#), you can couple your EMM solution with the EMM solution's DPC, [Android's DPC](#), or with a [custom DPC that you develop](#).

End users can provision a fully managed or dedicated device using a DPC identifier (e.g. "afw#"), according to the implementation guidelines defined in the [Play EMM API](#) developer documentation.

- The EMM's DPC must be publicly available on Google Play, and the end user must be able to install the DPC from the device setup wizard by entering a DPC-specific identifier.
- Once installed, the EMM's DPC must guide the user through the process of provisioning a fully managed or dedicated device.

## 2.6. Work Profile Separation

Fully managed devices with work profiles are for company-owned devices that are used for both work and personal purposes. The organization still manages the entire device. However, the separation of work data and apps into a work profile allows organizations to enforce two separate sets of policies. For example:

- A stronger set of policies for the work profile that applies to all work apps and data.
- A more lightweight set of policies for the personal profile that applies to the user's personal apps and data.

You can learn more about work profile separation in section 7.

## 2.7. VPN Connectivity

IT admins can specify an Always On VPN to ensure that data from specified managed apps will always go through a configured VPN. **Note:** this feature requires deploying a VPN client that supports both Always On and per-app VPN features. IT admins can [specify an arbitrary VPN application \(specified by the application package name\)](#) to be set as an Always On VPN. IT admins can use managed configurations to specify the VPN settings for an app.

You can read more about VPN configuration options in section 6.

## 2.8. Audit Logging

IT admins can gather usage data from devices that can be parsed and programmatically evaluated for malicious or risky behavior. Activities logged include Android Debug Bridge (adb) activity, app launches, and screen unlocks.

- IT admins can [enable security logging](#) for target devices, and the MDM's DPC must be able to retrieve both [security logs](#) and [pre-reboot security logs](#) automatically.
- IT admins can review [enterprise security logs](#) for a given device and configurable time window, in the MDM console.
- IT admins can export enterprise security logs from the MDM console.

IT admins can also capture relevant logging information from Logcat which does not require any additional configuration to be enabled.

You can see a detailed audit logging table in section 9, along with information on how to view and export the different types of audit logs.

## 3. Security Configuration

The Pixel phones offer a rich built-in interface and MDM callable interface for security configuration. This section identifies the security parameters for configuring your device in Common Criteria mode and for managing its security settings.

### 3.1. Management Enrollment

The Pixel phones offer a built-in DPC that can act as the MDM Agent. This is found in an application called Device Policy (this can be seen in Settings>Apps>All Apps>Device Policy). The device can utilize the built-in DPC or an installed DPC to be managed in the evaluated configuration.

There are many possible ways to enroll a Pixel device into management utilizing the built-in MDM Agent. These can be found in more detail at the Android Management API page and [device provisioning](#).

To be in the evaluated configuration, the MDM Server account needs to be configured to be in the Common Criteria (NIAP) configuration. This is handled through the MDM Server and varies with the integration. It may be enabled as a setting within the MDM Server or as a direct configuration of the account through a support request if the MDM Server does not provide the setting directly.

Once the MDM Server account has been properly configured, the client will automatically be configured properly as it enrolls with the MDM Server.

#### 3.1.1. Android Device Policy

When using the built-in Android Device Policy as the DPC, enrollment is handled the same as for any other DPC, but without the need to download and install a separate DPC application (instructions provided by the MDM administrator). The Android Device Policy agent in the evaluated configuration utilizes a Google server (<https://m.google.com>) as the server it communicates with. When using the Android Device Policy agent, the MDM Server will send policy commands and receive responses through the Google server. The MDM Server in this case needs to be configured to work with the Google services to provide management to utilize the Android Device Policy (this is handled as part of the MDM Server configuration and is automatic for the agent). All enrollment actions (including key import for the purposes of verifying the policies) happen automatically for the user during the enrollment process.

### 3.2. Common Criteria Mode

To configure the device into Common Criteria Mode, you must set the following options:

1. Require a lockscreen password
  - Please review the Password Management items in section 3.4 (Common Criteria Related Settings)
2. Disable Smart Lock
  - Smart Lock can be disabled using [KEYGUARD\\_DISABLE\\_TRUST\\_AGENTS](#)
3. Disable Face Unlock (Pixel 10 Pros, 10a, 9, 8, 7 and Fold)
  - Face unlock can be disabled using [KEYGUARD\\_DISABLE\\_FACE](#)
4. Enable Encryption of Wi-Fi and Bluetooth secrets

- This can be enabled by using `setCommonCriteriaModeEnabled()`
- 5. Disable Debugging Features (Developer options)
  - By default Debugging features are disabled. The system administrator can prevent the user from enabling them by using [`DISALLOW\_DEBUGGING\_FEATURES\(\)`](#)
- 6. Disable installation of applications from unknown sources
  - This can be disabled by using [`DISALLOW\_INSTALL\_UNKNOWN\_SOURCES\(\)`](#)
- 7. Turn off usage & diagnostics
  - Open your device's Settings app
  - Tap Google, then More, then Usage & diagnostics
  - Turn Usage & diagnostics off
- 8. Enable Audit Logging
  - Audit Logging can be enabled using [`setSecurityLoggingEnabled.`](#)
  - For certain items Logcat can be used which does not require any additional enablement
- 9. Applications that require PP\_MD\_V3.3 compliant Sensitive Data Protection, Hostname Checking, Revocation Checking, or TLS Ciphersuite restriction must implement the NIAPSEC library.

No additional configuration is required to ensure key generation, key sizes, hash sizes, and all other cryptographic functions meet NIAP requirements.

### 3.3. Cryptographic Module Identification

The TOE implements CAVP certified cryptographic algorithms that are provided by the following cryptographic components:

1. BoringSSL Library:
  - BoringCrypto version 20250107
2. The TOE's LockSettings service
  - Android LockSettings service KBKDF (version 77561fc30db9aedc1f50f5b07504aa65b4268b88)
3. Hardware Cryptography:
  - TOE's Wi-Fi Chipset provides an AES-CCMP implementation
  - The TOE's application processor (Google Tensor G5, Google Tensor G4, Google Tensor G3, Google Tensor G2, and Google Tensor) provides additional cryptographic algorithms. The CAVP certificates correctly identify the specific hardware.
4. Secure Chipset (Titan M2):
  - Titan M2 hardware version H1D3M, firmware MP-SC-05a (Pixel 10, 9, 8, 7, Fold Tablet series devices, not available on the Pixel 6 series)

The use of other cryptographic components beyond those listed above was neither evaluated nor tested during the TOE's Common Criteria evaluation.

No additional configuration is needed for the cryptographic modules in order to be compliant.

### 3.4. Permissions Model

Android runs all apps inside sandboxes to prevent malicious or buggy app code from compromising other apps or the rest of the system. Because the application sandbox is enforced in the kernel, this enforcement extends to the entire app regardless of the specific development environment, APIs used, or programming language. A memory corruption error in an app only allows arbitrary code execution in the context of that particular app, with the permissions enforced by the OS.

Similarly, system components run in least-privileged sandboxes in order to prevent compromises in one component from affecting others. For example, externally reachable components, like the media server and WebView, are isolated in their own restricted sandbox.

Android employs several sandboxing techniques, including Security-Enhanced Linux (SELinux), seccomp, and file-system permissions.

The purpose of a *permission* is to protect the privacy of an Android user. Android apps must request permission to access sensitive user data (such as contacts and SMS), as well as certain system features (such as camera and internet). Depending on the feature, the system might grant the permission automatically or might prompt the user to approve the request.

A central design point of the Android security architecture is that no app, by default, has permission to perform any operations that would adversely impact other apps, the operating system, or the user. This includes reading or writing the user's private data (such as contacts or emails), reading or writing another app's files, performing network access, keeping the device awake, and so on.

The DPC can pre-grant or pre-deny specific permissions using [PERMISSION\\_GRANT\\_STATE](#) API's. In addition the end user can revoke a specific apps permission by:

1. Tapping on Settings>Apps & notifications
2. Tapping on the particular app and then tapping on Permissions
3. From there the user can toggle off any specific permission

You can learn more about Android Permissions on [developer.android.com](https://developer.android.com).

### 3.5. Common Criteria Related Settings

The Common Criteria evaluation requires a range of security settings be available. Those security settings are identified in the table below. In many cases, the administrator or user has to have the ability to configure the setting but no specific value is required.

| Security Feature    | Setting                 | Description                                           | Required Value    | Device API                                            | Android Management API                     | User Interface                                                                                          |
|---------------------|-------------------------|-------------------------------------------------------|-------------------|-------------------------------------------------------|--------------------------------------------|---------------------------------------------------------------------------------------------------------|
| Encryption          | Device Encryption       | Encrypts all internal storage                         | N/A               | Encryption on by default with no way to turn off      |                                            |                                                                                                         |
|                     | Wipe Device             | Removes all data from device                          | No required value | <a href="#">wipeData()</a>                            | <a href="#">enterprises.devices.delete</a> | To wipe the device go to Settings>System>Reset options and select <b>Erase all data (factory reset)</b> |
|                     | Wipe Enterprise Data    | Remove all enterprise data from device                | No required value | <a href="#">wipeData()</a> called from secondary user |                                            |                                                                                                         |
| Password Management | Password Length         | Minimum number of characters in a password            | No required value | <a href="#">setPasswordMinimumLength()</a>            | <a href="#">PasswordRequirements</a>       | To set a screen lock go to Settings>Security & location>Screen lock and tap on <b>Password</b>          |
|                     | Password Complexity     | Specify the type of characters required in a password | No required value | <a href="#">setPasswordQuality()</a>                  |                                            | To set a screen lock go to Settings>Security & location>Screen lock and tap on <b>Password</b>          |
|                     | Password Expiration     | Maximum length of time before a password must change  | No required value | <a href="#">setPasswordExpirationTimeout()</a>        |                                            |                                                                                                         |
|                     | Authentication Failures | Maximum number of authentication failures             | Between 1-50      | <a href="#">setMaximumFailedPasswordsForWipe()</a>    |                                            |                                                                                                         |

| Security Feature | Setting               | Description                                                                    | Required Value                     | Device API                                     | Android Management API               | User Interface                                                                                                                                                                |
|------------------|-----------------------|--------------------------------------------------------------------------------|------------------------------------|------------------------------------------------|--------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Lockscreen       | Inactivity to lockout | Time before lockscreen is engaged                                              | No required value                  | <a href="#">setMaximumTimeToLock()</a>         | <a href="#">PasswordRequirements</a> | To set an inactivity lockout go to Settings>Security & location> and tap on the gear icon next to Screen lock then tap on Automatically lock and select the appropriate value |
|                  | Banner                | Banner message displayed on the lockscreen                                     | Administrator or user defined text | <a href="#">setDeviceOwnerLockScreenInfo()</a> | deviceOwnerLockScreenInfo            | To set a banner go to Settings>Display>Lock screen>Add text on lock screen. Set a message and tap <b>Save</b>                                                                 |
|                  | Remote Lock           | Locks the device remotely                                                      | Function must be available         | <a href="#">lockNow()</a>                      | Device issueCommand: LOCK            | The user can lock the device by pressing the power button. This can be configured for immediate locking or to start the timer to a lock.                                      |
|                  | Show Password         | Disallows the displaying of the password on the screen of lock-screen password | Disable                            | This is disabled by default                    |                                      | To set the power button to immediate lock go to Settings>Security & privacy>Device lock>Screen lock settings (gear icon)>Power button instantly locks                         |

| Security Feature       | Setting                       | Description                                                                     | Required Value    | Device API                                                                                                           | Android Management API                                           | User Interface                                                                                               |
|------------------------|-------------------------------|---------------------------------------------------------------------------------|-------------------|----------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
|                        | Notifications                 | Controls whether notifications are displayed on the lockscreen                  | Enable/Disable    | <a href="#">KEYGUARD_DISABLE_SECURE_NOTIFICATIONS()</a><br><a href="#">KEYGUARD_DISABLE_UNREDACTED_NOTIFICATIONS</a> | keyguardDisabledFeatures                                         |                                                                                                              |
|                        | Control Biometric Face        | Control the use of Biometric Face authentication factor                         | Enable/Disable    | <a href="#">KEYGUARD_DISABLE_FACE</a>                                                                                | <a href="#">PasswordRequirements</a><br>keyguardDisabledFeatures |                                                                                                              |
|                        | Control Biometric Fingerprint | Control the use of Biometric Fingerprint authentication factor                  | Enable/Disable    | <a href="#">KEYGUARD_DISABLE_FINGERPRINT</a>                                                                         | <a href="#">PasswordRequirements</a><br>keyguardDisabledFeatures |                                                                                                              |
|                        |                               |                                                                                 |                   |                                                                                                                      |                                                                  |                                                                                                              |
| Certificate Management | Import CA Certificates        | Import CA Certificates into the Trust Anchor Database or the credential storage | No required value | <a href="#">installCaCert()</a>                                                                                      |                                                                  | Tap on Settings>Security & location>Advanced>Encryption & credentials and select <b>Install from storage</b> |

| Security Feature | Setting                    | Description                                                                  | Required Value    | Device API                           | Android Management API         | User Interface                                                                                                                                                                                                                                                                                                                                                                       |
|------------------|----------------------------|------------------------------------------------------------------------------|-------------------|--------------------------------------|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                  | Remove Certificates        | Remove certificates from the Trust Anchor Database or the credential storage | No required value | <a href="#">uninstallCACert()</a>    | CERT_INSTALL<br>CERT_SELECTION | To clear all user installed certificates tap on Settings>Security & location>Advanced>Encryption & credentials and select Clear credentials<br>To remove a specific user installed certificate tap on Settings>Security & location>Advanced>Encryption & credentials>Trusted credentials. Switch to the User tab, select the certificate you want to delete and tap on <b>Remove</b> |
|                  | Import Client Certificates | Import client certificates in to Keychain                                    | No required value | <a href="#">installKeyPair()</a>     |                                | Tap on Settings>Security & location>Advanced>Encryption & credentials and select <b>Install from storage</b>                                                                                                                                                                                                                                                                         |
|                  | Remove Client Certificates | Remove client certificates from Keychain                                     | No required value | <a href="#">removeKeyPair()</a>      |                                | To remove a specific user installed client certificate tap on Settings>Security & location>Advanced>Encryption & credentials>User credentials. Switch to the User tab, select the certificate you want to delete and tap on <b>Remove</b>                                                                                                                                            |
|                  |                            |                                                                              |                   |                                      |                                |                                                                                                                                                                                                                                                                                                                                                                                      |
| Radio Control    | Control Wi-Fi              | Control access to Wi-Fi                                                      | Enable/Disable    | <a href="#">DISALLOW_CONFIG_WIFI</a> | wifiConfigDisabled             | To disable Wi-Fi tap on Settings>Network & internet and toggle Airplane mode to <b>On</b>                                                                                                                                                                                                                                                                                            |

| Security Feature | Setting                  | Description                                                                                   | Required Value    | Device API                                                                                                                    | Android Management API                | User Interface                                                                                                                                                                                                                                                                                                                                                               |
|------------------|--------------------------|-----------------------------------------------------------------------------------------------|-------------------|-------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                  | Control Cellular         | Control access to Cellular                                                                    | Enable/Disable    | <a href="#">DISALLOW_CONFIG_MOBILE_NETWORKS</a>                                                                               | mobileNetworksConfigDisabled          | <p>To disable Cellular tap on Settings&gt;Network &amp; internet&gt;Mobile network and tap on your carrier and toggle to <b>Off</b></p> <p>To disable 2G go to Settings&gt;Network &amp; internet&gt;SIMs&gt;Carrier and toggle Allow 2G to <b>Off</b></p> <p>To disable NFC tap on Settings&gt;Connected devices&gt;Connection preferences and toggle NFC to <b>Off</b></p> |
|                  | Restrict 2G Cellular     | Control access to 2G Cellular                                                                 | Enable/Disable    | <a href="#">DISALLOW_CELLULAR_2G</a>                                                                                          | no_cellular_2g                        |                                                                                                                                                                                                                                                                                                                                                                              |
|                  | Control NFC              | Control access to NFC                                                                         | Enable/Disable    | <a href="#">DISALLOW_OUTGOING_BEAM</a>                                                                                        | outgoingBeamDisabled                  |                                                                                                                                                                                                                                                                                                                                                                              |
|                  | Control Bluetooth        | Control access to Bluetooth                                                                   | Enable/Disable    | <a href="#">DISALLOW_BLUETOOTH</a><br><a href="#">DISALLOW_BLUETOOTH_SHARING</a><br><a href="#">DISALLOW_CONFIG_BLUETOOTH</a> | bluetoothDisabled                     |                                                                                                                                                                                                                                                                                                                                                                              |
|                  | Control Location Service | Control access to Location Service                                                            | Enable/Disable    | <a href="#">DISALLOW_SHARE_LOCATION</a><br><a href="#">DISALLOW_CONFIG_LOCATION</a>                                           | shareLocationDisabled<br>locationMode |                                                                                                                                                                                                                                                                                                                                                                              |
|                  |                          |                                                                                               |                   |                                                                                                                               |                                       |                                                                                                                                                                                                                                                                                                                                                                              |
| Wi-Fi Settings   | Specify Wi-Fi SSIDs      | Specify SSID values for connecting to Wi-Fi. Can also create white and black lists for SSIDs. | No required value | <a href="#">WifiEnterpriseConfig()</a>                                                                                        |                                       |                                                                                                                                                                                                                                                                                                                                                                              |

| Security Feature | Setting                                | Description                                               | Required Value    | Device API                                 | Android Management API      | User Interface                                                                                                  |
|------------------|----------------------------------------|-----------------------------------------------------------|-------------------|--------------------------------------------|-----------------------------|-----------------------------------------------------------------------------------------------------------------|
|                  | Set WLAN CA Certificate                | Select the CA Certificate for the Wi-Fi connection        | No required value |                                            | <a href="#">WiFiManager</a> |                                                                                                                 |
|                  | Specify security type                  | Specify the connection security (WEP, WPA2, WPA3, etc)    | No required value |                                            |                             |                                                                                                                 |
|                  | Select authentication protocol         | Specify the EAP-TLS connection values                     | No required value |                                            |                             |                                                                                                                 |
|                  | Select client credentials/certificates | Specify the client credentials to access a specified WLAN | No required value |                                            |                             |                                                                                                                 |
|                  | Control Always-on VPN                  | Control access to Always-on VPN                           | Enable/Disable    | <a href="#">setAlwaysOnVPNPackage()</a>    | alwaysOnVpnPackage          |                                                                                                                 |
| Hardware Control | Control Microphone (across device)     | Control access to microphone across the device            | Enable/Disable    | <a href="#">DISALLOW_UNMUTE_MICROPHONE</a> | microphoneAccess            |                                                                                                                 |
|                  | Control Microphone (per-app basis)     | Control access to microphone per application              | Enable/Disable    |                                            |                             | Tap on 'Settings>Apps & notifications>App permissions>Microphone' then de-select the apps to remove permissions |
|                  | Control Camera (per-app basis)         | Control access to camera per application                  | Enable/Disable    |                                            | cameraAccess                | Tap on 'Settings>Apps & notifications>App permissions>Camera' then de-select the apps to remove permissions     |

| Security Feature    | Setting                                | Description                                                 | Required Value    | Device API                                    | Android Management API                                                | User Interface                                                                                                                |
|---------------------|----------------------------------------|-------------------------------------------------------------|-------------------|-----------------------------------------------|-----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
|                     | Control USB Mass Storage               | Control access to mounting the device for storage over USB. | Enable/Disable    | <a href="#">DISALLOW_MOUNT_PHYSICAL_MEDIA</a> | usbMassStorageEnabled<br><br>UsbDataAccess.DISALLOW_USB_DATA_TRANSFER |                                                                                                                               |
|                     | Control USB Debugging                  | Control access to USB debugging.                            | Enable/Disable    | <a href="#">DISALLOW_DEBUGGING_FEATURES</a>   | AdvancedSecurityOverrides.developerSettings                           |                                                                                                                               |
|                     | Control USB Tethered Connections       | Control access to USB tethered connections.                 | Enable/Disable    | <a href="#">DISALLOW_CONFIG_TETHERING</a>     | DeviceConnectivityManagement.teheringSettings                         |                                                                                                                               |
|                     | Control Bluetooth Tethered Connections | Control access to Bluetooth tethered connections.           | Enable/Disable    | <a href="#">DISALLOW_CONFIG_TETHERING</a>     | DeviceConnectivityManagement.teheringSettings                         |                                                                                                                               |
|                     | Control Hotspot Connections            | Control access to Wi-Fi hotspot connections                 | Enable/Disable    | <a href="#">DISALLOW_CONFIG_TETHERING</a>     | DeviceConnectivityManagement.teheringSettings                         |                                                                                                                               |
|                     | Automatic Time                         | Allows the device to get time from the Wi-Fi connection     | Enable/Disable    | <a href="#">setAutoTimeRequired()</a>         | autoTimeRequired                                                      |                                                                                                                               |
|                     |                                        |                                                             |                   |                                               |                                                                       | Tap on 'Settings>System>Date & time' and toggle Automatic date & time to On                                                   |
| Application Control | Install Application                    | Installs specified application                              | No required value | <a href="#">PackageInstaller.Session()</a>    |                                                                       | To uninstall an application tap on Settings>Applications & notifications>See all. Select the application and tap on Uninstall |
|                     | Uninstall Application                  | Uninstalls specified application (and its data)             | App to uninstall  | <a href="#">uninstall()</a>                   |                                                                       |                                                                                                                               |

| Security Feature | Setting                | Description                                                     | Required Value    | Device API                                                                                                                    | Android Management API                                                       | User Interface                                                                                                                    |
|------------------|------------------------|-----------------------------------------------------------------|-------------------|-------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
|                  | Application Allowlist  | Specifies a list of applications that may be installed          | No required value | This is done by the EMM/MDM when they setup an application catalog which leverages <a href="#">PackageInstaller.Session()</a> | Applications (direct application management is handled through Managed Play) |                                                                                                                                   |
|                  | Application Blocklist  | Specifies a list of applications that may not be installed      | No required value | <a href="#">PackageInstaller.SessionInfo()</a>                                                                                |                                                                              |                                                                                                                                   |
|                  | Application Repository | Specifies the location from which applications may be installed | No required value | <a href="#">DISALLOW_INSTALL_UNKNOWN_SOURCES</a>                                                                              | advancedSecurityOverrides.untrustedAppsPolicy                                |                                                                                                                                   |
|                  |                        |                                                                 |                   |                                                                                                                               |                                                                              |                                                                                                                                   |
| TOE Management   | Enrollment             | Enroll TOE in management                                        | No required value |                                                                                                                               |                                                                              | During device setup scan EMM/MDM provided QR code or enter EMM/MDM DPC identifier                                                 |
|                  | Disallow Unenrollment  | Prevent the user from removing the managed profile              | Enable/Disable    | <a href="#">DISALLOW_FACTORY_RESET</a>                                                                                        | factoryResetDisabled                                                         | Refer to section 2.5.2 for more details<br>The only way for a user to remove management from a device is through a factory reset. |

| Security Feature | Setting                                           | Description                                                           | Required Value   | Device API                                                                                                                                                                                                    | Android Management API                          | User Interface                                                                                                                                                                                |
|------------------|---------------------------------------------------|-----------------------------------------------------------------------|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                  | Unenrollment                                      | Unenroll TOE from management                                          | App to uninstall | <a href="#">uninstall()</a> – this API can be used to uninstall the MDM Agent from the device. Uninstalling the MDM agent from an enterprise profile will delete the entire profile and all its applications. |                                                 | This API can be used to uninstall enterprise apps. If an admin uninstalls the MDM agent installed on an enterprise profile, the entire profile and all enterprise applications are deleted.   |
|                  | Backup Services                                   | Controls backup services available to applications & profile settings | Enable/Disable   | <a href="#">setBackupServiceEnabled()</a>                                                                                                                                                                     | AdvancedSecurityOverrides<br>.developerSettings | Tap on Settings>System>Backup to configure backup settings for the device<br><br>By default this is hidden. If available, it is under Setting>System>Developer options>User developer options |
|                  | Allow Developer Mode                              | Controls Developer Mode access                                        | Enable/Disable   | <a href="#">DISALLOW_DEBUGGING_FEATURES</a>                                                                                                                                                                   |                                                 | To activate, go to System>Settings>About phone and tap on Build number until enabled                                                                                                          |
|                  | Sharing Data Between Enterprise and Personal Apps | Controls data sharing between enterprise and work apps                | Enable/Disable   | <a href="#">DISALLOW_CROSS_PROFILE_COPY_PASTE</a><br><a href="#">addCrossProfileIntentFilter()</a>                                                                                                            | crossProfilePolicies                            |                                                                                                                                                                                               |

### 3.6. Password Recommendations

When setting a password, you should select a password that:

- Does not use known information about yourself (e.g. pets names, your name, kids names or any information available in the public domain);
- Is significantly different from previous passwords (adding a '1' or "!" to the end of the password is not sufficient); or
- Does not contain a complete word. (Password!).
- Does not contain repeating or sequential numbers and/or letters.

### 3.7. Bug Reporting Process

Google supports a bug filing system for the Android OS outlined here:

<https://source.android.com/setup/contribute/report-bugs>.

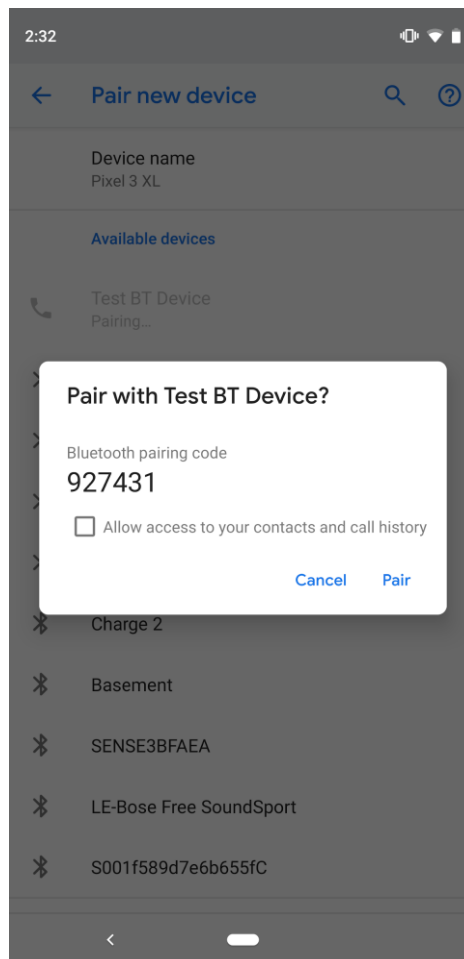
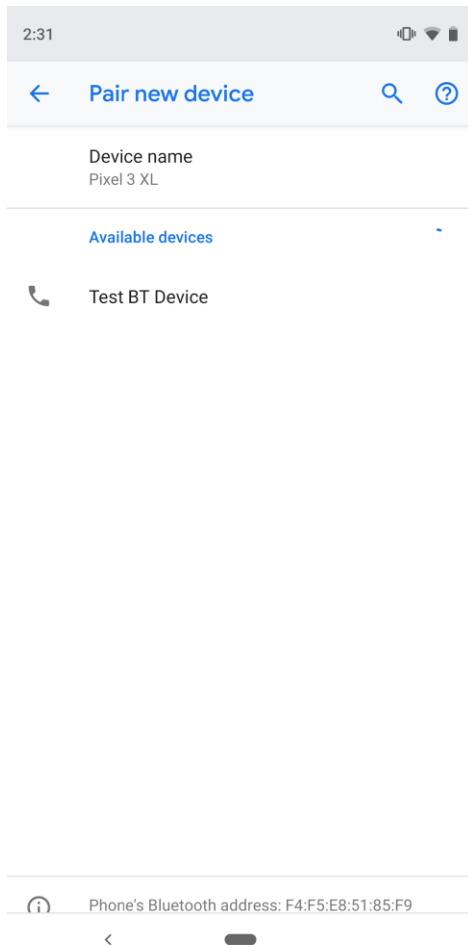
This allows developers or users to search for, file, and vote on bugs that need to be fixed. This helps to ensure that all bugs that affect large numbers of people get pushed up in priority to be fixed.

## 4. Bluetooth Configuration

Follow the below steps to pair and connect using Bluetooth

### Pair

1. Open your phone or tablet's Settings app .
2. Tap Connected devices > Connection preferences > Bluetooth. Make sure Bluetooth is turned on.
3. Tap Pair new device.
4. Tap the name of the Bluetooth device you want to pair with your phone or tablet.
5. Follow any on-screen steps.



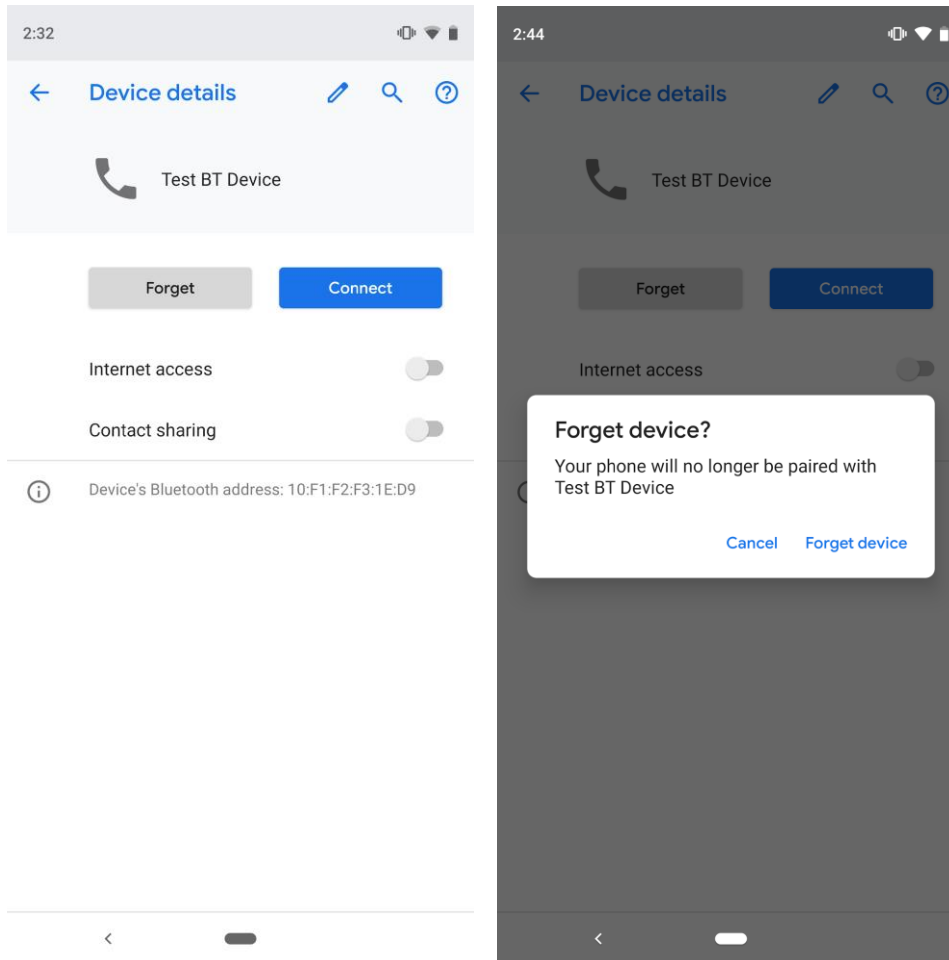
### Connect

1. Open your phone or tablet's Settings app .
2. Tap Connected devices > Connection preferences > Bluetooth.
3. Make sure Bluetooth is turned on.
4. In the list of paired devices, tap a paired but unconnected device.
5. When your phone or tablet and the Bluetooth device are connected, the device shows as "Connected" in the list.

Tip: If your phone is connected to something through Bluetooth, at the top of the screen, you'll see a Bluetooth icon .

## Remove Previously Paired Device

1. Open your phone or tablet's Settings app .
2. Tap Connected devices > Previously connected devices
3. Tap the gear icon to the right of the device you want to unpair
4. Tap on Forget and confirm in the popup window by tapping on Forget device



For additional support information around Bluetooth please take a look at this [support link](#).

## 5. Wi-Fi Configuration

Android supports the WPA2-Enterprise (802.11i) and WPA3-Enterprise protocols, which are specifically designed for enterprise networks and can be integrated into a broad range of Remote Authentication Dial-In User Service (RADIUS) authentication servers.

IT admins can silently provision enterprise Wi-Fi configurations on managed devices, including:

- SSID, via the [EMM's DPC](#)
- Password, via the [EMM's DPC](#)
- Identity, via the [EMM's DPC](#)
- Certificate for clients authorization, via the [EMM's DPC](#)
- CA certificate(s), via the [EMM's DPC](#)

IT admins can lock down Wi-Fi configurations on managed devices, to prevent users from creating new configurations or modifying corporate configurations.

IT admins can lock down corporate Wi-Fi configurations in either of the following configurations:

- Users cannot modify [any Wi-Fi configurations provisioned by the EMM](#), but may add and modify their own user-configurable networks (for instance personal networks).
- Users cannot [add or modify any Wi-Fi network on the device](#), limiting Wi-Fi connectivity to just those networks provisioned by the EMM.

When the device tries to connect to a Wi-Fi network it performs a standard captive portal check which bypasses the full tunnel VPN configuration. If the administrator wants to turn the captive portal check off they need to do this physically on the device before enrolling it in to the MDM by:

1. Enable Developer Options by tapping on Settings>About phone and tapping on Build number five times until they see that Developer options has been enabled
2. Enable Android Debug Bridge (ADB) over USB by tapping on Settings>System>Advanced>Developer options and scroll down to USB debugging and enable the toggle to On
3. Connect to the device to a workstation that has ADB installed and type in “adb shell settings put global captive\_portal\_mode 0” and hit enter
4. You can verify the change by typing “adb shell settings get global captive\_portal\_mode” and the return value should be “0”
5. Turn off Developer options by tapping on Settings>System>Advanced>Developer options and toggling the On option to Off at the top

If a Wi-Fi connection unintentionally terminates, the end user will need to reconnect to re-establish the session.

## 6. VPN Configuration

Android supports securely connecting to an enterprise network using VPN:

- **Always-on VPN**—The VPN can be configured so that apps don't have access to the network until a VPN connection is established, which prevents apps from sending data across other networks.
  - Always-on VPN supports VPN clients that implement [VpnService](#). The system automatically starts that VPN after the device boots. [Device owners](#) and [profile owners](#) can direct work apps to always connect through a specified VPN. Additionally, users can manually set Always-on VPN clients that implement VpnService methods using **Settings>More>VPN**. Always-on VPN can also be enabled manually from the settings menu.

## 7. Work Profile Separation

Work profile mode is initiated when the DPC initiates a [managed provisioning flow](#). The work profile is based on the Android [multi-user](#) concept, where the work profile functions as a separate Android user segregated from the primary profile. The work profile shares common UI real estate with the primary profile. Apps, notifications, and widgets from the work profile show up next to their counterparts from the primary profile and are always badged so users have an indication as to what type of app it is.

With the work profile, enterprise data does not intermix with personal application data. The work profile has its own apps, downloads folder, settings, and KeyChain. It is encrypted using its own encryption key, and it can have its own passcode to gate access.

The work profile is [provisioned](#) upon installation, and the user can only remove it by removing the entire work profile. Administrators can also remotely instruct the device policy client to remove the work profile, for instance, when a user leaves the organization or a device is lost. Whether the user or an IT administrator removes the work profile, user data in the primary profile remains on the device.

A DPC running in profile owner mode can require users to specify a security challenge for apps running in the work profile. The system shows the security challenge when the user attempts to open any work apps. If the user successfully completes the security challenge, the system unlocks the work profile and decrypts it, if necessary.

Android also provides support for a separate work challenge to enhance security and control. The work challenge is a separate passcode that protects work apps and data. Admins managing the work profile can choose to set the password policies for the work challenge differently from the policies for other device passwords. Admins managing the work profile set the challenge policies using the usual [DevicePolicyManager](#) methods, such as [setPasswordQuality\(\)](#) and [setPasswordMinimumLength\(\)](#). These admins can also configure the primary device lock, by using the DevicePolicyManager instance returned by the `DevicePolicyManager.getParentProfileInstance()` method.

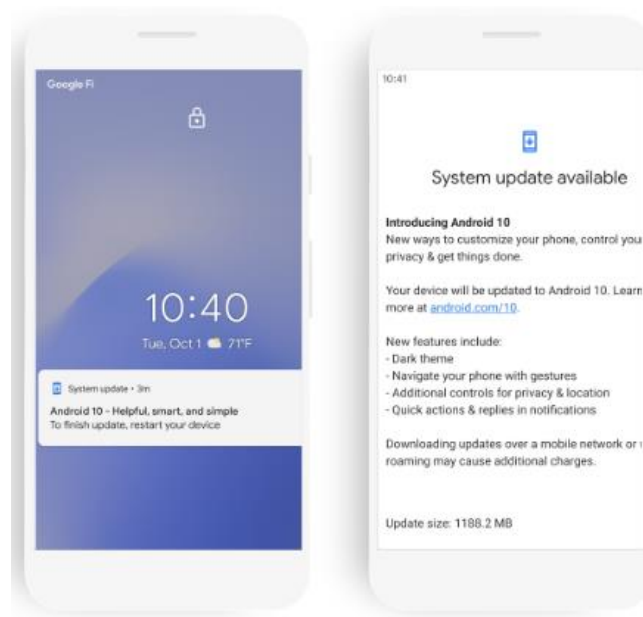
As part of setting up a separate work challenge, users may also elect to enroll fingerprints to unlock the work profile more conveniently. Fingerprints must be enrolled separately from the primary profile as they are not shared across profiles.

As with the primary profile, the work challenge is verified within secure hardware, ensuring that it's difficult to brute-force. The passcode, mixed in with a secret from the secure hardware, is used to derive the disk encryption key for the work profile, which means that an attacker cannot derive the encryption key without either knowing the passcode or breaking the secure hardware.

## 8. Secure Update Process

Over the Air (OTA) updates (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. Additionally, the Pixel phones also provide roll-back protection for OTA updates to prevent a user from installing a prior/previous version of software by check.

The Pixel phones leverage [A/B system updates](#), also known as seamless updates. This approach ensures that a workable booting system remains on the disk during an over-the-air (OTA) update. This approach reduces the likelihood of an inactive device after an update, which means fewer device replacements and device reflashes at repair and warranty centers. Other commercial-grade operating systems such as ChromeOS also use A/B updates successfully.



The user will get a notification when an update is made available. No special configuration will be required to ensure a secure update process.

### 8.1. Google Play System Updates

Google Play System Updates offer a simple and fast method to deliver updates. End-user devices receive the components from the Google Play Store or through a partner-provided over-the-air (OTA) mechanism.

The components are delivered as either APK or APEX files — APEX is a new file format which loads earlier in the booting process. Important security and performance improvements that previously needed to be part of full OS updates can be downloaded and installed similarly to an app update. Updates delivered in this way are secured by being cryptographically signed.

Google Play System Updates can also deliver faster security fixes for critical security bugs by modularizing media components, which accounted for nearly 40% of recently patched vulnerabilities, and allowing updates to conscript, the Java Security Provider.

## 9. Audit Logging

### **Security Logs:**

A MDM agent acting as Device Owner can control the logging with [DevicePolicyManager#setSecurityLoggingEnabled](#). When the SecurityLog is enabled, device owner apps receive periodic callbacks from [DeviceAdminReceiver#onSecurityLogsAvailable](#), at which time new batch of logs can be collected [via DevicePolicyManager#retrieveSecurityLogs](#). SecurityEvent describes the type and format of security logs being collected.

Audit events from the Security Log are marked with SecurityLog. The format of the events is:

<Keyword> (<Date><Timestamp>): <message>

### **Logcat Logs:**

Logcat logs can be read by a command issued via an ADB shell running on the phone. Information about reading Logcat logs can be found [here](#). The command to issue a dump of the logcat logs is:

> adb logcat

When using the Android DPC as the MDM Agent, logcat needs to be configured to collect additional events to fully cover everything presented here (if the Android DPC is not being used, this step is not required):

> adb logcat -G 32M && adb shell setprop persist.log.tag.dpcsupport VERBOSE && adb shell setprop persist.log.tag.cloudipc VERBOSE && adb shell setprop persist.log.tag.Volley VERBOSE; adb shell setprop persist.log.tag.all VERBOSE

Logcat logs cannot be exported from the device outside of using the above ADB command to dump to a file, then retrieving the file from the device (which requires developer settings enabled and administrative permissions).

Logcat logs can also be read by an application (for example an MDM agent) granted permission from an ADB shell:

> adb shell pm grant <application\_package\_name> android.permission.READ\_LOGS

Audit events from the Logcat log are marked with Logcat. The format of the events is:

<Date> <Time> <ID> | <Keyword> <Message>

The table below provides audit events for the general system:

| Requirement   | Auditable Events                                               | Additional Audit Record Contents                    | Log Events & Examples                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------|----------------------------------------------------------------|-----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FAU_GEN.1     | Start-up and shutdown of the audit functions                   |                                                     | <p>SecurityLog</p> <p>Start-up:<br/>LOGGING_STARTED (Thu Sep 24 10:53:19 EDT 2020)</p> <p>Shutdown:<br/>All logs are stored in memory. When audit functions are disabled, all memory being used by the audit functions is released by the OS, and so this log cannot be seen.</p> <p>Subject identity: Android system (not shown)</p>                                                                                                                                        |
|               | Start-up and shutdown of the Rich OS                           |                                                     | <p>SecurityLog</p> <p>&lt;verified boot status&gt; is green (all verified), yellow (boot partition only verified) or orange (device unlocked)</p> <p>&lt;dm-verify status&gt; is enforcing, eio (error) or disabled</p> <p>Start-up:<br/>OS_STARTUP: [&lt;verified boot status&gt; &lt;dm-verity status&gt;]</p> <p>Shutdown:<br/>All logs are stored in memory. This log is not capturable or persistent through boot, and thus isn't available to an MDM Administrator</p> |
| FCS_STG_EXT.1 | Import or destruction of key.                                  | Identity of key.<br>Role and identity of requestor. | <p>SecurityLog</p> <p>KEY_IMPORTED &lt;timestamp&gt;: &lt;success-boolean&gt; &lt;key name&gt; &lt;requesting process / role / identity&gt;</p> <p>SecurityLog</p> <p>KEY_DESTROYED &lt;timestamp&gt;: &lt;success-boolean&gt; &lt;key name&gt; &lt;requesting process / role / identity&gt;</p>                                                                                                                                                                             |
| FCS_STG_EXT.3 | Failure to verify integrity of stored key.                     | Identity of key being verified.                     | KEY_INTEGRITY_VIOLATION (Thu Oct 29 16:20:44 EDT 2020): USRPKEY_"corrupt" 1010                                                                                                                                                                                                                                                                                                                                                                                               |
| FDP_DAR_EXT.2 | [None]                                                         |                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| FDP_STG_EXT.1 | Addition or removal of certificate from Trust Anchor Database. | Subject name of certificate.                        | <p>SecurityLog</p> <p>CERT_AUTHORITY_INSTALLED (Thu Sep 24 12:22:17 EDT 2020): 1 cn=rootca-rsa,1.2.840.113549.1.9.1=#161a726f6f7463612d72736140676f7373616d65727365632e636f6d,o=gss,l=catonsville,st=md,c=us 0</p> <p>CERT_AUTHORITY_REMOVED (Thu Sep 24 12:22:30 EDT 2020): 1 cn=rootca-rsa,1.2.840.113549.1.9.1=#161a726f6f7463612d72736140676f7373616d65727365632e636f6d,o=gss,l=catonsville,st=md,c=us 0</p>                                                             |

|                                                     |                                          |                                   |                                                                                                                                                                                               |
|-----------------------------------------------------|------------------------------------------|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FIA_X509_EXT.1                                      | Failure to validate X.509v3 certificate. | Reason for failure of validation. | Logcat<br>System.err: java.security.cert.CertPathValidatorException [<error message>]<br><br>ValidatableSSLSocket: Failed to establish a TLS connection to <IP address> ... [<error message>] |
| FMT_SMF_EXT.2                                       | [none].                                  | [none].                           |                                                                                                                                                                                               |
| FPT_NOT_EXT.1                                       | [None].                                  | [No additional information].      |                                                                                                                                                                                               |
| FPT_TST_EXT.1                                       | Initiation of self-test.                 | [none]                            | SecurityLog<br><br>CRYPTO_SELF_TEST_COMPLETED (Thu Sep 24 10:53:19 EDT 2020): 1<br><br>Subject identity: Android system (not shown)                                                           |
|                                                     | Failure of self-test.                    |                                   | SecurityLog<br><br>CRYPTO_SELF_TEST_COMPLETED (Thu Sep 24 10:53:19 EDT 2020): 0<br><br>Subject identity: Android system (not shown)                                                           |
| FPT_TST_EXT.2/PR<br>EKERNEL (Selection is optional) | Start-up of TOE.                         | No additional information.        | SecurityLog<br><br>OS_STARTUP (Thu Sep 24 10:53:18 EDT 2020): orange enforcing<br><br>Subject identity: Android system (not shown)                                                            |
|                                                     | [none]                                   | No additional information.        |                                                                                                                                                                                               |

The table below provides audit events for the Wi-Fi connectivity:

| Requirement             | Auditable Events                         | Additional Audit Record Contents | Log Events & Examples                                                                                                                                                                                                                                                                                                        |
|-------------------------|------------------------------------------|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FCS_TLSC_EXT.1/<br>WLAN | Failure to establish an EAP-TLS session. | Reason for failure               | Logcat<br><br>wpa_supplicant: TLS - SSL error: <error message><br><br>wpa_supplicant: wlan0: CTRL-EVENT-EAP-TLS-CERT-ERROR<br><br>wpa_supplicant: wlan0: CTRL-EVENT-EAP-FAILURE<br>EAP authentication failed<br><br><b>Termination:</b><br>wpa_supplicant: wlan0: CTRL-EVENT-DISCONNECTED bssid=<BSSID> reason=<reason code> |

| Requirement                                                           | Auditable Events                                    | Additional Audit Record Contents                                                                         | Log Events & Examples                                                                                                                                                                                                                                         |
|-----------------------------------------------------------------------|-----------------------------------------------------|----------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                       | Establishment/termination of an EAP-TLS session.    | Non-TOE endpoint of connection                                                                           | Logcat<br><br><b>Establishment:</b><br>wpa_supplicant: wlan0: CTRL-EVENT-CONNECTED - Connection to <BSSID> completed<br><br><b>Termination:</b><br>wpa_supplicant: wlan0: CTRL-EVENT-DISCONNECTED bssid=<BSSID> reason=<reason code>                          |
| FIA_X509_EXT.1/WLAN                                                   | Failure to validate X.509v3 certificate.            | Reason for failure of validation.                                                                        | Logcat<br>wpa_supplicant: TLS: Certificate verification failed, <reason/details>                                                                                                                                                                              |
| FIA_X509_EXT.6                                                        | Attempts to load and revoke certificates            | No additional information                                                                                | SecurityLog<br><br><result> 1 = successful, 0 = failure<br><br><key alias>: code string for the key<br><br>KEY_IMPORT: [<result>,<key alias>,<requesting process ID>]<br><br>KEY_DESTRUCTION: [<result>,<key alias>,<requesting process ID>]                  |
| FPT_TST_EXT.3/WLAN<br>(note: can be performed by TOE or TOE platform) | Execution of this set of TSF self-tests.<br>[none]. | [no additional information].                                                                             | These TSF Self-tests are included in the self-tests.                                                                                                                                                                                                          |
| FTA_WSE_EXT.1                                                         | All attempts to connect to access points.           | Certificate Check Message and the MAC address<br><br>Success and failures (including reason for failure) | Logcat<br>wpa_supplicant: wlan0: Trying to associate with SSID <SSID><br><br>wpa_supplicant: wlan0: Associated with <MAC><br><br>For success and failures – see audits for FCS_TLSC_EXT.1/WLAN which will immediately follow the attempt to connect to the AP |
| FTP_ITC.1/WLAN                                                        | All attempts to establish a trusted channel.        | Identification of the non-TOE endpoint of the channel.                                                   | Logcat<br>wpa_supplicant: wlan0: Trying to associate with SSID <SSID><br><br>wpa_supplicant: wlan0: Associated with <MAC>                                                                                                                                     |

The table below provides audit events for the Bluetooth connectivity:

| Requirement   | Auditable Events                                       | Additional Audit Record Contents                        | Log Events & Examples                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------|--------------------------------------------------------|---------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FIA_BLT_EXT.1 | Failed user authorization of Bluetooth device.         | User authorization decision                             | SecurityLog<br><masked MAC> in the form: xx:xx:xx:xx:AA:BB<br><br><Success/Failure>: 1 = successful, 0 = failure<br><br><details> will have information about pairing<br><br>security_bluetooth_connection: [<masked MAC>,<Success/Failure>,<details>]<br><br>LogCat<br><br><state>: 0 is DISCONNECTED, 1 is CONNECTING, 2 is CONNECTED, and 3 is DISCONNECTING<br><br>CachedBluetoothDevice: onProfileStateChanged: profile <profilename>, device <masked MAC>, newProfileState=<state> |
|               | Failed user authorization for local Bluetooth Service. | Bluetooth address and name of device. Bluetooth profile |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| FIA_BLT_EXT.2 | Initiation of Bluetooth connection.                    | Bluetooth address and name of device.                   | SecurityLog<br><br>security_bluetooth_connection: [<masked MAC>,<Success/Failure>,<details>]                                                                                                                                                                                                                                                                                                                                                                                             |
|               | Failure of Bluetooth connection.                       | Reason for failure.                                     | SecurityLog<br><br>security_bluetooth_connection: [<masked MAC>,<Success/Failure>,<details>]                                                                                                                                                                                                                                                                                                                                                                                             |

The below table provides samples management function audits:

| REQUIREMENT               | FUNCTION                     | Required Value             | AUDIT LOG                                                                                                                   |
|---------------------------|------------------------------|----------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| FMT_SMF.1.1<br>Function 1 | minimum password length      | Greater than or equal to 8 | SecurityLog<br><br>PASSWORD_COMPLEXITY_SET (Thu Jun 18 19:50:21 EDT 2020): com.afwsamples.testdpc 0 0 16 393216 0 0 0 0 0 0 |
|                           | minimum password complexity  | No required value          | SecurityLog<br><br>PASSWORD_COMPLEXITY_SET (Sun Jul 05 19:01:57 EDT 2020): com.afwsamples.testdpc 0 0 0 393216 1 0 1 0 0 1  |
|                           | maximum password lifetime    |                            | SecurityLog<br><br>PASSWORD_EXPIRATION_SET (Sun Jul 05 19:03:55 EDT 2020): com.afwsamples.testdpc 0 0 600000                |
| FMT_SMF.1.1<br>Function 2 | screen-lock enabled/disabled | Enabled                    | SecurityLog                                                                                                                 |

| REQUIREMENT               | FUNCTION                                                                                                   | Required Value     | AUDIT LOG                                                                                                                                                                                        |
|---------------------------|------------------------------------------------------------------------------------------------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           |                                                                                                            |                    | PASSWORD_COMPLEXITY_SET (Sun Jul 05 19:01:57 EDT 2020): com.afwsamples.testdpc 0 0 0 393216 1 0 1 0 0 1                                                                                          |
|                           | screen-lock enabled/disabled (after requiring a password above, admin can request the user set a password) | No required value  | Logcat<br><br>09-23 13:17:18.528 1499 6482 I<br>ActivityTaskManager: START u0 {act=android.app.action.SET_NEW_PASSWORD cmp=com.android.settings/.password.SetNewPasswordActivity} from uid 10245 |
|                           | screen lock timeout                                                                                        | 10 minutes or less | SecurityLog<br><br>MAX_SCREEN_LOCK_TIMEOUT_SET (Mon Jul 13 21:39:23 EDT 2020): com.afwsamples.testdpc 0 0 120000                                                                                 |
|                           | screen lock timeout (after setting a max time, the admin can prevent any user changes with this)           |                    | SecurityLog<br><br>USER_RESTRICTION_ADDED (Mon Jul 13 21:42:18 EDT 2020): com.afwsamples.testdpc 0 no_config_screen_timeout                                                                      |
|                           | number of authentication failures                                                                          | 10 or less         | SecurityLog<br><br>MAX_PASSWORD_ATTEMPTS_SET (Wed Sep 23 13:22:53 EDT 2020): com.afwsamples.testdpc 0 0 10                                                                                       |
| FMT_SMF.1.1<br>Function 8 | restricting the sources of applications                                                                    | Enable/Disable     | SecurityLog<br><br>USER_RESTRICTION_ADDED: [<MDM agent> <no_install_unknown_sources/no_install_apps>]                                                                                            |
|                           | denying installation of applications                                                                       | Enable/Disable     | USER_RESTRICTION_REMOVED: [<MDM agent> <no_install_unknown_sources/no_install_apps>]                                                                                                             |

The table below provides audit events for the MDM Agent:

The following information pertains to the MDM Agent events:

- the clouddpc identifier may include additional naming depending on the version being used on the device (for example customers utilizing early access or testing versions may see additional information on the name of the application after the “clouddpc”)
- the MDM Agent is fixed to the address m.google.com, so the address may not be included in the log as this is the only possible address that can be used

| Requirement    | Auditable Events                         | Additional Audit Record Contents  | Log Events & Examples                                                                                                                                                                                        |
|----------------|------------------------------------------|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FAU_ALT_EXT.2  | Success or failure of sending an alert   |                                   | Logcat<br>cloudrpc: Event logged: ComplianceReportSent<br>cloudrpc: Event logged: ComplianceReportFailed                                                                                                     |
| FAU_GEN.1      | MDM policy updated or modified           |                                   | Logcat<br>cloudrpc: Applying policy: <Policy setting> value: <policy value><br>cloudrpc: Policy signature verification failed. Policy not applied.                                                           |
| FCS_TLSC_EXT.1 | Establishment/Termination of TLS session |                                   | Logcat<br>cloudrpc: NlapSSLsocket verified<br>cloudrpc: Closing socket, terminating connection to m.google.com<br>See FIA_X509_EXT.1 audit records for additional information.                               |
|                | Failure to verify identifier             |                                   | Logcat<br>cloudrpc: Failed to validate presented identifier: Host: m.google.com. App: com.google.android.apps.work.cloudrpc                                                                                  |
| FIA_ENR_EXT.2  | Enrollment                               | Success/Failure                   | Logcat<br>cloudrpc: Device registered successfully<br>cloudrpc: Setup failed. <error message>                                                                                                                |
| FIA_X509_EXT.1 | Failure to validate X.509v3 certificate. | Reason for failure of validation. | Logcat<br>Specifically for MDM Agent, this events will be logged in addition to the general events:<br>cloudrpc: Failed TLD wildcard check for {dnsName}<br>Note that the {dnsName} is fixed to m.google.com |
| FMT_POL_EXT.2  | Policy verification                      |                                   | Logcat<br>cloudrpc: Policy signature verification succeeded<br>cloudrpc: Policy signature verification failed. Policy not applied.                                                                           |
| FMT_SMF_EXT.4  | Key import                               | Success/Failure                   | See audit for FIA_ENR_EXT.2. Audits for successful enrollment imply the success of the import of the server public key which only happens at enrollment                                                      |
|                | Policy applied                           | Policy and values                 | Logcat<br>cloudrpc: Applying policy: <key> value: <value>                                                                                                                                                    |

| Requirement   | Auditable Events       | Additional Audit Record Contents | Log Events & Examples                                                                                                   |
|---------------|------------------------|----------------------------------|-------------------------------------------------------------------------------------------------------------------------|
|               | Lock Policy applied    |                                  | Logcat<br>clouddpc: Executing action: [Lock]<br>clouddpc: Successfully executed action: [Lock]                          |
| FMT_UNR_EXT.1 | Attempted unenrollment |                                  | As unenrollment is a factory reset, there is no specific log for a failure to unenroll, just a continuation of logging. |

## 10.FDP\_DAR\_EXT.2 & FCS\_CKM.2(2) – Sensitive Data Protection Overview

This section describes the architecture and cryptographic protocols used to protect Data-at-Rest (DAR) when the mobile device is in a Locked state. This implementation complies with the FDP\_DAR\_EXT.2 (Defines Data-at-Rest Protection) requirements, maintaining robust data protection in the background even when the device is locked.

To request access to the SDP File Encryption library, please reach out to: [niapsec@google.com](mailto:niapsec@google.com).

### 10.1. Architecture Overview

The core of the data protection mechanism is driven by the EncryptionManager and RawHybridKeyProvider classes. These modules integrate directly with the Android Keystore system and the OS-level DirectBoot mechanism to dynamically detect the device's lock state (Unlocked / Locked) and automatically switch encryption modes.

- **DEK (Data Encryption Key):** A symmetric key (AES-GCM 256-bit) used to encrypt the actual user data (files or strings).
- **KEK (Key Encryption Key):** A cryptographic key used to protect (wrap) the DEK. Depending on the device state, an asymmetric or symmetric approach is dynamically selected.
- **UDR (User Data Release) Key:** A hardware-protected symmetric key residing in the Android Keystore, accessible only when the device is in an "Unlocked" state.

This library enforces the following standard processes for the generation, derivation, and destruction of cryptographic keys:

1. **Key Establishment (Asymmetric Mode):** When the device is locked, the RawHybridKeyProvider generates an Ephemeral Key Pair using Elliptic Curve (EC P-521) and calculates a Shared Secret via the ECDH (Elliptic Curve Diffie-Hellman) protocol.
2. **Key Derivation:** The generated Shared Secret is not used directly. Instead, it is cryptographically derived into a secure 32-byte Asymmetric KEK using the HKDF (HMAC-based Extract-and-Expand Key Derivation Function) utilizing the SHA-256 hash algorithm.
3. **Key Destruction (Zeroization):** To prevent memory harvesting attacks, highly sensitive key materials (such as the plaintext DEK, ephemeral private keys, and the Shared Secret) are immediately zeroized (explicitly overwritten) from volatile memory using mechanisms like CleanSecretKeySpec after the encryption/decryption process completes or when a lock state transition occurs.

### 10.2. Dynamic State Transition and Re-wrapping (FDP\_DAR\_EXT.2.4)

The EncryptionManager embeds a state flag within the header of the encrypted data to manage the lifecycle of the DEK.

- **Locked State (0x01 - Asymmetric Mode):** Because the UDR Key is inaccessible when the device is locked, data is encrypted using a public key (decryption is not possible in this state).
- **Unlocked State (0x02 - Symmetric Mode):** Once unlocked, the UDR Key becomes accessible.

**Re-wrapping Process:** If data encrypted in the 0x01 (Asymmetric) mode is read while the device is in the Unlocked state, the library decrypts the data and immediately securely re-wraps the DEK using the UDR Key in 0x02 (Symmetric) mode, saving it back to storage. This significantly improves access efficiency and security for subsequent reads.

### 10.3. Implementation Guide and Sample Code

Application developers can implement secure Data-at-Rest protection through the EncryptionManager without needing to manage complex cryptographic algorithms or state transitions manually.

The following Kotlin sample code demonstrates the basic flow for encrypting and decrypting sensitive data. (To manage the lifecycle of the encryptionManager properly, please refer to the sample in the repository).

#### Code examples:

```
import android.content.Context
import com.android.niasec.encryption.api.EncryptionManager
import com.android.niasec.encryption.api.KeyProviderType
import java.io.File

class SecureStorageManager(private val context: Context) {

    // Utilizing RAW_HYBRID type automatically handles state
    // detection // and key zeroization (FCS_CKM.4) internally.
    private val encryptionManager = EncryptionManager(
        context = context.applicationContext,
        masterKeyUri = "android-keystore://my_app_udr_key",
        providerType = KeyProviderType.RAW_HYBRID,
        unlockedDeviceRequired = true
    )

    // Write and encrypt file with stream.
    fun saveSensitiveFile(targetFile: File, plaintext: ByteArray) {
        val outputStream =
            encryptionManager.encryptToFileStream(targetFile)
        outputStream.use { stream ->
            stream.write(plaintext)
        }

        plaintext.fill(0)
    }

    // Read and decrypt file with stream.
    fun readSensitiveFile(targetFile: File): ByteArray? {
        return try {
```

```

        val inputStream =
encryptionManager.decryptFromFileStream(targetFile)
        inputStream.use { stream ->
            stream.readBytes()
        }
    } catch (e: Exception) {
        // if the device is locked and the UDR Key cannot be
accessed.
        e.printStackTrace()
        null
    }
}

/**
 * Sweeps the application directory and re-wraps all files
 */
fun performPendingReWraps() {
    encryptionManager.sweepAndRewrapPendingFiles()
}

/**
 * Securely destroys the cryptographic instances
 */
fun destroy() {
    encryptionManager.destroy()
}
}

```

## 11.API Specification

This section provides a list of the evaluated cryptographic APIs that developers can use when writing their mobile applications.

1. Cryptographic APIs
  - This section lists all the APIs for the algorithms and random number generation
2. Key Management
  - APIs for importing, using, and destroying keys
3. Certificate Validation, TLS, HTTPS
  - API used by applications for configuring the reference identifier
  - APIs for validation checks (should match the test program provided)
  - TLS, HTTPS, Bluetooth BR/EDR (any other protocol available to applications)

### 11.1. Cryptographic APIs

Code samples to do encryption and decryption, including random number generation.

#### Code examples:

```
// Data to encrypt
byte[] clearText = "Secret Data".getBytes(StandardCharsets.UTF_8);

// Create a Biometric Support object to handle key authentication
BiometricSupport biometricSupport = new BiometricSupportImpl(activity,
getApplicationContext()) {
    ...
};

SecureCipher secureCipher = SecureCipher.getDefault(biometricSupport);
secureCipher.encryptSensitiveData("niapKey", clearText, new
SecureCipher.SecureSymmetricEncryptionCallback() {
    @Override
    public void encryptionComplete(byte[] cipherText, byte[] iv) {
        // Do something with the encrypted data
    }
});

// to decrypt
secureCipher.decryptSensitiveData("niapKey", cipherText, iv, new
SecureCipher.SecureDecryptionCallback() {
    @Override
    public void decryptionComplete(byte[] clearText) {
        // do something with the encrypted data
    }
});

// Generate ephemeral key (random number generation)
int keySize = 256;
SecureRandom secureRandom = SecureRandom.getInstanceStrong();
byte[] key = new byte[keySize / 8];
secureRandom.nextBytes(key);
```

```
// Encrypt / decrypt data with the ephemeral key
EphemeralSecretKey ephemeralSecretKey = new EphemeralSecretKey(key,
SecureConfig.getStrongConfig());
Pair<byte[], byte[]> ephemeralCipherText =
secureCipher.encryptEphemeralData(ephemeralSecretKey, clearText);
byte[] ephemeralClearText =
secureCipher.decryptEphemeralData(ephemeralSecretKey,
ephemeralCipherText.first, ephemeralCipherText.second);
```

### 11.1.1. SecureCipher

*Included in the NIAPSEC library.*

Handles low-level cryptographic operations including encryption and decryption. For sensitive data protection this library is not used directly by developers.

Supported Algorithms:

AES256 - AES/GCM/NoPadding

RSA3072 - RSA/ECB/OAEPWithSHA-256AndMGF1Padding

| Public Static Accessors |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SecureCipher            | <b>SecureCipher.getDefault</b> (BiometricSupport)<br><i>See BiometricSupport</i><br><br>API to get an instance of the SecureCipher with Biometric support.                                                                                                                                                                                                                                                                                                                                                                               |
| Public Methods          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| void                    | <b>encryptSensitiveData</b> (String keyAlias, byte[] clearData, SecureSymmetricEncryptionCallback callback)<br><br>Encrypt sensitive data using the <i>symmetric</i> algorithm specified by the default configuration, using NIAP standards.<br><i>See SecureConfig.getStrongConfig() - Default is AES256 GCM.</i><br><br>-keyAlias - Encrypt data with the AndroidKeyStore key referenced<br>-clearData - the data to be encrypted<br>-callback, the callback to return the cipherText after encryption is complete.                    |
| void                    | <b>encryptSensitiveDataAsymmetric</b> (String keyAlias, byte[] clearData, SecureAsymmetricEncryptionCallback callback)<br><br>Encrypt sensitive data using the <i>asymmetric</i> algorithm specified by the default configuration, using NIAP standards.<br><i>See SecureConfig.getStrongConfig() - Default is RSA3072 with OAEP.</i><br><br>-keyAlias - Encrypt data with the AndroidKeyStore key referenced<br>-clearData - the data to be encrypted<br>-callback, the callback to return the cipherText after encryption is complete. |

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pair<byte[], byte[]> | <p><b>encryptEphemeralData</b><br/>(EphemeralSecretKey ephemeralSecretKey, byte[] clearData)</p> <p>Encrypt data with an Ephemeral AES 256 GCM key, used for encrypting file data for SDP.</p> <ul style="list-style-type: none"> <li>-The Ephemeral key to use</li> <li>-clearData, the data to be encrypted</li> </ul> <p>Returns a Pair of the cipherText, and IV byte arrays respectively.</p>                                                                                                                                                                                                                                                                          |
| void                 | <p><b>decryptSensitiveData</b> (String keyAlias, byte[] encryptedData, byte[] initializationVector, SecureDecryptionCallback callback)</p> <p>Decrypt sensitive data using the <i>symmetric</i> algorithm specified by the default configuration, using NIAP standards.<br/><i>See SecureConfig.getStrongConfig() - Default is AES256 GCM.</i></p> <ul style="list-style-type: none"> <li>-keyAlias - Encrypt data with the AndroidKeyStore key referenced</li> <li>-encryptedData - the data to be decrypted</li> <li>-initializationVector - the IV used for encryption</li> <li>-callback, the callback to return the clearText after decryption is complete.</li> </ul> |
| void                 | <p><b>decryptSensitiveData</b> (String keyAlias, byte[] encryptedData, SecureDecryptionCallback callback)</p> <p>Decrypt sensitive data using the <i>asymmetric</i> algorithm specified by the default configuration, using NIAP standards.<br/><i>See SecureConfig.getStrongConfig() - Default is RSA3072 with OAEP.</i></p> <ul style="list-style-type: none"> <li>-keyAlias - Encrypt data with the AndroidKeyStore key referenced</li> <li>-encryptedData - the data to be decrypted</li> <li>-callback, the callback to return the clearText after decryption is complete.</li> </ul>                                                                                  |
| byte[]               | <p><b>decryptEphemeralData</b><br/>(EphemeralSecretKey ephemeralSecretKey, byte[] encryptedData, byte[] initializationVector)</p> <p>Decrypt data with an Ephemeral AES 256 GCM key, used for encrypting file data for SDP.</p> <ul style="list-style-type: none"> <li>-The Ephemeral key to use</li> <li>-encryptedData - the data to be decrypted</li> <li>-initializationVector - the IV used for encryption</li> </ul> <p>Returns a byte array of the clear text.</p>                                                                                                                                                                                                   |

**Built using the JCE libraries for more information please see the following resources:**

AndroidKeyStore - <https://developer.android.com/training/articles/keystore>

Cipher - <https://developer.android.com/reference/javax/crypto/Cipher>

SecretKey - <https://developer.android.com/reference/javax/crypto/SecretKey>

SecureRandom - <https://developer.android.com/reference/java/security/SecureRandom>  
BiometricPrompt - <https://developer.android.com/reference/android/hardware/biometrics/BiometricPrompt>

### 11.1.2. FCS\_CKM.2/UNLOCKED – Key Establishment (RSA)

Assume that Alice knows a private key and Bob knows Alice's public key. Bob sent a key encrypted by the public key. This example shows how Alice gets a plain key sent by Bob. Alice needs her own private key to decrypt an encrypted key.

```
KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA",
    "AndroidOpenSSL");
keyGen.initialize(keySize);
KeyPair keyPair = keyGen.generateKeyPair();
RSAPublicKey pub = (RSAPublicKey) keyPair.getPublic();
RSAPrivateCrtKey priv = (RSAPrivateCrtKey) keyPair.getPrivate();

// Encrypt
Cipher cipher = Cipher.getInstance("RSA/ECB/OAEPWithSHA-
    256AndMGF1Padding");
cipher.init(Cipher.ENCRYPT_MODE, publicKey, new OAEPParameterSpec("SHA-
    256",
        "MGF1", new MGF1ParameterSpec("SHA-1"),
        PSource.PSpecified.DEFAULT));
byte[] cipherText =
    cipher.doFinal(data.getBytes(StandardCharsets.UTF_8));

// Decrypt
Cipher cipher = Cipher.getInstance("RSA/ECB/OAEPWithSHA-
    256AndMGF1Padding");
cipher.init(Cipher.DECRYPT_MODE, privateKey, new
    OAEPParameterSpec("SHA-256",
        "MGF1", new MGF1ParameterSpec("SHA-1"),
        PSource.PSpecified.DEFAULT));
Byte[] plainText = cipher.doFinal(cipherText);
```

#### Algorithms::

**RSA/ECB/OAEPWithSHA-256AndMGF1Padding**

#### Reference:

Cipher - <https://developer.android.com/reference/javax/crypto/Cipher>

### 11.1.3. FCS\_CKM.2/UNLOCKED – Key Establishment (ECDSA) & FCS\_COP.1/SIGN – Signature Algorithms (ECDSA)

Assume that Alice knows a private key and Bob's public key. Bob knows his private key and Alice's public key. Then Alice and Bob can sign/verify the contents of a message.

```
KeyPairGenerator keyGen = KeyPairGenerator.getInstance("EC",
    "AndroidOpenSSL");
ECGenParameterSpec ecParams = new ECGenParameterSpec(spec);
```

```

keyGen.initialize(ecParams);
KeyPair keyPair = keyGen.generateKeyPair();
ECPrivateKey pubKey = (ECPrivateKey) keyPair.getPublic();
ECPrivateKey privKey = (ECPrivateKey) keyPair.getPrivate();

// Sign
Signature signature = Signature.getInstance(algorithm);
signature.initSign(privateKey);
signature.update(data.getBytes(StandardCharsets.UTF_8));
byte[] signature = signature.sign();

// Verify
Signature signature = Signature.getInstance(algorithm);
signature.initVerify(publicKey);
signature.update(data.getBytes(StandardCharsets.UTF_8));
boolean verified = signature.verify(sig);

```

#### Algorithms:

"SHA256withECDSA", "secp256r1"

"SHA384withECDSA", "secp384r1"

"SHA512withECDSA", "secp521r1"

#### Reference:

Signature - <https://developer.android.com/reference/java/security/Signature>

### 11.1.4. FCS\_CKM.1 – Key Generation (ECDSA)

Assume that Alice knows a private key and Bob's public key. Bob knows his private key and Alice's public key.

```

KeyPairGenerator keyGen = KeyPairGenerator.getInstance("EC",
"AndroidOpenSSL");
ECGenParameterSpec ecParams = new ECGenParameterSpec(spec);
keyGen.initialize(ecParams);
KeyPair keyPair = keyGen.generateKeyPair();
ECPrivateKey pubKey = (ECPrivateKey) keyPair.getPublic();
ECPrivateKey privKey = (ECPrivateKey) keyPair.getPrivate();

```

#### Algorithms:

"SHA256withECDSA", "secp256r1"

"SHA384withECDSA", "secp384r1"

"SHA512withECDSA", "secp521r1"

#### Reference:

Signature - <https://developer.android.com/reference/java/security/Signature>

### 11.1.5. FCS\_COP.1/ENCRYPT – Encryption/Decryption (AES)

Cipher class encrypts or decrypts a plain text.

```

KeyGenerator keyGenerator = KeyGenerator.getInstance("AES",
"AndroidOpenSSL");
keyGenerator.init(keySize);
secretKey key = keyGenerator.generateKey();

// Encrypt
Cipher cipher = Cipher.getInstance(transformation);
cipher.init(Cipher.ENCRYPT_MODE, secretKey);
byte[] iv = cipher.getIV();
byte[] clearData = data.getBytes(UTF_8);
byte[] cipherText = cipher.doFinal(clearData);
Pair<byte[], byte[]> result = Pair<>(cipherText, iv);

// Decrypt
Cipher cipher = Cipher.getInstance(transformation);
cipher.init(Cipher.DECRYPT_MODE, secretKey, spec);
String plainText = new String(cipher.doFinal(cipherText), UTF_8);

```

#### Algorithms:

AES/CBC/NoPadding

AES/GCM/NoPadding

#### Reference:

Cipher - <https://developer.android.com/reference/javax/crypto/Cipher>

### 11.1.6. FCS\_COP.1/HASH – Hashing (SHA)

You can use MessageDigest class to calculate the hash of plaintext.

```

MessageDigest messageDigest = MessageDigest.getInstance(algorithm);
messageDigest.update(data.getBytes(StandardCharsets.UTF_8));
byte[] digest = messageDigest.digest();

```

#### Algorithms:

SHA-1

SHA-256

SHA-384

SHA-512

#### Reference:

MessageDigest - <https://developer.android.com/reference/java/security/MessageDigest>

### 11.1.7. FCS\_COP.1/SIGN – RSA (Signature Algorithms)

KeyFactory class generates RSA private key and public key. Signature class signs a plaintext with private key generated above and verifies it with public key

```

KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA",
"AndroidOpenSSL");
keyGen.initialize(keySize);

```

```

KeyPair keyPair = keyGen.generateKeyPair();
RSAPublicKey pub = (RSAPublicKey) keyPair.getPublic();
RSAPrivateCrtKey priv = (RSAPrivateCrtKey) keyPair.getPrivate();

// Sign
Signature signature = Signature.getInstance(algorithm);
signature.initSign(privateKey);
signature.update(data.getBytes(StandardCharsets.UTF_8));
byte[] sig = signature.sign();

// Verify
Signature signature = Signature.getInstance(algorithm);
signature.initVerify(publicKey);
signature.update(data.getBytes(StandardCharsets.UTF_8));
boolean verified = signature.verify(sig);

```

#### Algorithms:

SHA256withRSA

SHA384withRSA

#### Reference:

Signature - <https://developer.android.com/reference/java/security/Signature>

#### 11.1.8. FCS\_CKM.1 –Key Generation (RSA)

KeyFactory class generates RSA private key and public key.

```

KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA",
"AndroidOpenSSL");
keyGen.initialize(keySize);
KeyPair keyPair = keyGen.generateKeyPair();
RSAPublicKey pub = (RSAPublicKey) keyPair.getPublic();
RSAPrivateCrtKey priv = (RSAPrivateCrtKey) keyPair.getPrivate();

```

#### Algorithms:

SHA256withRSA

SHA384withRSA

#### Reference:

Signature - <https://developer.android.com/reference/java/security/Signature>

#### 11.1.9. FCS\_COP.1/KEYHMAC - HMAC

Mac class calculates the hash of plaintext with key.

```

KeyGenerator keyGenerator = KeyGenerator.getInstance(
    algorithm, "AndroidOpenSSL");
keyGenerator.init(keySize);
secretKey key = keyGenerator.generateKey();

// Mac
Mac mac = Mac.getInstance(algorithm);

```

```
mac.init(secretKey);
byte[] mac = mac.doFinal(data.getBytes(StandardCharsets.UTF_8));
```

### Algorithms:

HmacSHA1  
HmacSHA256  
HmacSHA384  
HmacSHA512

### Reference:

Mac - <https://developer.android.com/reference/javax/crypto/Mac>

## 11.2. Key Management

Code samples to do key management.

### Code examples:

```
SecureKeyGenerator keyGenerator = SecureKeyGenerator.getInstance();
// Generate Keypair
keyGenerator.generateAsymmetricKeyPair(KEY_PAIR_ALIAS);
// Generate Symmetric Key
keyGenerator.generateKey(KEY_ALIAS);

// Generate ephemeral key (random number generation)
keyGenerator.generateEphemeralDataKey();

// To delete a key stored in the Android Keystore
KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
keyStore.load(null);
keyStore.deleteEntry("KEY_TO_REMOVE");
```

### 11.2.1. SecureKeyGenerator

*Included in the NIAPSEC library.*

Handles low-level key generation operations using the AndroidKeyStore. For sensitive data protection this library is not used directly by developers.

### Supported Algorithms:

AES256 - AES/GCM/NoPadding

RSA3072 - RSA/ECB/OAEPWithSHA-256AndMGF1Padding

| Public Static Accessors |                                                                                                        |
|-------------------------|--------------------------------------------------------------------------------------------------------|
| SecureKeyGenerator      | <b>SecureCipher.getDefault()</b><br><br>API to get an instance of the SecureCipher with NIAP settings. |
| Public Methods          |                                                                                                        |

|                    |                                                                                                                                                                                                                                                                                          |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| boolean            | <p><b>generateKey(String keyAlias)</b></p> <p>Generate an AES key with NIAP settings that is stored and protected in the AndroidKeyStore.</p> <p><i>See SecureConfig.getStrongConfig() - Default is AES256 GCM.</i></p> <p>-keyAlias - name for the key</p>                              |
| boolean            | <p><b>generateKeyAsymmetricKeyPair(String keyAlias)</b></p> <p>Generate an RSA key pair with NIAP settings that is stored and protected in the AndroidKeyStore.</p> <p><i>See SecureConfig.getStrongConfig() - Default is RSA3072 OAEP.</i></p> <p>-keyAlias - name for the key pair</p> |
| EphemeralSecretKey | <p><b>generateEphemeralDataKey()</b></p> <p>Generate an AES key with NIAP settings. This key is not stored in the AndroidKeyStore</p> <p>Uses SecureRandom.getInstanceStrong() to generate a random key.</p> <p><i>See SecureConfig.getStrongConfig() - Default is AES256 GCM.</i></p>   |

**Built using the JCE libraries for more information please see the following resources:**

AndroidKeyStore - <https://developer.android.com/training/articles/keystore>  
 KeyPairGenerator- <https://developer.android.com/reference/java/security/KeyPairGenerator>  
 SecretKey - <https://developer.android.com/reference/javax/crypto/SecretKey>  
 SecureRandom - <https://developer.android.com/reference/java/security/SecureRandom>  
 KeyGenParameterSpec - <https://developer.android.com/reference/android/security/keystore/KeyGenParameterSpec>

### 11.3. FCS\_TLSC\_EXT.1 - Certificate Validation, TLS, HTTPS

*Included in the NIAPSEC library.*

SecureURL automatically configures TLS and can perform certificate and host validation checking. At construction, SecureURL requires a reference identifier.

**Code examples:**

```
SecureURL url = new SecureURL(referenceIdentifier, "google_cert");
HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
conn.setRequestMethod("GET");
conn.setDoInput(true);
conn.connect();
```

```
// Manual check
SecureURL url = new SecureURL(referenceIdentifier, "google_cert");
boolean valid = url.isValid(urlConnection);
```

|                            |                                                                                                                                                                                   |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Public Constructors</b> |                                                                                                                                                                                   |
| SecureURL                  | <b>new SecureURL(String referenceIdentifier, String clientCert)</b><br><br>API to create an instance of the SecureURL with NIAP settings. clientCert is optional.                 |
| <b>Public Methods</b>      |                                                                                                                                                                                   |
| HttpsURLConnection         | openConnection<br><br>Opens an HttpsURLConnection using TLS by default and handles OCSP validation checks and does a hostname verification check on initiation of the connection. |
| boolean                    | isValid(String hostname, SSLSocket socket)<br><br>A manual OCSP certificate and hostname check.<br><br>Based on a hostname and underlying SSLSocket.                              |
| boolean                    | isValid(HttpsURLConnection conn)<br><br>A manual OCSP certificate and hostname check.<br><br>Based on an existing HttpsURLConnection.                                             |
| boolean                    | isValid(Certificate cert)<br><br>A manual OCSP certificate check.                                                                                                                 |
| boolean                    | isValid(List<Certificate> certs)<br><br>A manual OCSP certificates check.                                                                                                         |

**Built using the networking libraries for more information please see the following resources:**

HttpsURLConnection -

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

PKIXRevocationChecker -

<https://developer.android.com/reference/java/security/cert/PKIXRevocationChecker>

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

### 11.3.1. Cipher Suites

When applications utilize the NIAPSEC library, no configuration is needed to restrict or allow ciphersuites to be compliant. A list of the ciphersuites supported by Android 16 NIAPSEC can be found below:

For TLS 1.2 with mutual authentication:

| Approved Cipher Suites                                                                                                                                                                                                                                                                                                         | TLS Version |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,<br>TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,<br>TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,<br>TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,<br>TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289 | TLS v1.2    |

The device supports TLS versions 1.1, and 1.2 for use with EAP-TLS as part of WPA2 and WPA3. The TOE supports the following ciphersuites for this:

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

Additional ciphersuites are supported by the TOE for Wi-Fi networks but outside the specific scope of testing for the compliant configuration. It is expected that the Wi-Fi AP will be configured to utilize the proper ciphersuites to ensure compliance with the expected algorithms.

### 11.3.2. Guidance for Bluetooth Low Energy APIs

Provides classes that manage Bluetooth functionality, such as scanning for devices, connecting with devices, and managing data transfer between devices. The Bluetooth API supports both "Classic Bluetooth" and Bluetooth Low Energy.

For more information about Classic Bluetooth, see the [Bluetooth](#) guide. For more information about Bluetooth Low Energy, see the [Bluetooth Low Energy \(BLE\)](#) guide.

The Bluetooth APIs let applications:

- Scan for other Bluetooth devices (including BLE devices).
- Query the local Bluetooth adapter for paired Bluetooth devices.
- Establish RFCOMM channels/sockets.
- Connect to specified sockets on other devices.
- Transfer data to and from other devices.

- Communicate with BLE devices, such as proximity sensors, heart rate monitors, fitness devices, and so on.
- Act as a GATT client or a GATT server (BLE).

To perform Bluetooth communication using these APIs, an application must declare the [BLUETOOTH](#) permission. Some additional functionality, such as requesting device discovery, also requires the [BLUETOOTH\\_ADMIN](#) permission.

## Interfaces

|                                                  |                                                                                                                       |
|--------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| <a href="#">BluetoothAdapter.LeScanCallback</a>  | Callback interface used to deliver LE scan results.                                                                   |
| <a href="#">BluetoothProfile</a>                 | Public APIs for the Bluetooth Profiles.                                                                               |
| <a href="#">BluetoothProfile.ServiceListener</a> | An interface for notifying BluetoothProfile IPC clients when they have been connected or disconnected to the service. |

## Classes

|                                             |                                                                                                     |
|---------------------------------------------|-----------------------------------------------------------------------------------------------------|
| <a href="#">BluetoothA2dp</a>               | This class provides the public APIs to control the Bluetooth A2DP profile.                          |
| <a href="#">BluetoothAdapter</a>            | Represents the local device Bluetooth adapter.                                                      |
| <a href="#">BluetoothAssignedNumbers</a>    | Bluetooth Assigned Numbers.                                                                         |
| <a href="#">BluetoothClass</a>              | Represents a Bluetooth class, which describes general characteristics and capabilities of a device. |
| <a href="#">BluetoothClass.Device</a>       | Defines all device class constants.                                                                 |
| <a href="#">BluetoothClass.Device.Major</a> | Defines all major device class constants.                                                           |
| <a href="#">BluetoothClass.Service</a>      | Defines all service class constants.                                                                |
| <a href="#">BluetoothDevice</a>             | Represents a remote Bluetooth device.                                                               |

|                                                 |                                                                                                                                                                                                                                                                                                                                            |
|-------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">BluetoothGatt</a>                   | Public API for the Bluetooth GATT Profile.                                                                                                                                                                                                                                                                                                 |
| <a href="#">BluetoothGattCallback</a>           | This abstract class is used to implement <a href="#">BluetoothGatt</a> callbacks.                                                                                                                                                                                                                                                          |
| <a href="#">BluetoothGattCharacteristic</a>     | Represents a Bluetooth GATT Characteristic<br><br>A GATT characteristic is a basic data element used to construct a GATT service, <a href="#">BluetoothGattService</a> .                                                                                                                                                                   |
| <a href="#">BluetoothGattDescriptor</a>         | Represents a Bluetooth GATT Descriptor<br><br>GATT Descriptors contain additional information and attributes of a GATT characteristic, <a href="#">BluetoothGattCharacteristic</a> .                                                                                                                                                       |
| <a href="#">BluetoothGattServer</a>             | Public API for the Bluetooth GATT Profile server role.                                                                                                                                                                                                                                                                                     |
| <a href="#">BluetoothGattServerCallback</a>     | This abstract class is used to implement <a href="#">BluetoothGattServer</a> callbacks.                                                                                                                                                                                                                                                    |
| <a href="#">BluetoothGattService</a>            | Represents a Bluetooth GATT Service<br><br>Gatt Service contains a collection of <a href="#">BluetoothGattCharacteristic</a> , as well as referenced services.                                                                                                                                                                             |
| <a href="#">BluetoothHeadset</a>                | Public API for controlling the Bluetooth Headset Service.                                                                                                                                                                                                                                                                                  |
| <a href="#">BluetoothHealth</a>                 | <i>This class was deprecated in API level 29. Health Device Profile (HDP) and MCAP protocol are no longer used. New apps should use Bluetooth Low Energy based solutions such as <a href="#">BluetoothGatt</a>, <a href="#">BluetoothAdapter#listenUsingL2capChannel()</a>, or <a href="#">BluetoothDevice#createL2capChannel(int)</a></i> |
| <a href="#">BluetoothHealthAppConfiguration</a> | <i>This class was deprecated in API level 29. Health Device Profile (HDP) and MCAP protocol are no longer used. New apps should use Bluetooth Low Energy based solutions such as <a href="#">BluetoothGatt</a>, <a href="#">BluetoothAdapter#listenUsingL2capChannel()</a>, or <a href="#">BluetoothDevice#createL2capChannel(int)</a></i> |

|                                                  |                                                                                                                                                                                                                                                                                                                                            |
|--------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">BluetoothHealthCallback</a>          | <i>This class was deprecated in API level 29. Health Device Profile (HDP) and MCAP protocol are no longer used. New apps should use Bluetooth Low Energy based solutions such as <a href="#">BluetoothGatt</a>, <a href="#">BluetoothAdapter#listenUsingL2capChannel()</a>, or <a href="#">BluetoothDevice#createL2capChannel(int)</a></i> |
| <a href="#">BluetoothHearingAid</a>              | This class provides the public APIs to control the Hearing Aid profile.                                                                                                                                                                                                                                                                    |
| <a href="#">BluetoothHidDevice</a>               | Provides the public APIs to control the Bluetooth HID Device profile.                                                                                                                                                                                                                                                                      |
| <a href="#">BluetoothHidDevice.Callback</a>      | The template class that applications use to call callback functions on events from the HID host.                                                                                                                                                                                                                                           |
| <a href="#">BluetoothHidDeviceAppQosSettings</a> | Represents the Quality of Service (QoS) settings for a Bluetooth HID Device application.                                                                                                                                                                                                                                                   |
| <a href="#">BluetoothHidDeviceAppSdpSettings</a> | Represents the Service Discovery Protocol (SDP) settings for a Bluetooth HID Device application.                                                                                                                                                                                                                                           |
| <a href="#">BluetoothManager</a>                 | High level manager used to obtain an instance of an <a href="#">BluetoothAdapter</a> and to conduct overall Bluetooth Management.                                                                                                                                                                                                          |
| <a href="#">BluetoothServerSocket</a>            | A listening Bluetooth socket.                                                                                                                                                                                                                                                                                                              |
| <a href="#">BluetoothSocket</a>                  | A connected or connecting Bluetooth socket.                                                                                                                                                                                                                                                                                                |

<https://developer.android.com/reference/android/bluetooth/package-summary.html>

How to connect and pair with a bluetooth device:

```
// get bluetooth adapter
BluetoothAdapter bluetoothAdapter =
BluetoothAdapter.getDefaultAdapter();
if (bluetoothAdapter == null) {
    // Device doesn't support Bluetooth
}

// make sure bluetooth is enabled
if (!bluetoothAdapter.isEnabled()) {
```

```

        Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
    }

    // query for devices
    Set<BluetoothDevice> pairedDevices =
    bluetoothAdapter.getBondedDevices();
    if (pairedDevices.size() > 0) {
        // There are paired devices. Get the name and address of each
        paired device.
        for (BluetoothDevice device : pairedDevices) {
            String deviceName = device.getName();
            String deviceHardwareAddress = device.getAddress(); // MAC
            address
        }
    }

    // Connect to devices.
    private class AcceptThread extends Thread {
        private final BluetoothServerSocket mmServerSocket;

        public AcceptThread() {
            // Use a temporary object that is later assigned to
            mmServerSocket
            // because mmServerSocket is final.
            BluetoothServerSocket tmp = null;
            try {
                // MY_UUID is the app's UUID string, also used by the
                client code.
                tmp =
                bluetoothAdapter.listenUsingRfcommWithServiceRecord(NAME, MY_UUID);
            } catch (IOException e) {
                Log.e(TAG, "Socket's listen() method failed", e);
            }
            mmServerSocket = tmp;
        }

        public void run() {
            BluetoothSocket socket = null;
            // Keep listening until exception occurs or a socket is
            returned.
            while (true) {
                try {
                    socket = mmServerSocket.accept();
                } catch (IOException e) {
                    Log.e(TAG, "Socket's accept() method failed", e);
                    break;
                }

                if (socket != null) {
                    // A connection was accepted. Perform work associated
                    with
                    // the connection in a separate thread.
                    manageMyConnectedSocket(socket);
                    mmServerSocket.close();
                }
            }
        }
    }

```

```

        break;
    }
}

// Closes the connect socket and causes the thread to finish.
public void cancel() {
    try {
        mmServerSocket.close();
    } catch (IOException e) {
        Log.e(TAG, "Could not close the connect socket", e);
    }
}
}

```

More information here

<https://developer.android.com/guide/topics/connectivity/bluetooth.html#SettingUp>

Sample service to interact with a bluetooth APIs.

*// A service that interacts with the BLE device via the Android BLE API.*

```

public class BLEService extends Service {
    private final static String TAG = "BLEService";
    private BluetoothManager mBluetoothManager;
    private BluetoothAdapter mBluetoothAdapter;
    private String mBluetoothDeviceAddress;
    private BluetoothGatt mBluetoothGatt;
    private int mConnectionState = STATE_DISCONNECTED;
    private static final int STATE_DISCONNECTED = 0;
    private static final int STATE_CONNECTING = 1;
    private static final int STATE_CONNECTED = 2;
    public final static String ACTION_GATT_CONNECTED =
        "com.niap.ble.ACTION_GATT_CONNECTED";
    public final static String ACTION_GATT_DISCONNECTED =
        "com.niap.ble.ACTION_GATT_DISCONNECTED";
    public final static String ACTION_GATT_SERVICES_DISCOVERED =
        "com.niap.ble.ACTION_GATT_SERVICES_DISCOVERED";
    public final static String ACTION_DATA_AVAILABLE =
        "com.niap.ble.ACTION_DATA_AVAILABLE";
    public final static String EXTRA_DATA =
        "com.niap.ble.EXTRA_DATA";

    // Various callback methods defined by the BLE API.
    private final BluetoothGattCallback mGattCallback =
        new BluetoothGattCallback() {
            @Override
            public void onConnectionStateChange(BluetoothGatt gatt,
int status,
int newState) {
                String intentAction;
                if (newState == BluetoothProfile.STATE_CONNECTED) {
                    intentAction = ACTION_GATT_CONNECTED;
                    mConnectionState = STATE_CONNECTED;
                    broadcastUpdate(intentAction);
                    Log.i(TAG, "Connected to GATT server.");

```

```

        Log.i(TAG, "Attempting to start service
discovery:" +
        mBluetoothGatt.discoverServices());
    } else if (newState ==
BluetoothProfile.STATE_DISCONNECTED) {
        intentAction = ACTION_GATT_DISCONNECTED;
        mConnectionState = STATE_DISCONNECTED;
        Log.i(TAG, "Disconnected from GATT server.");
        broadcastUpdate(intentAction);
    }
}

@Override
// New services discovered
public void onServicesDiscovered(BluetoothGatt gatt, int
status) {
    if (status == BluetoothGatt.GATT_SUCCESS) {
        broadcastUpdate(ACTION_GATT_SERVICES_DISCOVERED)
;
    } else {
        Log.w(TAG, "onServicesDiscovered received: " +
status);
    }
}

@Override
// Result of a characteristic read operation
public void onCharacteristicRead(BluetoothGatt gatt,
BluetoothGattCharacteri
stic characteristic,
                                int status) {
    if (status == BluetoothGatt.GATT_SUCCESS) {
        broadcastUpdate(ACTION_DATA_AVAILABLE,
characteristic);
    }
}
};
}

```