Evidence of Compliance to the ETSI TS 103 732-1/TS 103 732-2 and GSMA Requirements

Overview

The general purpose of this document is to provide a natural language translation of the Common Criteria requirements that can be more readily understood. The requirements here are a combination of Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs).

SFRs can generally be considered testable requirements on the device, though this is not always the case. In cases where it is not explicitly testable, generally evidence of meeting the requirement is provided, such as source code or pointing to another evaluation which proves the requirement is met. SFRs are always focused on the device itself, either a software or hardware component of the device which supports a security requirement.

SARs can generally be considered documentation requirements and may be focused on the device, on the production of the device (manufacturing) or in-market support. There may be some device testing done where it is possible to prove a requirement with a test (instead of documentation), but the majority of SARs will not require testing (this may change over time).

Beyond the TS 103 732 requirements, there are additional areas that require documentation to support the security evaluation of the device. These are listed after the TS 103 732 requirements.

How to complete the questionnaire:

<table>
<tr><td rowspan="4">TS 103 732 SFR, SAR or GSMA requirement</td><td>Requirement description</td><td rowspan="2">Supported? (Y/N)</td></tr>
<tr><td>Evidence required for submission</td></tr>
<tr><td>SFR from TS 103 732 or GSMA to be filled out</td><td rowspan="2">Y or N</td></tr>
<tr><td>OEM response goes here, depending on evidence requirement either provide a description, documentation or a reference to supporting material to be included with submission of the completed questionnaire.</td></tr>
</table>

Several items will need to be submitted along with the completed questionnaire. These may include, but not limited to:

- Process, policy, procedure, architectural and design documentation
- Source code listings, samples, and repositories

- Configuration data such as kernel defconfigs, bootloader configuration, SELinux policies
- Assessment reports for items such as privileged applications, OTA update services
- Devices configured as production, as well as userdebug, and any custom tools required for loading software

When answering the questionnaire, the set of devices being considered should be limited to those released in the last 12 months or with an impending release. Where appropriate, general responses covering all devices may be provided; device-specific responses should not be needed unless requirement implementation differs significantly.

The ETSI TS 103 732-1 questionnaire sections are grouped as they are in the Protection Profile:

- Cryptographic Support
- User Data Protection
- Identification and Authentication
- Security Management
- Privacy
- Protection of the TSF
- Trusted Path/Channels
-

The ETSI TS 103 732-2 sections:

- Identification and Authentication

The ETSI TS 103 732-4 sections:

- Applications
- Application Risk

The ETSI TS 103 732-5 sections:

- Bootloader - User Data Protection
- Bootloader - Protection of the TSF
- Bootloader - Life Cycle
- Root of Trust - Protection of the TSF

The GSMA FS.56 sections:

- Cryptographic Support
- Identification and Authentication
- Privacy
- Protection of the TSF
- Live Cycle Requirements

Devices for Certification

The following tables are for specifying the devices that will be certified. Add rows as necessary for each separate device covered in the evaluation.

Evaluated Devices

Device	Operating System	OS Version	Kernel Version
TB522FU	Android	17	6.6

Device	Model(s)	CPU Architecture	CPU
TB522FU	TB522FU	ARM64	Qualcomm SM 8750P

Equivalent Devices

The devices here are claimed by the developer as equivalent to an evaluated device.

Claimed Device	Evaluated Device	Differences between devices
-	-	-

TS 103 732-1 SFRs

Cryptographic Support

This section is focused on the cryptography that is used in the system.This includes both key lifecycles and the cryptographic algorithms that are in use.

The requirements for cryptographic algorithms are open as long as the algorithm is able to meet the requirements set by ISO for adding the algorithm to the ISO standard. Note that this does not mean that the algorithm shall be submitted to ISO for inclusion in their standards, only that it meets those requirements. This ensures that the algorithm has been publicly available and reviewed, and so is considered acceptable for use to meet the security claims of this evaluation.

FCS_RNG_EXT.1	Devices shall provide at least one random number generator (RNG) that produces uniformly-distributed, unpredictable output.	Supported? (Y/N)
	For each RNG on the device, provide a description of the type (such as physical or software) and how the amount of entropy provided has been verified. Common examples of proof would be NIST 800-90B or AIS 31 analysis.	
	FCS_RNG_EXT.1.1 The TSF shall provide a [selection: *Pseudo*] random number generator. **FCS_RNG_EXT.1.2** The TSF shall provide random numbers that meet [assignment: *FIPS SP 800-90B*].	***Y***
	SoC platform provided the PRNG that comply with SP800-90B, as listed: E207 The Android DRBGs are able to use the output from /dev/hw_random for seeding. As this device is normally restricted to only kernel processes, Android provides an entropy daemon that uses the BoringSSL AES CTR_DRBG to provide a constant source of entropy to user-space services (mainly BoringSSL itself). The DRBG in the entropy daemon is reseeded every 200 requests (well below the NIST 800-90A requirements for reseeding an approved DRBG function).	

FCS_CKM.1/Asymmetric	Devices shall generate asymmetric keys properly that can meet at least the equivalent of 128-bit symmetric encryption strength.	Supported? (Y/N)
	The process for generating asymmetric keys on the device shall be explained. This will be specific to the algorithm(s) in use.	

	FCS_CKM.1.1/Asymmetric The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [assignment:RSA] and specified cryptographic key sizes [assignment: 3072 bits, 4096 bits] that meet the following: [assignment: FIPS 186-4].	
	BoringSSL supports generating RSA and ECDSA keys compliant with NIST FIPS 186-4 that are equivalent (or greater) than 128-bit symmetric strength (RSA 3072 and ECC 256). BoringSSL supports generating keys that are of 128-bit strength and higher for both RSA and ECDSA. In both cases the DRBG from FCS_RNG_EXT.1 is used in the process.	***Y***

FCS_CKM.1 /Symmetric	Devices shall generate symmetric keys of at least 128-bit length either by generating the key using a RNG or by a derivation function.	Supported? (Y/N)
	The process for generating the keys on the device shall be explained. For example, the key is generated by calling the RNG.	
	FCS_CKM.1.1/Symmetric The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [~~assignment: cryptographic key generation algorithm~~ selection: using a random number generator as specified in FCS_RNG_EXT.1, a combination of precursor key material using a derivation function as specified in FCS_COP.1/Derivation] and specified cryptographic key sizes [assignment: *256 bits*] ~~that meet the following: [**assignment**: *list of standards*~~].	***Y***
	BoringSSL supports generating symmetric (AES) keys compliant with NIST FIPS 197 that are equivalent (or greater) than 128-bit symmetric strength. In both cases the DRBG from FCS_RNG_EXT.1 is used in the process.	

FCS_COP.1 Note

All the algorithms listed under FCS_COP.1 may have multiple implementations throughout the system. For example, symmetric encryption is likely to be available in low level hardware (i.e. SoC), a hardware storage encryption engine, a hardware-backed key store (if support is provided), the SEE, the kernel and within the main OS itself. Not all algorithms may be available in all components, and different configurations may provide different options.

The expectation for this section is that each instance of an algorithm type be specified if it is key to providing security services for the device (i.e. if an algorithm is available for user-installed applications or is not used by the security components of the device itself, they do not need to be listed).

FCS_COP.1 /SigGen	Devices shall support an asymmetric algorithm that is used to verify the signature of update packages (both for the OS and applications). Multiple algorithms may be supported for different purposes.	Supported? (Y/N)
	Each supported algorithm shall be listed along with the key sizes supported. The listing should point to the standard used to implement the algorithm (for example FIPS 186-4 for RSA).	
	FCS_COP.1.1/SigGen The TSF shall perform [*CRYPTOGRAPHIC SIGNATURE SERVICES (GENERATION AND VERIFICATION)*] in accordance with a specified cryptographic algorithm [assignment: *RSA, ECDSA*] and cryptographic key sizes [assignment: 3072 bits, 4096 bits for RSA, 256 and 384 bits for ECDSA] that meet the following: [assignment: *FIPS 186-5*].	Y
	BoringSSL and TEE supports generating RSA and ECDSA keys compliant with NIST FIPS 186-5 that are equivalent (or greater)	

FCS_COP.1 /KeyEst	Devices shall support a key establishment algorithm for the encryption/decryption of user data. This shall list the algorithm used for user data encryption in transit, and algorithms used in other parts of the system.	Supported? (Y/N)
	Each supported algorithm shall be listed along with the key sizes supported. The listing should point to the standard used to implement the algorithm (for example FIPS 197 for AES).	
	FCS_COP.1.1/Symmetric The TSF shall perform [*CRYPTOGRAPHIC KEY ESTABLISHMENT*] in accordance with a specified cryptographic algorithm [assignment: *cryptographic Elliptic curve-based KEY ESTABLISHMENT SCHEME algorithm*] and cryptographic key sizes [assignment: *256 bits*] that meet the following: [assignment: AES as in FIPS PUB 197 with CBC (SP800-38A), CCM (SP800-38C), Key Wrap (SP800-38F),GCM (SP800-38D), XTS (SP800-38E) and GCM (SP800-38D) modes].	Y
	To support TLS communications, BoringSSL supports ECDH key pair generation	

FCS_COP.1 /Symmetric	Devices shall support a symmetric algorithm for the encryption/decryption of user data. This shall list the algorithm used for user data encryption, and algorithms used in other parts of the system.	Supported? (Y/N)
	Each supported algorithm shall be listed along with the key sizes supported. The listing should point to the standard used to implement the algorithm (for example FIPS 197 for AES).	
	FCS_COP.1.1/Symmetric The TSF shall perform [*SYMMETRIC ENCRYPTION AND DECRYPTION*] in accordance with a specified cryptographic algorithm [assignment: *AES*] and cryptographic key sizes [assignment: *256 bits*] that meet the following: [assignment: AES as defined in FIPS PUB 197 with CBC (SP800-38A), CCM (SP800-38C), Key Wrap Y (SP800-38F), GCM (SP800-38D), XTS (SP800-38E) and GCM (SP800-38D) modes].	Y
	Multiple modules provide symmetric encryption using AES. These algorithms are all verified as part of the cryptographic test that are designed with NIST regulation.	

FCS_COP.1 /Derivation	Devices shall support a key derivation function.	Supported? (Y/N)
	Each supported function shall be listed along with the key sizes supported. The listing should point to the standard used to implement the algorithm (for example NIST SP 800-108 for KBKDF or NIST SP 800-132 for PBKDF2).	
	FCS_COP.1.1/Derivation The TSF shall perform [*DERIVATION FUNCTION*] in accordance with a specified cryptographic algorithm [assignment: HMAC-SHA-256] and cryptographic key sizes [assignment: *cryptographic key sizes 256 bits*] that meet the following: [assignment: RFC 5869].	Y
	To derive a key from user input (password/PIN/pattern), the user input (and salt) is processed using scrypt (which contains a PBKDF2 operation, followed by a series of ROMix operations and then one final PBKDF2). This key is then combined with a hardware-fused key using a KDF function inside the TEE (the output key is not used on its own, only in combination with the hardware-fused key).	

FCS_COP.1 /Hash	Devices shall support a cryptographic hash algorithm.	Supported? (Y/N)
	Each supported algorithm shall be listed along with the key sizes supported. The listing should point to the standard used to implement the algorithm (for example FIPS 180-4 for SHA).	
	FCS_COP.1.1/Hash The TSF shall perform [*CRYPTOGRAPHIC HASHING*] in accordance with a specified cryptographic algorithm [assignment: *SHA-256 and 384*] and cryptographic key sizes [*NONE*] that meet the following: [assignment: FIPS 180-4].	Y
	These algorithms are all verified as part of the cryptographic test that are designed with NIST regulation.	

FCS_COP.1 /KeyedHash	Devices shall support a message authentication code algorithm.	Supported? (Y/N)
	Each supported algorithm shall be listed along with the key sizes supported. The listing should point to the standard used to implement the algorithm (for example FIPS 198-1 for HMAC).	
	FCS_COP.1.1/KeyedHash The TSF shall perform [*KEYED-HASH MESSAGE AUTHENTICATION*] in accordance with a specified cryptographic algorithm [assignment: SHA] and cryptographic key sizes [assignment: *256,384 bits*] that meet the following: [assignment: FIPS 180-4].	Y
	These algorithms are all verified as part of the cryptographic test that are designed with NIST regulation	

FCS_CKH_ EXT.1/Low	Devices shall provide encrypted storage that is not tied to the user credentials. This can require separate apps to explicitly implement the functionality, but it shall be available in the device.	Supported? (Y/N)
	Documentation shall provide: ● Description of how DE keys are generated/derived ○ Algorithms, key lengths used in the process ● Description of how DE keys are cryptographically tied to hardware-backed keystore	
	FCS_CKH_EXT.1.1/Low The TSF shall support a key hierarchy for data encryption keys to protect [*LOW USER DATA ASSETS*]. **FCS_CKH_EXT.1.2/Low** The TSF shall ensure that all keys in the key hierarchy are derived and/or generated according to [assignment: file encryption keys are generated using AES_CTR DRBG from BoringSSL and are encrypted using a key that is	Y

	derived from the DUK] ensuring that the key hierarchy uses the DUK and [NO USER CREDENTIALS] directly or indirectly in the derivation of the data encryption key(s) for [*LOW USER DATA ASSETS*]. **FCS_CKH_EXT.1.3/Low** The TSF shall ensure that all keys in the key hierarchy and all data used in deriving the keys in the hierarchy are protected according to [assignment: no other *rules*].	
	https://source.android.com/docs/security/features/encryption/file based https://developer.android.com/training/articles/direct-boot.html Android provides a key hierarchy tied to the device root key that allows some data to be encrypted by access before the user has successfully authenticated (such as phone calls or alarms). This follows the credentialed encryption key hierarchy but uses a hardcoded password instead of the user's password (after that all data stored for DE storage is handled the same, just using the separate key hierarchy to handle the encryption keys). Use of this requires the app to explicitly take advantage of the DE storage capability	

FCS_CKH_EXT.1/MediumHigh	Devices shall provide encrypted storage that is tied to the user credentials. By default all user data shall be decrypted after the user authenticates the first time (device boot). Additionally the device shall provide the functionality to keep user data encrypted once the device has been unlocked but the user is not active (the user logged into the device at start-up and then the device has since been screen locked and requires the user to provide credentials again). This additional functionally can require separate apps to explicitly implement the functionality, but it shall be available in the device.	Supported? (Y/N)
	Documentation shall provide: ● Description of how CE keys are generated/derived ○ Algorithms, key lengths used in the process ● Description of how CE keys are cryptographically tied to hardware-backed keystore ● Description of how CE keys are cryptographically tangled with the user's credentials	
	FCS_CKH_EXT.1.1/MediumHigh The TSF shall support a key hierarchy for data encryption keys to protect [*MEDIUM AND HIGH USER DATA ASSETS*].	Y

	FCS_CKH_EXT.1.2/MediumHigh The TSF shall ensure that all keys in the key hierarchy are derived and/or generated according to [assignment: *file encryption keys are generated using AES_CTR DRBG from BoringSSL and are encrypted using a key that is derived from the DUK*] ensuring that the key hierarchy uses the DUK and [THE USER CREDENTIALS] directly or indirectly in the derivation of the data encryption key(s) for [*MEDIUM AND HIGH USER DATA ASSETS*]. **FCS_CKH_EXT.1.3/MediumHigh** The TSF shall ensure that all keys in the key hierarchy and all data used in deriving the keys in the hierarchy are protected according to [assignment: *no other rules*].	
	MEDIUM https://source.android.com/docs/security/features/encryption/file-based https://developer.android.com/training/articles/direct-boot.html Android provides a key hierarchy tied to both the user's password and the device root key that protects all data by default such that it is not available until after a successful authentication from the user. This is called Credentialed Encryption (CE). The CE key hierarchy binds the master encryption key to both the user's credential and the root key from the device. If an app does not do anything specific to store data, it will be stored as CE data. HIGH Android provides the ability for an app to be aware of the lock state of the device. This can be obtained through APIs: https://developer.android.com/reference/android/security/keystore/KeyGenParameterSpec.Builder#setUnlockedDeviceRequired(boolean) https://developer.android.com/reference/android/security/keystore/KeyGenParameterSpec.Builder#setUserAuthenticationRequired(boolean) When combined with the app being able to individually encrypt its own files: https://developer.android.com/topic/security/data#java An app can be developed that can maintain that its data is only able to be decrypted for use when the user is active on the device (the device is considered unlocked).	

	This is an app-specific capability, and so requires an app to be developed to utilize this capability.	

FCS_CKM.4	Devices shall clear plain-text keys when no longer needed.	Supported? (Y/N)
	Describe how keys are cleared from memory (both volatile and non-volatile). Note that some of these may not be applicable in all devices (for example there may be no non-volatile flash memory that is not wear-leveled), in which case it is sufficient to specify it is not applicable.	
	FCS_CKM.4.1 The TSF shall destroy **keys from the key hierarchy for Low, Medium, and High** cryptographic keys in accordance with a specified cryptographic key destruction method [~~assignment:~~ • *FOR VOLATILE MEMORY, BY A SINGLE DIRECT OVERWRITE CONSISTING OF [selection: ZEROES];* • *FOR NON-VOLATILE EEPROM, BY A SINGLE DIRECT OVERWRITE CONSISTING OF A RANDOM PATTERN, USING THE TSF'S RNG, FOLLOWED BY A READ-VERIFY;* • *FOR NON-VOLATILE FLASH MEMORY, THAT IS NOT WEAR-LEVELLED, BY [selection: , A BLOCK ERASE THAT ERASES THE REFERENCE TO MEMORY THAT STORES DATA AS WELL AS THE DATA ITSELF];* • *FOR NON-VOLATILE FLASH MEMORY, THAT IS WEAR-LEVELLED, BY [selection: A BLOCK ERASE];* • *FOR NON-VOLATILE MEMORY OTHER THAN EEPROM AND FLASH, BY A SINGLE DIRECT OVERWRITE WITH A RANDOM PATTERN THAT IS CHANGED BEFORE EACH WRITE*]; that meets the following: [assignment: *list of standards*].	Y
	The TOE clears sensitive cryptographic material (plaintext keys authentication data, and other security parameters) from memory when no longer needed. Public keys can remain in memory when the phone is locked, but all crypto-related private keys are evicted from memory upon device lock. No plain text cryptographic material resides in the TOE's Flash as the TOE encrypts all keys stored in Flash. When performing a full wipe of	

	protected data, the TOE cryptographically erases the protected data by clearing the Data-At-Rest DEK. Because the Android Keystore of the TOE resides within the user data partition, the TOE effectively cryptographically erases those keys when clearing the Data-At-Rest DEK. In turn, the TOE clears the DataAt-Rest DEK and Secure Key Storage SEK through a secure direct overwrite (BLKSECDISCARD ioctl) of the wear-leveled Flash memory containing the key followed by a read-verify	

User Data Protection

FDP_ACC.1/APP_Update **FDP_ACF.1/APP_Update** **FDP_UPF_EXT.1/APP_Update**	All applications delivered via the app store (distribution platform) shall provide a means for applications to be updated.	Supported? (Y/N)
	Document the following information about the update process: ● Frequency of automatic checking with the app store ● Method for verifying new updates are available (e.g. versioning such that older versions cannot be installed as an update) ● Method for verifying validity of the package (i.e. signature checks)	
	FDP_ACC.1.1/APP_Update The TSF shall enforce the [*APP_UPDATE POLICY*] on [*SUBJECTS: THE TSF, OBJECTS: APP, APP_UPDATE_PACKAGE, OPERATIONS: UPDATE_APP*]. **FDP_ACF.1.1/APP_Update** The TSF shall enforce the [*APP_UPDATE POLICY*] to objects based on the following: [*SUBJECTS: THE TSF, OBJECTS[ATTRIBUTES]: APP[VERSION_ID, SIGNATURE], APP_UPDATE_PACKAGE[VERSION_ID, SIGNATURE, PACKAGE_SOURCE]*]. **FDP_ACF.1.2/APP_Update** The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [● *THE TSF SHALL ALLOW THE TSF TO UPDATE_APP WITH AN APP_UPDATE_PACKAGE ONLY IF:* ○ *THE TSF SUCCESSFULLY VERIFIES THE SIGNATURE OF THE APP_UPDATE_PACKAGE AND THE SIGNATURE IS FROM THE SAME APP OR APP DEVELOPER; AND* ○ *[selection: THE VERSION_ID OF THE APP_UPDATE_PACKAGE IS NOT LOWER THAN THE VERSION_ID OF THE INSTALLED APP];* ● *THE TSF SHALL UPDATE_APP IN AS AN ATOMIC UPDATE FUNCTION*]. **FDP_ACF.1.3/APP_Update** The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [assignment : no other rules]. **FDP_ACF.1.4/APP_Update** The TSF shall explicitly deny access of subjects to objects based on the following additional rules: [*THE TSF SHALL NOT ALLOW ANY TSF-MEDIATED*	Y

	ACTIONS RELATED TO THE UPDATE_APP OPERATION OR ACCESS TO THE APP DURING ITS UPDATING]. **FDP_UPF_EXT.1.1/APP_Update** The TSF shall be able to check for a [APP] update package every [assignment: ~~*interval period*~~ *Once a day*].	
	The Play Store checks for app updates on a daily basis. Based on the user settings, the apps will be automatically updated when the device is not in use or the user will be notified that updates are available. The user can force a check for updates (or when notified start the update immediately). When an app update is available, it will only be installed if the signature is from the same app developer as the currently installed version and that the version ID is not lower than the currently installed version of the app. When an app is being updated, if the app is currently running, the instance will be closed during the update process. If the update fails for any reason, the installation process will roll back to the version that existed at the time of the download	

FDP_ACC.2 /SSW_Update **FDP_ACF.1/ SSW_Update** **FDP_UPF_ EXT.1/SSW _Update**	The device shall support system software updates (i.e. OTA updates) and secure how they are downloaded and installed.	Supported? (Y/N)
	Document the following information about the update process: ● Frequency of automatic checking with the update location ● Method for verifying new updates are available (e.g. versioning such that older versions cannot be installed as an update) ● Method for verifying validity of the package (i.e. signature checks) ● How security is maintained during the update process (i.e. that the system cannot be circumvented during the update process) ***NOTE: The yellow highlight is a modification from GSMA FS.56.***	
	FDP_ACC.2.1/SSW_Update The TSF shall enforce the [*SSW_UPDATE POLICY*] on [*SUBJECTS: THE TSF, OBJECTS: THE SSW, SSW_UPDATE_PACKAGE, OPERATIONS: UPDATE_SSW*]. **FDP_ACC.2.2/SSW_Update** The TSF shall ensure that all operations between any subject controlled by the TSF and any	Y

	object controlled by the TSF are covered by an access control SFP. **FDP_ACF.1.1/SSW_Update** The TSF shall enforce the [*SSW_UPDATE POLICY*] to objects based on the following: [*SUBJECTS: THE TSF, OBJECTS[ATTRIBUTES]: SSW[VERSION_ID, SIGNATURE], SSW_UPDATE_PACKAGE[VERSION_ID, SIGNATURE, PACKAGE_SOURCE]*]. **FDP_ACF.1.2/SSW_Update** The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [● *THE TSF IS ALLOWED TO PERFORM THE UPDATE_SSW OPERATION IF THE FOLLOWING CONDITIONS HOLD:* ○ *[selection: THE SSW_UPDATE_PACKAGE[VERSION_ID] IS NOT LOWER THAN THE SSW[VERSION_ID]]* ○ *THE SSW_UPDATE_PACKAGE[SIGNATURE] IS VERIFIED BY A DIGITAL SIGNATURE FROM THE TOE MANUFACTURER STORED ON THE DEVICE* ○ *THE SIGNATURE CHECK AND THE UPDATE_SSW ARE PERFORMED AS AN ATOMIC UPDATE FUNCTION***]**. **FDP_ACF.1.3/SSW_Update** The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [assignment: *no other rules*]. **FDP_ACF.1.4/SSW_Update** The TSF shall explicitly deny access of subjects to objects based on the following additional rules: [*THE TSF SHALL NOT ALLOW ANY TSF-MEDIATED ACTIONS RELATED TO THE UPDATE_SSW FUNCTION DURING ITS UPDATING*]. **FDP_UPF_EXT.1.1/SSW_Update** The TSF shall be able to check for a [SYSTEM SOFTWARE] update package every [assignment: ~~*interval perio once a day*~~]and provide a notification to the user when an update is available.	
	The device verifies all updates using a public key chaining to the root public key. The SHA-256 hash of the root public key is fused into the SoC during manufacturing. Once the update has been downloaded the version and	

	signature are checked to ensure the version is not older than the current version and that the signature is valid (verified to the hardware hash). https://source.android.com/docs/core/ota/ab If the checks complete successfully the update process begins immediately where the update will be installed on the slot (partition) that is not currently in use (i.e. the one that was not used to boot the device at this time). Once the update has been completely written to the slot, the update process will switch the active slot to the one just updated and restart the device (with user permission). If the reboot is not successful the system will automatically switch back to the previous slot and restart again, returning the device to the previous state. If the checks, write process or reboot fail at any time during this sequence, the system will remain/revert to the OS version that existed prior to the update being downloaded, such that the update process happens completely or not at all. The device checks for updates on a daily basis (once every 24 hours) to the update server. When an update is found the user will be notified to download the update. The user can also check to automatically install the update at a later time (or set to be notified again later).	

FDP_ACC.2 /Permissions **FDP_ACF.1/ Permissions**	Devices shall provide access controls (permissions) to components on the system and provide the user with the opportunity to grant or block access to these components to any application or if permissions are specifically granted by the device (i.e. in the operating system the application has specific privileges already). This is divided into what the user can explicitly control and what the OS controls.	Supported? (Y/N)
	Document the following: ● List of user permissions (i.e. camera, microphone, contacts) ● List of OS permissions (i.e. Device ID, system permissions, sockets/IPC, files) ● How permissions are granted (including for pre-installed applications where it is granted without user interaction) ● How access is determined (allow/block)	

	● SELinux is the basis for these permissions and controls	
	FDP_ACC.2.1/Permissions The TSF shall enforce the [*PERMISSIONS POLICY*] on [● *SUBJECTS: APPS, PROCESSES;* ● *USER OBJECTS: [selection: CAMERA, MICROPHONE, LOCATION, CONTACTS, CALENDAR, CALL LOG, STORED PICTURES, TEXT MESSAGES, THE LIST OF INSTALLED APPS, [assignment: body sensors, music and audio, nearby devices, notifications, phone, physical activity, SMS, storage, car information, install shortcuts, read instant messages, write instant messages];* ● *MANUFACTURER OBJECTS: DEVICE ID, SYSTEM PERMISSIONS, FILES (INCLUDING INDIVIDUAL APP DATA), [selection: SECURE SOCKETS, SECURE IPC, [assignment: LIST OF OTHER SENSITIVE USER DATA ASSETS AND/OR SYSTEM SERVICES THAT CAN BE ACCESSED BY AN APP OR PROCESS]];* ● *OPERATIONS: READ, WRITE, EXECUTE*]. **FDP_ACC.2.2/Permissions** The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP. **FDP_ACF.1.1/Permissions** The TSF shall enforce the [*PERMISSIONS POLICY*] to objects based on the following: [● *[APPS AND PROCESSES] AND THE OPERATIONS ASSOCIATED WITH THE SUBJECT;* ● *THE ACCESS CONTROL LIST ASSOCIATED WITH THE OBJECT BEING REQUESTED*]. **FDP_ACF.1.2/Permissions** The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [● *THE SUBJECT, OR A GROUPING THE SUBJECT IS MAPPED TO, IS EXPLICITLY GRANTED PERMISSION BY THE USER OR TOE MANUFACTURER TO THE USER OBJECT IN THE ACCESS CONTROL LIST***].** **FDP_ACF.1.3/Permissions** The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [● *THE SUBJECT IS GRANTED PERMISSION BY THE TOE MANUFACTURER TO THE MANUFACTURER OBJECT IN THE ACCESS CONTROL LIST*].	Y

	FDP_ACF.1.4/Permissions The TSF shall explicitly deny access of subjects to objects based on the following additional rules: [• *THE SUBJECT IS EXPLICITLY BLOCKED BY THE USER FROM ACCESSING THE USER OBJECT;* • *THERE IS NO RULE GRANTING THE SUBJECT ACCESS TO THE USER OR MANUFACTURER OBJECT IN THE ACCESS CONTROL LIST*].	
	The TOE provides the following categories of system services to applications. 1. Normal - A lower-risk permission that gives an application access to isolated application-level features, with minimal risk to other applications, the system, or the user. The system automatically grants this type of permission to a requesting application at installation, without asking for the user's explicit approval (though the user always has the option to review these permissions before installing). 2. Dangerous - A higher-risk permission that would give a requesting application access to private user data or control over the device that can negatively impact the user. Because this type of permission introduces potential risk, the system cannot automatically grant it to the requesting application. For example, any dangerous permissions requested by an application will be displayed to the user and require confirmation before proceeding or some other approach can be taken to avoid the user automatically allowing the use of such facilities. 3. Signature - A permission that the system is to grant only if the requesting application is signed with the same certificate as the application that declared the permission. If the certificates match, the system automatically grants the permission without notifying the user or asking for the user's explicit approval. 4. SignatureOrSystem - A permission that the system is to grant only to packages in the Android system image or that are signed with the same certificates. Please avoid using this option, as the signature protection level should be sufficient for most needs and works regardless of exactly where applications are installed. This permission is used for certain special situations where multiple vendors have applications built in to a system image which need to share specific features explicitly because they arebeing built together.	

	An example of a normal permission is the ability to vibrate the device: android.permission.VIBRATE. This permission allows an application to make the device vibrate, and an application that does not request (or declare) this permission would have its vibration requests ignored. An example of a dangerous privilege would be access to location services to determine the location of the mobile device: android.permission.ACCESS_FINE_LOCATION. The TOE controls access to Dangerous permissions during the running of the application. The TOE prompts the user to review the application's requested permissions (by displaying a description of each permission group, into which individual permissions map, that an application requested access to). If the user approves, then the application is allowed to continue running. If the user disapproves, the devices continues to run, but cannot use the services protected by the denied permissions. Thereafter, the mobile device grants that application during execution access to the set of permissions declared in its Manifest file. An example of a signature permission is the android.permission.BIND_VPN_SERVICE that an application must declare in order to utilize the VpnService APIs of the device. Because the permission is a Signature permission, the mobile device only grants this permission to an application (2nd installed app) that requests this permission and that has been signed with the same developer key used to sign the application (1st installed app) declaring the permission (in the case of the example, the Android Framework itself). An example of a signatureOrSystem permission is the android.permission.LOCATION_HARDWARE, which allows an application to use location features in hardware (such as the geofencing API). The device grants this permission to requesting applications that either have been signed with the same developer key used to sign the Android application declaring the permissions or that reside in the "system" directory within Android (which for Android 4.4 and above, are applications residing in the /system/priv-app/ directory on the read-only system partition). Put another way, the device grants systemOrSignature permissions by Signature or by virtue of the requesting application being part of the "system image". Additionally, Android includes the following flags that layer atop the base categories. 1. privileged - this permission can also be granted to any applications installed as privileged apps on the system image. Please avoid using this option, as the signature protection level should be sufficient for most needs and works regardless of	

	exactly where applications are installed. This permission flag is used for certain special situations where multiple vendors have applications built into a system image which need to share specific features explicitly because they are being built together. 2. system - Old synonym for 'privileged'. 3. development - this permission can also (optionally) be granted to development applications (e.g., to allow additional location reporting during beta testing). 4. appop - this permission is closely associated with an app op for controlling access. 5. pre23 - this permission can be automatically granted to apps that target API levels below API level 23 (Marshmallow/6.0). 6. installer - this permission can be automatically granted to system apps that install packages. 7. verifier - this permission can be automatically granted to system apps that verify packages. 8. preinstalled - this permission can be automatically granted to any application pre-installed on the system image (not just privileged apps) (the TOE does not prompt the user to approve the permission). For older applications (those targeting Android's pre-23 API level, i.e., API level 22 [lollipop] and below), the TOE will prompt a user at the time of application installation whether they agree to grant the application access to the requested services. Thereafter (each time the application is run), the TOE will grant the application access to the services specified during install. For newer applications (those targeting API level 23 or later), the TOE grants individual permissions at application run-time by prompting the user for confirmation of each permissions category requested by the application (and only granting the permission if the user chooses to grant it). These permissions can be tested using an application built using code that can be found at https://github.com/android/securitycertificationresources/tree/master/niap-cc/Permissions. While Android provides a large number of individual permissions, they are generally grouped into categories or features that provide similar functionality. Below what the user can directly manage through the UI as permissions, Android uses SELinux in enforcing mode for Mandatory Access Control (and all permission policies are implemented as SELinux policies, just exposed in a more userfriendly way). The SELinux policies can be found on the device using the command: adb pull /sys/fs/selinux/policy This is the compiled policy file that is enforced on the device and contains all the settings.	

	AOSP defines a number of highly dangerous SELinux controls (DAC_OVERRIDE, NET_ADMIN and SYS_ADMIN) and associates them to services that require those controls. These can be found in the AOSP source in the private and public folders.	

FDP_ACC.1 /UserDataA sset **FDP_ACF.1/ UserDataAs set**	Data stored on the device is protected with different classifications (DE/CE). Describe how user data is protected and how it is classified on the system. DE/CE is sufficient to meet the Low/Medium differences. Explain how High can be implemented.	Supported? (Y/N)
	FDP_ACC.1.1/UserDataAsset The TSF shall enforce the [*USER DATA ASSET DECRYPTION POLICY*] on [• *SUBJECTS: THE TSF;* • *OBJECTS: INTERNAL STORAGE (ALL SAVED USER DATA ASSETS), [selection: NO OTHER STORAGE];* • *OPERATIONS: DECRYPT*]. **FDP_ACF.1.1/UserDataAsset** The TSF shall enforce the [*USER DATA ASSET DECRYPTION POLICY*] to objects based on the following: [*SUBJECTS: THE TSF, OBJECTS: USER DATA ASSETS, ATTRIBUTES: SENSITIVE LEVEL OF OBJECTS, LOW, MEDIUM, HIGH*]. **FDP_ACF.1.2/UserDataAsset** The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [• *THE TSF IS ALLOWED TO DECRYPT LOW USER DATA ASSETS IF AND ONLY IF THE TOE IS SUCCESSFULLY POWERED ON; AND* • *THE TSF IS ALLOWED TO DECRYPT MEDIUM USER DATA ASSETS IF AND ONLY IF THE TOE IS SUCCESSFULLY POWERED ON AND THE USER IS SUCCESSFULLY AUTHENTICATED DURING THE FIRST AUTHENTICATION AFTER POWER ON; AND* • *THE TSF IS ALLOWED TO DECRYPT HIGH USER DATA ASSETS IF AND ONLY IF THE TOE IS SUCCESSFULLY POWERED ON, THE USER IS SUCCESSFULLY AUTHENTICATED AND THE SCREEN OF THE TOE IS NOT LOCKED*].	Y

	FDP_ACF.1.3/UserDataAsset The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [*NO ADDITIONAL RULES*]. **FDP_ACF.1.4/UserDataAsset** The TSF shall explicitly deny access of subjects to objects based on the following additional rules: [*NO ADDITIONAL RULES*].	
	The device provides different classes of storage for all user data. By default all user data is protected with a key that is derived from the user's password (this is considered Medium protection). The device only supports internal storage. For Low data, an application must be explicitly developed to take advantage of this feature which will allow data saved by the app as DE data to be available on reboot before the user has authenticated. For High data, an application must be explicitly developed to be aware of the lock status of the device to be able to close and lock its data. Android APIs provide the ability to hook into the notification of the lock status to enable this functionality. In all cases, keys that are stored on the device are entangled with the hardware root key (in the case of Low, this is combined with a hardcoded device key to take the place of the user's password). This ensures that data cannot be ported to another device and decrypted as only the device where the data is encrypted will have access to the hardware key used to generate the encryption key.	

Identification and Authentication

FIA_UAU.1 **FIA_UID.1**	When authentication is enabled, some actions may be performed before a successful login. Document what actions are allowed before a successful login. Access to stored data shall not be allowed (i.e. access to CE data)	Supported? (Y/N)
	FIA_UAU.1.1 The TSF shall allow [assignment: • *Enter password to unlock* • *Tak pictures (stored internally) - unless the camera was disabled* • *Turn the TOE off* • *Restart the TOE* • *See notifications (note that some notifications identify actions,for example to view a screenshot; however, selecting those notifications highlights the password prompt and require the password to access that data)* • *Configure sound, vibrate, or mute* • *Set the volume (up and down) for ringtone* • *Access notification widgets (without authentication):* *- Enable and disable Wi-Fi connection* *- Enable and disable Bluetooth connection* *- Switch between Silent mode, Vibrate and Ring* *- Enable and disable Flashlight* *- Enable and disable Airplane mode* *- Enable and disable Dark mode toggle* *- Take Screenshots* *- Enable and disable Personal hotspot* *- Change Display brightness* *- Enable and disable auto display brig*] on behalf of the user to be performed before the user is authenticated. **FIA_UAU.1.2** The TSF shall require the user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user. **FIA_UID.1.1** The TSF shall allow [assignment: *list of TSF-mediated actions， which are listed in FIA-UAU.1.1*] on behalf of the user to be performed before the user is identified. **FIA_UID.1.2** The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.	Y

	No action can be taken before the user has been authenticated after reboot. The following actions can be taken before the user has authenticated to the device when device is locked but not right after reboot: • *Enter password to unlock* • *Make/receive emergency calls* • *Take pictures (stored internally) - unless the camera was disabled* • *Turn the TOE off* • *Restart the TOE* • *See notifications (note that some notifications identify actions, for example to view a screenshot; however, selecting those notifications highlights the password prompt and require the password to access that data)* • *Configure sound, vibrate, or mute* • *Set the volume (up and down) for ringtone* • *Access notification widgets (without authentication):* *- Enable and disable Wi-Fi connection* *- Enable and disable Bluetooth connection* *- Switch between Silent mode, Vibrate and Ring* *- Enable and disable Flashlight* *- Enable and disable Airplane mode* *- Enable and disable Auto-rotate* *- Enable and disable reverse wireless charging* *- Enable and disable Dark mode toggle* *- Take Screenshots* *- Enable and disable Personal hotspot* *- Change Display brightness* *- Enable and disable auto display brig* All other actions require the user to be authenticated.	

FIA_UAU.5/ Local	Multiple methods of authentication may be supported.	Supported? (Y/N)
	Describe the methods of authentication that are provided including any biometric modalities or external devices that may be supported. Describe the ordering for how multiple methods may be used at one time (for example if fingerprint is enabled, how does the device decide whether to ask for the fingerprint or PIN).	
	FIA_UAU.5.1/Local The TSF shall provide [*PASSWORD, PIN* and selection: 4 digits PIN, 6 digits PIN, 4-16 digits PIN,	Y

	PATTERN, 2D Face [assignemnt: EAF], NO OTHER MECHANISM] to support user authentication. **FIA_UAU.5.2/Local** The TSF shall authenticate any user's claimed identity according to the [assignment: *To authenticate unlocking the device immediately after boot (first unlock after reboot):* • *User passwords are required after reboot to unlock the user's Credential encrypted (CE files) and keystore keys. Biometric authentication is disabled immediately after boot.* *To authenticate unlocking the device after device lock (not following a reboot):* ● *The TOE verifies user credentials (password) via the gatekeeper, which compares the entered credential to a derived value or template.* *To change protected settings or issue certain commands:* ● *The TOE requires password after a reboot, when changing settings (Screen lock, and Smart Lock settings), and when factory resetting.*].	
	The device supports the use of a password, PIN, pattern and 2D Face for authentication. When the device is first started (or restarted), the biometric cannot be used for authentication until after a password, pattern or PIN has been entered first.	

FIA_UAU.5/Peer

This is for functionality that is provided with the device from the developer and does not include apps that may be downloaded by the user after purchase.

FIA_UAU.5/ Peer	When connecting to a peer device or a service, the user shall successfully authenticate before the connection is allowed.	Supported? (Y/N)
	Document the methods for connecting to: ● peer-to-peer device connections ○ Bluetooth pairing methods are handled in FTP_ITC_EXT.1/BT and not necessary here ● Other services ○ For example, backup/sync services using a trusted server ○ Screen mirroring/sharing	

	For other services, specify what is allowed after the successful pairing	
	FIA_UAU.5.1/Peer The TSF shall provide **support to use** [selection: *EXPLICIT ACCEPTANCE OF CONNECTION, QR CODES, NFC LABELS, PINS, USING A COMMON USER ACCOUNT TO A REMOTE SERVICE ON BOTH DEVICES, [assignment: OTHER MECHANISM TO VERIFY THE PEER IDENTITY]*] **for** ~~to support~~ ~~user~~ authentication **to peer devices before allowing any actions on behalf of the user**. **FIA_UAU.5.2/Peer** The TSF shall authenticate any ~~user's~~ **peer device's** claimed identity according to the [assignment: *rules describing how the multiple authentication mechanisms provide **peer device** authentication*].	Y（not support nfc）
	The device provides the following ways to authenticate connections to peer devices/services: ● QR codes - these can be used to add Wi-Fi information to connect to an AP ○ Note that QR codes can be read automatically by the camera, but by default are handed to the web browser and are not tied to the device itself ● Apps can register accounts (seen in Settings -> Passwords & accounts) that can be used to connect to remote services (such as a mail server or other cloud service). These require the user to enter a valid username/password combination on the device that matches an account on the service being connected to.	

FIA_UAU.6	The user needs to be re-authenticated to perform specific actions (this is specifically after the initial authentication to unlock the device after a bootup/startup).	Supported? (Y/N)
	Document what actions require authentication: ● Unlocking the device after it has become locked (mandatory) ● Change of authentication data (any change) (mandatory) ● Other conditions?	
	FIA_UAU.6.1 The TSF shall re-authenticate the user under the conditions [● *ATTEMPTED CHANGE OF ANY USER AUTHENTICATION FACTOR;* ● *ATTEMPTED UNLOCKING OF A LOCKED TOE;*	Y

	• *[selection: NO OTHER CONDITIONS]*.	
	The device requires the user to enter their password or supply their biometric in order to unlock the device. Additionally, the device requires the user to confirm their current password when accessing the Settings -> Security and Privacy -> Device unlock menu in the user interface. Only after entering their current user password can the user then elect to change their password.	

FIA_UAU.7	The user should see only limited feedback during authentication is in progress	Supported? (Y/N)
	Describe what feedback is provided to the user during authentication, such as password/PIN display (how long before masking).	
	FIA_UAU.7.1 The TSF shall provide only [*BRIEF FEEDBACK ABOUT THE ENTERED CREDENTIALS*] to the user while the authentication is in progress.	Y
	The device allows the user to enter the user's password from the lock screen. The device will display the most recently entered character of the password briefly or until the user enters the next character in the password, at which point the device obscures the character by replacing the character with a dot symbol. Further, the device provides no feedback other than whether the unlock attempt succeeded or failed.	

FIA_SOS.1	The user shall be able to set credentials of a minimum strength	Supported? (Y/N)
	Document the minimum/maximum requirements for password/PIN/pattern settings for the user and specify the character set used for a password.	
	FIA_SOS.1.1 The TSF shall provide a mechanism to verify that secrets meet [• *FOR PASSWORD AND PIN: LENGTH 4 OR MORE FROM A DEFINED CHARACTER SET;* • *FOR PATTERNS: CONSISTS OF AT LEAST 4 FROM A SET OF AT LEAST 9 AVAILABLE POINTS, WHERE EACH POINT SHALL ONLY BE USED ONCE*].	Y
	The minimum length for any password or PIN is 4 characters/numbers. Passwords consist of basic Latin characters (upper and lower case, numbers, and the following	

	special characters: ! @ # $ % ^ & * ( ) [= + - _ ` ~ \ |] } [{ ‘ “ ; : / ? . > , < The maximum length for a password or PIN is 16 characters/numbers. For a pattern the user must trace a pattern that touches at least 4 individual points from the 9 available.	

FIA_AFL.1	When repeated authentication attempts are detected, the device should deter continued attempts. This includes all available authentication methods.	Supported? (Y/N)
	Document what actions are taken when some number of attempts have been detected (between 3-10).	
	FIA_AFL.1.1 The TSF shall detect when [*AN INTEGER BETWEEN 3 AND 10*] unsuccessful authentication attempts occur related to [~~assignment: list of authentication events~~ *selection: PASSWORD, PIN, PATTERN, **2D FACE***]. **FIA_AFL.1.2** When the defined number of unsuccessful authentication attempts has been [MET], the TSF shall [selection: *REBOOT, PROGRESSIVELY LENGTHEN THE TIME TO ATTEMPT AN AUTHENTICATION, MAKE ALL USER DATA ASSETS UNREADABLE, [assignment:* • *Block the biometric use after 5 unsuccessful attempts* • *Require a 30 second delay after 5 unsuccessful non-biometric attempts, and then again after 10 total attempts* • *Additional attempts past 11 will have at least a 30 second delay on every attempt (eventually growing to a day between attempts)*].	Y
	The device maintains in persistent storage the number of failed authentication attempts since the last login. The Gatekeeper code for calculating the delay is here (written below based on the number of attempts taken). • [0, 4] -> 0 • 5 -> 60 • 6 -> 300 • 7 -> 900 • 8 -> 1800 • 9 -> 5400 • >9 -> 60 * 3^(failure count –5) • > 11 -> 24 hours Every 5 consecutive failed attempts the device will force the user to wait for a period of time before further attempts (starting at 30	

	seconds and then doubling in time). Every 10 attempts it alsorequires the device to reboot to perform further attempts. Biometric authentication can be attempted up to 5 times before the biometric is disabled and the user can only use a password, PIN or pattern to authenticate. When biometrics are enabled, they can be used once the screenwakes up. Entering a password, PIN or pattern requires an additional swipe of the screen to activate the sign in dialog to enter the credential. https://source.android.com/docs/security/features/authentication/rate-limiting	

Security Management

FMT_MSA.1 /Permissions **FMT_MSA.3 /Permissions**	The user shall be able to manage the user permissions on the available objects. The default policy if no permission is assigned should be restrictive.	Supported? (Y/N)
	Document what the user can do in setting permissions for access (approve/reject persistently or temporarily, etc)	
	FMT_MSA.1.1/Permissions The TSF shall enforce the [*PERMISSIONS POLICY*] to restrict the ability to [selection: ~~*change_default, query,*~~ *APPROVE PERSISTENTLY, APPROVE TEMPORARILY, REJECT PERSISTENTLY, REJECT TEMPORARILY, MODIFY*, ~~*delete*~~ *[assignment: other operations]*] the security attributes [*LIST OF USER OBJECTS] to [THE CURRENT USER FOR THEIR OWN PERMISSIONS]*. **FMT_MSA.3.1/Permissions** The TSF shall enforce the [*PERMISSIONS POLICY*] to provide [RESTRICTIVE] default values for security attributes that are used to enforce the SFP. **FMT_MSA.3.2/Permissions** The TSF shall allow the [*CURRENT USER FOR THEIR OWN PERMISSIONS*] to specify alternative initial values to override the default values when an object or information is created.	Y
	The user can manage the permissions for an application based on 9 different groups of permissions. Permissions for an app can be modified after they have initially been set through Settings -> App management -> Specific App Here the user can choose to change the permissions that have	

	been granted (or rejected).	

FMT_SMF.1 /Authentication	The user shall be able to specify and later change their authentication credentials	Supported? (Y/N)
	Document how the user can set and change the password/PIN/pattern/biometric	
	FMT_SMF.1.1/Authentication The TSF shall be capable of performing the following management functions: [~~assignment~~ selection: • *ENTERING AN INITIAL OR CHANGING (INCLUDING REMOVAL OF) THE KAF]*	Y
	The user can change their authentication credentials through Settings -> Security & privacy -> Device lock From here the user can choose Screen lock to change their credentials (set, remove, modify, etc). Setting a 2D face will require the user to also set a password, PIN or pattern	

FMT_SMF.1 /Permissions	The user shall be able to manage the permissions provided to apps and some system services	Supported? (Y/N)
	• Document how the user can view, grant and revoke permissions for any app	
	FMT_SMF.1.1/Permissions The TSF shall be capable of performing the following management functions: [• *VIEW PERMISSIONS GRANTED TO AN APP; AND* • *GRANT/REVOKE PERMISSION TO/FROM AN APP OR PROCESS TO HAVE READ AND/OR WRITE ACCESS TO A USER OBJECT*].	Y
	Permissions for an app can be modified after they have initially been set through Settings -> App management -> Specific App Here the user can choose to change the permissions that have been granted (or rejected).	

FMT_SMF.1 /UserControls	The user shall be able to manage various settings on the device • Document the settings for accessibility service, notifications • Document how the user can set the default USB mode when connecting to a computer • Removable media encryption	Supported? (Y/N)
	FMT_SMF.1.1/UserControls The TSF shall be capable of performing the following management functions: [• *GRANT/REVOKE PERMISSION TO/FROM AN APP OR PROCESS TO HAVE ACCESS TO ACCESSIBILITY SERVICE; AND* • *GRANT/REVOKE PERMISSION TO/FROM AN APP OR PROCESS TO HAVE ACCESS TO DEVICE NOTIFICATION; AND* • *[selection: CHARGE ONLY MODE BY DEFAULT, FILE TRANSFER MODE, [assignment: OTHER WIRED CHARGING MODE]] WHEN THE TOE IS CONNECTED VIA THE WIRED CHARGING INTERFACE TO ANOTHER DEVICE; AND* • *[selection: NO SUPPORT FOR REMOVABLE MEDIA]; and* • *[assignment: None]*.	Y
	The user can manage apps that can access the accessibility service through: Settings -> Accessibility & convenience The user can manage the notifications (which apps can provide notifications, access to read notifications, whether they can be seen on the lock screen) through: Settings -> Notifications The user can manage USB connectivity to another device (such as a PC) through the USB preferences. By default the selection is to not transfer data over the connection. This option becomes available when the device is connected to another device, at which point these settings can be adjusted for the connection. The device does not support removable media	

FMT_SMF.1 /Privacy	The user shall be able to change the alias used as an identifier for ads and developers	Supported? (Y/N)
	Document how the user can change the alias (or disable it).	

	FMT_SMF.1.1/Privacy The TSF shall be capable of performing the following management functions: [~~assignment:~~ selection: • *CHANGE OR RESET THE PRIVACY ALIASES;* • *BLOCK THE CREATION/USE OF A UNIQUE ID FOR ADVERTISING (NO PERSONALIZED TRACKING)*].	Y
	The user can access the advertising controls in: Settings -> Security & privacy -> Privacy Controls -> Ads Here the user can reset the advertising ID or delete it completely. If the ID is deleted then personalized tracking can be disabled.	

FMT_SMF.1/APP_Update	The user shall be able to manage applications on the device (update/uninstall)	Supported? (Y/N)
	Describe how application updates can be initiated and how apps (including pre-installed) can be uninstalled.	
	FMT_SMF.1.1/APP_Update The TSF shall be capable of performing the following management functions: [~~assignment:~~ • *SPECIFY TO [selection: NOTIFY THE USER WITHOUT DOWNLOADING, AUTOMATICALLY INSTALL] WHEN AN AUTOMATIC CHECK TO THE ADP HAS FOUND AN UPDATE; AND* • *INITIATE AN IMMEDIATE CHECK FOR UPDATE; AND* • *INITIATE AN UPDATE OF THE APP (IF AVAILABLE); AND* • *DISPLAY THE VERSION NUMBER OF THE APP; AND* • *UNINSTALL DOWNLOADED APPS (INCLUDING APPS DOWNLOADED AS PART OF THE SETUP PROCESS)*].	Y
	Through the Play Store the user can manage the apps they download to the device. The Play Store allows the user to install apps, check for updates, and uninstall apps. The Play Store checks for updates to the installed apps roughly once per day to see if there are new versions to be installed. In general the apps will be updated automatically, though if there are permission changes to the new version, the user will be required to approve the installation of the new version. The user can also uninstall any app (that can be uninstalled) in the Settings page Settings -> App management -> <app in question> Click the Uninstall button for the app	

FMT_SMF.1 /SSW_Update	The user shall be able to check for system software updates Describe how the user can check for new system software updates and how they can be installed.	Supported? (Y/N)
	FMT_SMF.1.1/SSW_Update The TSF shall be capable of performing the following management functions: [~~assignment:~~ • • *SPECIFY TO [selection: NOTIFY THE USER WITHOUT DOWNLOADING, AUTOMATICALLY INSTALL] WHEN AN AUTOMATIC CHECK TO THE TRUSTWORTHY UPDATE SOURCE HAS FOUND AN UPDATE; AND* • *INITIATE AN IMMEDIATE CHECK FOR UPDATE; AND* • *INITIATE AN UPDATE OF THE SYSTEM SOFTWARE (IF AVAILABLE); AND* • *PROVIDE THE STATUS OF THE UPDATE PROCESS AND THE RESULTS OF THE UPDATE; AND* • *[selection: DELAY TEMPORARILY, BLOCK PERSISTENTLY] AUTOMATIC INSTALLATION OF SYSTEM SOFTWARE UPDATES; AND* • *DISPLAY THE VERSION NUMBER OF THE SYSTEM SOFTWARE*].	Y
	The user can check for new updates by going to: Settings -> System&Update -> Software Update	

Privacy

FPR_PSE.1	Advertisers and app developers shall be provided a unique device identifier that can be modified by the user. Describe how the unique identifier is provided and how this can be changed by the user.	Supported? (Y/N)
	FPR_PSE.1.1/Advertisers The TSF shall ensure that [*AD NETWORKS*] are unable to ~~determine~~ **access** the ~~real user name~~ **Device ID** bound to [*THE TOE (HARDWARE PLATFORM)*] **according to the Permissions Policy**. **FPR_PSE.1.2/Advertisers** The TSF shall be able to provide [*AT LEAST ONE UNIQUE*] alias**(**es**)** of the ~~real user name~~ **Device ID** to [*AD NETWORKS*].	Y

	FPR_PSE.1.3/Advertisers The TSF shall [DETERMINE AN ALIAS FOR A DEVICE ID] ~~and verify that it conforms to the [assignment: *alias metric*]~~. **FPR_PSE.1.1/APP_Dev** The TSF shall ensure that [*APP DEVELOPERS*] are unable to ~~determine~~ **access** the ~~real user name~~ **Device ID** bound to [*THE TOE (HARDWARE PLATFORM)*] **according to the Permissions Policy**. **FPR_PSE.1.2/APP_Dev** The TSF shall be able to provide [*AT LEAST ONE UNIQUE*] alias**(**es**)** of the ~~real user name~~ **Device ID** to [*EACH APP DEVELOPER*]. **FPR_PSE.1.3/APP_Dev** The TSF shall [PROVIDE AN ALIAS FOR A DEVICE ID] ~~and verify that it conforms to the [assignment: *alias metric*]~~ **upon request by the App developer**.	
	The device provides a single identifier for advertisers to use. This identifier is not tied to the hardware (it is not derived from any hardware identifiers) and is randomly generated. Advertisers are not able to access unique hardware identifiers. User can reset and delete advertising ID in the same page below Settings -> Security & privacy -> Privacy Controls -> Ads	

Protection of the TSF

FPT_PHP.3	OEMs and ODMs shall provide a hardware-backed key store implemented within a SEE, Secure Element or equivalent solution. Plaintext private key material shall not exist outside of the hardware-backed key store.	Supported? (Y/N)
	Provide documentation on the hardware-backed credential storage capabilities of the device, as well as the specific configurations.	
	FPT_PHP.3.1 The TSF shall resist [*to:* • *READ OR MODIFY THE DUK; AND* • *READ OR MODIFY [assignment: no keys (keys are not stored unencrypted]; AND* • *MODIFY [assignment: HASHES OF KEY(S)] USED TO VERIFY THE INTEGRITY OF THE TSF IN FPT_TST.1; AND* • *MODIFY [assignment: HASHES OF KEY(S)] USED TO VERIFY THE INTEGRITY AND AUTHENTICITY OF UPDATES TO THE TSF IN FCS_COP.1/ASYMMETRIC; AND* • *READ OR MODIFY [assignment: no other keys]*]; to the **[***HARDWARE BASED SECURE ENVIRONMENT OF THE TSF***]** by responding automatically such that ~~the SFRs are always enforced~~ **it is impossible to read or modify this data and/or key(s)**.	Y
	The device provides multiple hardware-based storage locations for keys to maintain confidentiality and integrity. The DUK is stored in one-time fuses in the AP itself. These are never read directly by anything outside a security subsystem included in the AP. The device also provides support for the StrongBox Keystore. Apps and processes can use this to store keys instead of just placing them into the TEE Keystore. Access to StrongBox is through the normal Keystore API with a separate flag to note that the key must be stored in dedicated hardware.	

FPT_FLS.1	A failure in the system software update shall not expose user data.	Supported? (Y/N)

	Describe how a failure of the update process is managed. For example, an automatic rollback, or some other process which would allow the user to restore the system but which would not expose any encrypted user data.	
	FPT_FLS.1.1 The TSF shall preserve a secure state when the following types of failures occur: [*FAILURE OF THE UPDATE_THE_TOE_SOFTWARE OPERATION IN FDP_ACF.1/SSW_UPDATE*].	
	All user data on the device is encrypted. Except for data that is not encrypted with the user's credentials (DE data), user data cannot be decrypted without a successful user authentication. So, a failure to update the system software cannot unlock the user data as the new system will not be able to properly start (the failure). The device maintains two boot slots for the system. Only one is active at a time, and the alternate slot is used to update the device. If the update process fails at any point (such as during the write operation or upon reboot the system cannot successfully start and the user authenticates), the system will revert to the original boot slot and return to the previous version of Android (the version running at the time the update was downloaded)	Y

FPT_TST.1	During the device start-up process, the integrity of the system software shall be verified.	
	Different tests are allowed for different types of checks. For example: ● Cryptographic algorithms can run self-tests (RBG can verify output) ● Bootloader and other system checks using hash trees, signatures (or hashes)	Supported? (Y/N)
	FPT_TST.1.1 The TSF shall run a suite of self-tests **and integrity verification** [DURING INITIAL START-UP] to demonstrate the correct operation of [[selection: *[assignment: a safe version of Android], the TSF*] BY SELF TESTS AND THE BOOTLOADER, MAIN OS KERNEL [selection: SEE, *[assignment: PARTS OF TSF]*] BY INTEGRITY, WHERE INTEGRITY IS VERIFIED BY [selection: *a hash of an asymmetric key*]. **FPT_TST.1.2** The TSF shall provide authorised users with the capability to verify the integrity [*NO DATA*].	Y

	FPT_TST.1.3 The TSF shall provide authorised users with the capability to verify the integrity of [*NONE*].	
	The device verifies the integrity of the following components during the start-up: ● Bootloader ● OS kernel ● TEE ● Executable code stored in /system and /vendor partitions All these are verified by checking the signature using the hash of an asymmetric key that is fused into the SoC during manufacturing. The code stored in the /system and /vendor partitions is checked using dm-verity, where the tree is verified to the hash.	

FPT_RCV.2	The device shall support a maintenance mode to enable recovery from malicious/persistent software.	
	Describe the process by which malicious software may be removed automatically (where possible), and if this isn't possible, how the user may do so (for example, safe boot, manually re-installing the system software or a factory reset)	Supported? (Y/N)
	FPT_RCV.2.1 When automated recovery from [*DETECTION OF A MALEVOLENT PERSISTENT PRESENCE BY FPT_TST.1 OR AN UPDATE FAILURE BY FDP_ACF.1/SSW_UPDATE*] is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided. **FPT_RCV.2.2** For [*DETECTION OF A MALEVOLENT PERSISTENT PRESENCE BY FPT_TST.1 OR AN UPDATE FAILURE BY FDP_ACF.1/SSW_UPDATE*], the TSF shall ensure the return of the TOE to a secure state using automated procedures.	Y
	For detection of issues in the /system and /vendor partitions (such as an attempt to write malicious code or make other malicious changes to code stored there), most errors will be automatically corrected by the dm-verity check during the startup (the error will be detected and corrected automatically). For larger failures, such as changes to other code on the system (i.e. not in the /system or /vendor partitions), the device will boot into a recovery state where-in the user can manually reload known-good firmware onto the device.	

Trusted Path/Channels

FTP_ITC_EXT.1/BT	The device shall support at least Bluetooth Core Specification v4.1	Supported? (Y/N)
	Document the Bluetooth support on the device Describe the process for regenerating ECDH key pairs with paired devices	
	FTP_ITC_EXT.1.1/BT The TSF shall use [*BLUETOOTH® CORE SPECIFICATION THAT CONFORMS TO [selection: v4.1 [11], v4.2 [12], v5.0 [13], v5.1 [14], v5.2 [15], [assignment: A VERSION HIGHER THAN V5.2]]*] to provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure. **FTP_ITC_EXT.1.2/BT** The TSF shall permit [selection: *the TSF, another trusted IT product*] to initiate communication via the trusted channel. **FTP_ITC_EXT.1.3/BT** The TSF shall initiate communication via the trusted channel for [*CONNECTIONS TO BLUETOOTH DEVICES*]. **FTP_ITC_EXT.1.4/BT** The protocol used by the communications channel shall support the following requirements: [• *REQUIRE EXPLICIT USER AUTHORISATION BEFORE PAIRING; AND* • *USE SECURE SIMPLE PAIRING AND SECURE CONNECTIONS FOR PAIRING; AND* • *NOT ALLOW MORE THAN ONE BLUETOOTH CONNECTION TO THE SAME BLUETOOTH DEVICE ADDRESS; AND* • *GENERATE NEW ECDH PUBLIC/PRIVATE KEY PAIRS EVERY [selection: 24 HOURS, THREE FAILED AUTHENTICATION ATTEMPTS FROM ANY BLUETOOTH DEVICE ADDRESS, TEN SUCCESSFUL PAIRINGS FROM ANY BLUETOOTH DEVICE ADDRESS, [assignment: OTHER FREQUENCY AND/OR CRITERIA FOR NEW KEY PAIR GENERATION]]*]. The device has been qualified according to the Bluetooth 5.3 specification. Pairing is not allowed without explicit authorization	Y

	from the user (this cannot be programmatically entered but must be input directly from the user). Each time the device is connected to a peer over Bluetooth a new ECDH key pair will be created for the connection.	

FTP_ITC_EXT.1/HTTPS **FTP_ITC_EXT.1/TLS**	The device shall support TLS 1.2 or higher	Supported? (Y/N)
	Document the following: • versions of TLS that are supported (1.2 & 1.3 expected) • IETF RFC 5280 support for certificate revocation checking • What happens if a certificate is determined to be invalid • What TLS ciphersuites are supported (for apps/websites to use)	
	FTP_ITC_EXT.1.1/HTTPS The TSF shall use [*HTTPS THAT CONFORMS TO IETF RFC 2818 [5]*] to provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure. **FTP_ITC_EXT.1.2/HTTPS** The TSF shall permit [THE TSF] to initiate communication via the trusted channel. **FTP_ITC_EXT.1.3/HTTPS** The TSF shall initiate communication via the trusted channel for [*COMMUNICATION WITH A TRUSTED IT PRODUCT*]. **FTP_ITC_EXT.1.4/HTTPS** The protocol used by the communications channel shall support the following requirements: [*USE TLS AS SPECIFIED IN FTP_ITC_EXT.1/TLS TO IMPLEMENT HTTPS*]. **FTP_ITC_EXT.1.1/TLS** The TSF shall use [*TLS THAT CONFORMS TO [selection: TLS V1.2 [6], TLS V1.3 [10]]* to provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure. **FTP_ITC_EXT.1.2/TLS** The TSF shall permit [selection: *the TSF*] to initiate communication via the trusted channel.	Y

	FTP_ITC_EXT.1.3/TLS The TSF shall initiate communication via the trusted channel for [*COMMUNICATION WITH A TRUSTED IT PRODUCT*]. **FTP_ITC_EXT.1.4/TLS** The protocol used by the communications channel shall support the following requirements: [● *SUPPORT X.509V3 CERTIFICATES FOR MUTUAL AUTHENTICATION; AND* ● *DETERMINE VALIDITY OF THE PEER CERTIFICATE BY CERTIFICATE PATH, EXPIRATION DATE AND REVOCATION STATUS ACCORDING TO IETF RFC 5280 [7]; AND* ● *NOTIFY THE TSF AND [selection: NOT ESTABLISH THE CONNECTION, REQUEST APPLICATION AUTHORIZATION TO ESTABLISH THE CONNECTION, NO OTHER ACTION] IF THE PEER CERTIFICATE IS DEEMED INVALID; AND* ● *SUPPORTS THE FOLLOWING CIPHER SUITES [selection:* • *TLS_RSA_WITH_AES_256_GCM_SHA384 (IETF RFC 5288 [8]),* • *TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (IETF RFC 5288 [8]),* • *TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (IETF RFC 5289 [9]),* • *TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (IETF RFC 5289 [9]),* • *TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA 256 (IETF RFC 5289 [9]),* • *TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA 384 (IETF RFC 5289 [9]),* • *[assignment:* • *TLS_AES_128_GCM_SHA* • *TLS_AES_256_GCM_SHA384* • *TLS_CHACHA20_POLY1305_SHA25* • *TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256* • *TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256* • *TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA* • *TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA* • *TLS_RSA_WITH_AES_128_GCM_SHA256* • *TLS_RSA_WITH_AES_128_CBC_SHA* • *TLS_RSA_WITH_AES_256_CBC_SHA]* *]*	

	].	
	The device supports mutual authentication using certificates, and can check the validity of the peer certificate according to RFC 5280. Android supports the ability for an app to check the certificate revocation status using OCSP. The app must implement the APIs to enable the check and determine the action to take if the certificate is found to be revoked (Android does not perform any action based on the information itself other than notifies the app requesting the check of the state). The device does not act as a server so TLS connections must be initiated by the device (a request from a server to switch to a TLS connection will cause the device to start one) as opposed to accepting incoming connections that are.	

FTP_ITC_EXT.1/WLAN	The device shall support IEEE 802.11-2012, 802.1X & EAP-TLS connectivity	Supported? (Y/N)
	Document the Wi-Fi support on the device including ● Key lengths supported ● TLS 1.2 or higher support ● IETF RFC 5280 support for certificate revocation checking ● What happens if a certificate is determined to be invalid ● What TLS ciphersuites are supported ● MAC address randomization process (frequency)	
	FTP_ITC_EXT.1.1/WLAN The TSF shall use [*WLAN THAT CONFORMS TO 802.11-2012 [16]*] to provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure. **FTP_ITC_EXT.1.2/WLAN** The TSF shall permit [selection: *the TSF, another trusted IT product*] to initiate communication via the trusted channel. **FTP_ITC_EXT.1.3/WLAN** The TSF shall initiate communication via the trusted channel for [*COMMUNICATION WITH THE TRUSTED IT PRODUCT THROUGH WLAN CHANNEL*].	Y

	FTP_ITC_EXT.1.4/WLAN The protocol used by the communications channel shall support the following requirements: [● *GENERATE SYMMETRIC KEYS ACCORDING TO [selection: PRF-384 WITH KEY LENGTH 128 BIT, PRF-704 WITH KEY LENGTH 256 BIT]; AND* ● *USES [selection: TLS V1.2 [6], TLS V1.3 [10], [assignment: A VERSION OF TLS HIGHER THAN V1.2 [6]]]; AND* ● *SUPPORTS THE FOLLOWING CIPHER SUITES [selection:* ○ *TLS_RSA_WITH_AES_256_GCM_SHA384 (IETF RFC 5288 [8]),* ○ *TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (IETF RFC 5288 [8]),* ○ *TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (IETF RFC 5289 [9]),* ○ *TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (IETF RFC 5289 [9]),* ○ *TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (IETF RFC 5289 [9]),* ○ *TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (IETF RFC 5289 [9]),* ○ *TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (IETF RFC 5289 [9]),* ○ *TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (IETF RFC 5289 [9]),* ○ *TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (IETF RFC 5289 [9]),* ○ *TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (IETF RFC 5289 [9])]* ● *RANDOMLY GENERATE A NEW MAC ADDRESS EACH TIME IT CONNECTS TO A DIFFERENT ACCESS POINT*].	
	The device has been certified to meet the requirements for the Wi-Fi Alliance. The device supports PRF-384 and PRF-704 key generation and TLS v1.2. The following cipher suites are supported: ● 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	

TS 103 732-2 SFRs

Identification and Authentication

FIA_MBE_EXT.1	The user must be able to enroll their biometric sample to create an authentication template.	Supported? (Y/N)
	Document the process for enrolling the user's biometric.	
	FIA_MBE_EXT.1.1 The TSF shall provide a mechanism to enrol an authenticated user to the biometric system.	Y
	The device supports 2D face. User could enroll the 2D face by taking the face picture. Setting -> Biometrics&passwords-> Facial recognition	

FIA_MBV_EXT.1	Biometric authentication sensors shall meet minimum requirements for use.	Supported? (Y/N)
	Document the FAR/FRR information for each biometric modality available.	
	FIA_MBV_EXT.1.1 The TSF shall provide a biometric verification mechanism using [selection: *2D FACE*]. **FIA_MBV_EXT.1.2** The TSF shall provide a biometric verification mechanism with the [FAR] not exceeding [~~assignment:~~ selection: *1:50 000 FOR 2D FACE*] and [FRR] not exceeding [~~assignment:~~ selection: 6.627% *FOR 2D FACE*].	Y
	FAR and FRR on device has met the thresholds.	

Management

FMT_SMF.1/BAF	The user must be able to manage their biometric templates.	Supported? (Y/N)
	Document what the user can do in terms of managing enrolled biometrics.	
	FMT_SMF.1.1/BAF The TSF shall be capable of performing the following management functions: [• *ENROL THE INITIAL [selection: 2D FACE,]; AND* • *RE-ENROL OR CHANGE THE [selection: 2D FACE]*.	***Y***
	User could delete, disable their enrolled 2D face And add their face.	

TS 103 732-4 SFRs

Applications

FAP_LFC.1	The developer shall describe how preloaded apps included in the system image are updated.	Supported? (Y/N)
	The developer can provide a list of the apps along with how they are updated.	
	FAP_LFC.1.1 The TSF shall ensure that preloaded applications are updated by [selection: *using an ADP and system software updates (to the version maintained in firmware)*].	*Y*
	Some preloaded applications are only updated as part of the system software updates but most are provided as apps in the Play Store and can be updated as any application update.	

FAP_LFC.2	The developer shall specify what happens when a preloaded app is uninstalled and the app reverts back to the one included in the current system image.	Supported? (Y/N)
	The developer shall explain the actions that are taken automatically and what the user may be able to do with the app once it has been uninstalled.	
	FAP_LFC.2.1 When a preloaded app update is uninstalled, the TSF shall warn the user the preloaded app will revert to the version in the system image and allow the following actions: [selection: *disable the app*].	*Y*
	Some preloaded apps may be able to be disabled (along with uninstalling any app updates). This depends on the app (some are critical to the device and so cannot be disabled). When the app is not able to be disabled the user is warned about uninstalling the updates and then continuing to use the app from that point.	

FAP_LFC.3	The developer shall specify the ADP(s) where preinstalled apps are linked to for updates.	Supported? (Y/N)
	The developer should provide information about the ADP(s) where preinstalled apps will download updates from (updates should not be downloaded from a location that is not an ADP)..	
	FAP_LFC.3.1 The TSF shall ensure that preinstalled applications are only updated from the ADP(s) of the TOE manufacturer and/or OS developer.	*Y*
	All preinstalled apps are installed through the Play Store and so use the Play Store for all updates.	

FAP_LFC.4	The developer shall specify the ADP(s) where preinstalled apps are downloaded from during the installation process.	Supported? (Y/N)
	The developer should provide information about the allowed ADP(s) where apps may be downloaded from. The apps do not have to be specified.	

	FAP_LFC.4.1 The TSF shall ensure that preinstalled applications are only downloaded from the ADP(s) of the TOE manufacturer and/or OS developer.	*Y*
	All preinstalled apps are installed from the Play Store	

FAP_PRM.1	The developer shall specify the permissions that are restricted to only preloaded and vendor-owned preinstalled apps.	Supported? (Y/N)
	The developer provides a list of system permissions that are restricted to only these apps so that any other app cannot successfully request access to the specified permission.	
	FAP_PRM.1.1 The TSF shall restrict the ability of applications to request [assignment: *permissions defined as system permission by the TOE developer*] to only preloaded applications and preinstalled applications created by the TOE developer.	*Y*
	Permissions that are categorized at a Protection level of "signature" can only be assigned to apps that are signed by the platform signing key. These permissions are not accessible by any apps that are not signed with this key and so cannot be requested.	

Application Risk

FPA_RSK.1	No sensitive data should be stored outside of the app container or system credential storage facilities.	Supported? (Y/N)
	Output from a script that checks for proper usage and a report on any apps that could not provide a clear answer from the evaluator.	
	FPA_RSK.1.1 The applications on the TOE shall only store data within their own storage context. **FPA_RSK.1.2** The applications on the TOE shall only store keys using the TSF-provided key storage.	*Y*
	Tested as part of the Preloaded App Scripts	

FPA_RSK.2	The app does not rely on symmetric cryptography with hardcoded keys as a sole method of encryption.	Supported? (Y/N)
	Output from a script that checks for proper usage and a report on any apps that could not provide a clear answer from the evaluator.	
	FPA_RSK.2.1 The applications on the TOE shall use appropriate cryptographic primitives to support the algorithms in use. **FPA_RSK.2.2** The applications on the TOE shall not use deprecated cryptographic modes or algorithms.	*Y*
	Tested as part of the Preloaded App Scripts	

FPA_RSK.3	The app uses cryptographic primitives that are appropriate for the particular use-case, configured with parameters that adhere to industry best practices.	Supported? (Y/N)
	Output from a script that checks for proper usage and a report on any apps that could not provide a clear answer from the evaluator.	
	FPA_RSK.3.1 The applications on the TOE shall not have hardcoded symmetric keys as the method of internal data protection.	*Y*
	Tested as part of the Preloaded App Scripts	

FPA_RSK.4	Data is encrypted on the network using TLS. The secure channel is used consistently throughout the app.	Supported? (Y/N)
	Output from a script that checks for proper usage and a report on any apps that could not provide a clear answer from the evaluator.	
	FPA_RSK.4.1 The applications on the TOE shall not have hardcoded unencrypted URLs for network communications.	*Y*
	Tested as part of the Preloaded App Script	

FPA_RSK.5	The TLS settings are in line with current best practices, or as close as possible if the mobile operating system does not support the recommended standards.	Supported? (Y/N)
	Output from a script that checks for proper usage and a report on any apps that could not provide a clear answer from the evaluator.	
	FPA_RSK.5.1 The applications on the TOE shall use the trusted channels provided by FTP_ITC_EXT.1/HTTPS or FTP_ITC_EXT.1/TLS.	*Y*
	Tested as part of the Preloaded App Scripts	

FPA_RSK.6	The app verifies the X.509 certificate of the remote endpoint when the secure channel is established.	Supported? (Y/N)
	Output from a script that checks for proper usage and a report on any apps that could not provide a clear answer from the evaluator.	
	FPA_RSK.6.1 The applications on the TOE shall validate the X.509 server certificate according to the following rules: ● The certificate path must terminate with a certificate in the TOE trust anchor; ● The TOE shall use the main OS-provided method to verify the certificate.	*Y*
	Tested as part of the Preloaded App Scripts	

FPA_RSK.7	All inputs from external sources and the user are validated and if necessary sanitized. This includes data received via the UI, IPC mechanisms such as intents, custom URLs, and network sources.	Supported? (Y/N)
	Output from a script that checks for proper usage and a report on any apps that could not provide a clear answer from the evaluator.	
	FPA_RSK.7.1 The applications on the TOE shall sanitize all input from external sources via any available interface.	*Y*
	Tested as part of the Preloaded App Scripts	

FPA_RSK.8	The app does not export sensitive functionality via custom URL schemes, unless these mechanisms are properly protected. Apps by a common app developer may be acceptable in some circumstances.	Supported? (Y/N)
	Output from a script that checks for proper usage and a report on any apps that could not provide a clear answer from the evaluator.	
	FPA_RSK.8.1 The applications on the TOE shall not support unauthorized direct access to data from external apps.	*Y*
	Tested as part of the Preloaded App Scripts	

FPA_RSK.9	The app is signed and provisioned with a valid certificate for the platform.	Supported? (Y/N)
	Output from a script that checks for proper usage and a report on any apps that could not provide a clear answer from the evaluator.	
	FPA_RSK.9.1 The applications on the TOE shall be signed with certificates with a signature format valid for the platform.	*Y*
	Tested as part of the Preloaded App Scripts	

FPA_RSK.10	The app has been built in release mode, with settings appropriate for a release build (e.g. non-debuggable).	Supported? (Y/N)
	Output from a script that checks for proper usage and a report on any apps that could not provide a clear answer from the evaluator.	
	FPA_RSK.10.1 The applications on the TOE shall not have any debug functionality enabled.	*Y*
	Tested as part of the Preloaded App Scripts	

APA_LST.1	Enumerating the preloaded and preinstalled applications with system permissions is necessary to ensure how apps are separate from the OS.	Supported? (Y/N)

	The developer shall list the preloaded apps and any preinstalled apps (ones that are downloaded during the install) that are assigned system permissions.	
	APA_LST.1.1D The developer shall provide a list of all preloaded and preinstalled applications with system permissions included on the consumer mobile device at the completion of the initial CMD setup. **APA_LST.1.1C** The list of preloaded applications shall include all applications contained within the system software. **APA_LST.1.2C** The list of preinstalled applications shall include all applications installed on the consumer mobile device at the completion of the initial CMD setup that have been assigned system permissions. **APA_LST.1.1E** The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.	*Y*
	All preloaded apps' granted permission could be emulated via Setting -> APP -> All apps -> <app in question> -> Permissions	

TS 103 732-5 SFRs

Bootloader - User data protection

FDP_ULK.1	If the bootloader is able to be unlocked, user data shall be protected (though wipe).	Supported? (Y/N)
	Describe whether the bootloader can be unlocked, and if so, how the device is wiped when the bootloader state changes. (Locking does not require a wipe, only unlocking)	
	FDP_ULK.1.1 The TSF shall ensure that [selection: *the bootloader cannot be unlocked*].	*Y*
	The production device's bootloader cannot be unlocked.	

FDP_BBY.1	Any boot modes must continue to enforce user data security (no bypass)	Supported? (Y/N)
	Describe protections that ensure that alternative boot modes don't bypass the security functions. ***Note this normally would focus on how the data encryption can not be bypassed (i.e. the device booted to user data without any authentication).***	
	FPT_BBY.1.1 The TSF shall be enforced in any alternative boot modes.	*Y*
	There are no boot modes that can bypass the data security.	

Bootloader - Protection of the TSF

FPT_BLP.1	The bootloader must run in the least privileged mode (ARM EL1 for example) possible.	Supported? (Y/N)
	Provide documentation about the boot process and the modes the bootloader runs in, with particular focus on the last stage. If the bootloader is completely removed from memory after loading the OS, then that is sufficient to meet this.	
	FPT_BLP.1.1 The bootloader shall be configured to run in the least privileged mode of the processor.	*Y*
	The bootloader initially loads at EL3 (the first stage of the bootloader), but once it has completed the loading process moves to EL1 to proceed with the boot process	

FPT_PRT.1	The partition configuration protection shall be specified to ensure that the system is properly defined.	Supported? (Y/N)
	Provide information about how the integrity of the partition configuration is ensured.	

	FPT_PRT.1.1 The TSF shall protect the integrity of the partition configuration to prevent changes outside of the system image update process.	*Y*
	The integrity of the partition table is maintained using dm-verity	

FPT_PRT.2	Bootable partitions must be documented, including all settings used on bootable partitions.	
	Document the partitions of the device. Include both fstab (or equivalent partition table from the device) for all mounted partitions and any additional information for partitions that may not be mounted by the OS but are used for special operations.	Supported? (Y/N)
	FPT_PRT.2.1 The TOE has the following partitions marked as bootable: the main OS and [selection: *recovery*]. **FPT_PRT.2.2** The TSF enforces restrictions on writing data, code execution and system permissions through the use of partition flags during the mounting of partitions for use by the TOE.	*Y*
	AOSP part bootable partitions refers to https://source.android.com/docs/core/architecture/partitions . *Our fstab is：* *system /system ext4 ro,barrier=1,discard wait,slotselect,avb=vbmeta_system,logical,first_stage_mount,avb_keys=/avb/q-gsi.avbpubkey:/avb/r-gsi.avbpubkey:/avb/s-gsi.avbpubkey:/avb/t-gsi.avbpubkey:/avb/u-gsi.avbpubkey:/avb/v-gsi.avbpubkey* *system_ext /system_ext ext4 ro,barrier=1,discard wait,slotselect,avb=vbmeta_system,logical,first_stage_mount* *product /product ext4 ro,barrier=1,discard wait,slotselect,avb=vbmeta_system,logical,first_stage_mount* *vendor /vendor ext4 ro,barrier=1,discard wait,slotselect,avb=vbmeta,logical,first_stage_mount* *vendor_dlkm /vendor_dlkm ext4 ro,barrier=1,discard wait,slotselect,avb=vbmeta,logical,first_stage_mount* *system_dlkm /system_dlkm ext4 ro,barrier=1,discard wait,slotselect,avb=vbmeta,logical,first_stage_mount* *odm /odm ext4 ro,barrier=1,discard wait,slotselect,avb=vbmeta,logical,first_stage_mount* */dev/block/by-name/boot /boot emmc defaults slotselect,avb=vbmeta,first_stage_mount*	

	/dev/block/by-name/init_boot /init_boot emmc defaults slotselect,avb=vbmeta,first_stage_mount /dev/block/by-name/vendor_boot /vendor_boot emmc defaults slotselect,avb=vbmeta,first_stage_mount /dev/block/by-name/dtbo /dtbo emmc defaults slotselect,avb=vbmeta,first_stage_mount /dev/block/by-name/recovery /recovery emmc defaults slotselect,avb=vbmeta,first_stage_mount /dev/block/by-name/metadata /metadata f2fs noatime,nosuid,nodev,discard wait,check,formattable,first_stage_mount /dev/block/bootdevice/by-name/dataext /dataext ext4 ro,barrier=1,discard wait,slotselect,avb=vbmeta /dev/block/bootdevice/by-name/persist /mnt/vendor/persist ext4 noatime,nosuid,nodev,barrier=1 wait /dev/block/bootdevice/by-name/userdata /data f2fs noatime,nosuid,nodev,discard,reserve_root=32768,resgid=1065,fsync_mode=nobarrier,inlinecrypt latemount,wait,check,formattable,fileencryption=aes-256-xts:aes-256-cts:v2+inlinecrypt_optimized+wrappedkey_v0,keydirectory=/metadata/vold/metadata_encryption,metadata_encryption=aes-256-xts:wrappedkey_v0,quota,reservedsize=128M,sysfs_path=/sys/devices/platform/soc/1d84000.ufshc,checkpoint=fs /dev/block/bootdevice/by-name/misc /misc emmc defaults defaults /devices/platform/soc/8804000.sdhci/mmc_host* /storage/sdcard1 vfat nosuid,nodev wait,voldmanaged=sdcard1:auto,encryptable=footer /devices/platform/soc/*.ssusb/*.dwc3/xhci-hcd.*.auto* /storage/usbotg vfat nosuid,nodev wait,voldmanaged=usbotg:auto /dev/block/bootdevice/by-name/modem /vendor/firmware_mnt vfat ro,shortname=lower,uid=1000,gid=1000,dmask=227,fmask=337,context=u:object_r:firmware_file:s0 wait,slotselect /dev/block/bootdevice/by-name/dsp /vendor/dsp ext4 ro,nosuid,nodev,barrier=1 wait,slotselect /dev/block/bootdevice/by-name/vm-bootsys /product/vm-system ext4 ro,nosuid,nodev,barrier=1 wait,slotselect /dev/block/bootdevice/by-name/vm-persist /mnt/product/vm-persist ext4 noatime,nosuid,nodev,barrier=1 wait	

	/dev/block/bootdevice/by-name/bluetooth /vendor/bt_firmware vfat ro,shortname=lower,uid=1002,gid=3002,dmask=227,fmask=337, context=u:object_r:bt_firmware_file:s0 wait,slotselect /dev/block/bootdevice/by-name/qmcs /mnt/vendor/qmcs vfat noatime,nosuid,nodev,context=u:object_r:vendor_qmcs_file:s0 wait,check,formattable /dev/block/bootdevice/by-name/spunvm /mnt/vendor/spunvm vfat noatime,nosuid,nodev,context=u:object_r:vendor_spunvm_file:s0 wait,check,formattable	

FPT_ROL.1	Bootloaders must enforce rollback protection for firmware on the device. Tamper-evident storage must support the rollback implementation.	Supported? (Y/N)
	Provide documentation about how rollback protection is implemented and which components support it (and any conditions under which rollback may be bypassed and an earlier version installed).	
	FPT_ROL.1.1 The TSF shall permit rollback of the system software and [selection: *no other software/firmware*] under [selection: the case where the bootloader has been unlocked] conditions.	*Y*
	System software can only be rolled back when the bootloader has been unlocked. The production devices are locked. In the unlocked state any system image may be installed. Lenovo followed Android AVB rollback protection mechanism	

Bootloader - Functional Specification

ADV_FSP.2	Boot arguments/commands that may be passed to the kernel from the boot process shall be documented.	Supported? (Y/N)
	The focus here is on commands that can be provided outside of the normal programmed commands, so commands from the user that may be passed. This is a modification to the ADV_FSP.2 documentation that is already required as part of the evaluation.	
	As part of the functional specification, the interface between the bootloader and the kernel shall be considered as a TSFI. Boot arguments or commands that may be passed from the bootloader to the kernel outside those that are provided as part of the TOE configuration (such as arguments that may be passed by the user through an external connection to the device) shall be documented and reviewed.	*Y*
	OEM response Lenovo adds a few internal fastboot oem arguments specific to	

	the device beyond the normal ones found in AOSP. The commands available do not bypass any security functionality on the device such as enabling a bypass of the authentication process to unlock user data.	

Root of Trust - Protection of the TSF

FPT_INI.1	The device provides a method for verifying the integrity of the device during the boot process (a Root of Trust).	Supported? (Y/N)
	Document what the Root of Trust is, how it provides support for the initialization and what happens when an error is detected.	
	FPT_INI.1.1 The TOE shall provide an initialization function which is self-protected for integrity and authenticity. **FPT_INI.1.2** The TOE initialization function shall ensure that certain properties hold on certain elements immediately before establishing the TSF in a secure initial state, as specified in FPT_INI.1.2 Table. ID1 [*INTEGRITY*] [*ROOT OF TRUST*] ID2 [*PREVENTION OF DOWNGRADE TO PREVIOUS VERSIONS*] [*ELEMENTS AS SPECIFIED IN FPT_ROL.1.1*] **FPT_INI.1.3** The TOE initialization function shall detect and respond to errors and failures during initialization such that the TOE [selection: *is halted*]. **FPT_INI.1.4** The TOE initialization function shall only interact with the TSF in [assignment: *boot time*] during initialization.	*Y*
	A public key on the device used to verify the initial bootloader is considered the Root of Trust on the device. *This key is fused into the SoC into special OTP eFuses which cannot be changed once they have been set maintaining the integrity of the key.* *The device utilizes Android Verified Boot 2.0 to ensure the integrity of the system as it loads on the device.* *The trust of the boot sequence is derived from the public signature key hash which is set into One Time Programmable (OTP) eFuses in the SoC. This hash is used to verify that the signature provided as part of the vmbeta partition is valid (and hence that the partition can be trusted).*	

FPT_RDI.1	The integrity of the stored Root of Trust data is monitored.	

	Explain how the data used by the Root of Trust is monitored for integrity.	Supported? (Y/N)
	FPT_RDI.1.1 The TSF shall monitor Root of Trust data stored in containers controlled by the TSF for integrity errors.	*Y*
	A public key on the device used to verify the initial bootloader is considered the Root of Trust on the device. *The RoT key is not specifically monitored by a program but is stored into a set of OTP eFuses which cannot be changed once set.* *Integrity is implied when the bootloader image is successfully verified using the programmed key. If that fails, it will be a problem with the image*	

GSMA Requirements (FS.56)

Cryptographic Support

FCS_STG_EXT.1	Developers must provide a keystore that is tied to the hardware outside the main OS.	Supported? (Y/N)
	Provide documentation of how the keys are protected in a key store outside the main OS and how this is tied specifically to the hardware.	
	FCS_STG_EXT.1.1 The TSF shall provide [selection: *hardware-based*] secure cryptographic key storage. NOTE: Hardware-based => TEE or similar Hardware isolated => eSE, SPU or similar **FCS_STG_EXT.1.2** The TSF shall utilize the provided secure cryptographic key storage for protecting the key hierarchy.	*Y*
	The device has a hardware-based key store. This is used for the Android keystore as to where to store the key.	

Identification and Authentication

FIA_SAR.1	Biometric authentication shall meet minimum requirements to detect spoof attempts.	Supported? (Y/N)
	Document the SAR (or IAPMR) information for each biometric modality available.	
	FIA_SAR.1.1 The TSF shall provide a biometric verification mechanism with the SAR no exceeding [selection: • *BELOW 7%*	***Y***
	The SAR is lower than 7%.	

Privacy

FPR_ANO.2	The OTA client shall not access user data that is not necessary to perform an update.	Supported? (Y/N)
	Document what data is needed by the OTA client (such as IMEI or email to be authorized to access the update) and how the permissions on the device protect the client from user data. Since the client will have high permissions, showing how the user data is protected from the OTA client (or the client is prevented from accessing the data) is to be shown.	

	FPR_ANO.2.1 The TSF shall ensure that [THE SYSTEM SOFTWARE OTA CLIENT] is unable to determine the real user data bound to [the TOE (HARDWARE PLATFORM)]. **FPR_ANO.2.2** The TSF shall provide [SYSTEM SOFTWARE UPDATES] to [THE USER] without soliciting any reference to the real user data.	
	The OTA client is part of AOSP (https://cs.android.com/android/_/android/platform/system/update_engine) and maintained as part of that project. OTA client undergoes a privacy review with every release. The review ensures that the client does not export personal dentification information of the user. Additionally the client is not provided privileges which would provide it access to user/app data.	***Y***

Protection of the TSF

FPT_EAT_EXT.1	Access to telephony commands (such as AT commands or other terminal listeners) via USB or other external interfaces must be disabled at ship time. This is intended to cover commands such as those entered via the dialer to provide various device information and not access to the modem itself (such as programmatic access).	Supported? (Y/N)
	Document how telephony commands can be accessed (both locally through the interfaces and remotely, if possible, from external devices).	
	FPT_EAT_EXT.1.1 The TSF shall only allow access to AT modem commands from [selection: *the user interface]*]. **FPT_EAT_EXT.1.2** The TSF shall prompt for approval of AT modem commands sent to the device from outside the user interface [selection: *is not applicable*].	*Y*
	WiFi only product and has removed functions related to modem	

FPT_LNW_EXT.1	Root or system owned processes do not expose any listening network sockets externally or on loopback interfaces. Disabling a listening socket must not require an OTA.	Supported? (Y/N)
	Document the network sockets that are available after the boot has completed from the device. Both loopback and external open ports must be documented.	
	FPT_LNW_EXT.1.1 The TSF shall ensure that no listening network sockets to the external or loopback IP networks are	*Y*

	associated with processes with system permissions on the device. **FPT_LNW_EXT.1.2** The TSF shall ensure that [assignment: No *Network sockets and associated processes*] exposed to the external or loopback IP networks have no system permission on the device.	
	Android native mechanism has supported it	

Life Cycle Requirements

The highlighted text are changes from what is included in the TS 103 732-1 SARs.

ALC_CMC.2	Any third party code shall be added to the developer's CM system.	
	The process for importing third party code and then maintaining that code shall be described.	
	ALC_CMC.2.4D The developer shall provide guidance for importing and updating external software components into the CM system. **ALC_CMC.2.4C** The CM documentation shall describe the method used to identify external software components and how those components are updated.	*Y*
	all external code is bought into Lenovo internal repository with verification process.	

ALC_CMS.2	Any third party software components shall be documented where applicable.	
	For external software components, the original maintainer of the software shall be specified.	
	ALC_CMS.2.1C The configuration list shall include the following: the TOE itself; the evaluation evidence required by the SARs; and the parts that comprise the TOE including any external software components. **ALC_CMS.2.2E** The evaluator shall confirm that for external software components, the developer shall include the source and original maintainer of the component.	*Y*
	All external components that are included in the device are tracked in the CM system and included as part of the SBOM for the device. The source of all external components is maintained as part of that inclusion into the internal CM system.	

ALC_DVS_EXT.1	The developer shall describe the process for generating the signing keys and unique identifiers on the device (ones that are fixed).	Supported? (Y/N)

	The developer needs to provide documentation about how the identifiers are generated and put on the device. This includes how these are secured while in the factory and cleared when not needed. The developer must also describe how signing keys are generated, stored and protected. This includes how they are used (procedures for ensuring rogue builds cannot be created). Here the term keybox is used to identify the location on the device where the unique keys for the device will be stored. This can be in various hardware layers below the OS (the SEE). ***Note that the highlighted sections are where these are changes from the PP. Other sections without changes are not included.***	
	ALC_DVS_EXT.1.1D The developer shall produce and provide development security documentation on the generation, ~~and~~ protection and use of signing keys ~~and device-unique identifiers~~. **ALC_DVS_EXT.1.2D** The developer shall produce and provide development security documentation on the acquisition or generation of device unique identifiers. **ALC_DVS_EXT.1.3D** The developer shall produce and provide development security documentation on the provisioning of data or keys that may be used by a Root of Trust. **ALC_DVS_EXT.1.1C** The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the ~~following manufacturing components:~~ keys used to sign the publicly released system software and its updates. **ALC_DVS_EXT.1.2C** The development security documentation shall describe the procedures for selecting the proper signing keys used for a device and to ensure the use of the proper keys in the build process. **ALC_DVS_EXT.1.3C** The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the ~~following manufacturing components:~~ unique, non-modifiable identifiers (such as IMEI, attestation keys or Device Unique Keys) and how they are properly acquired/created and provisioned for each device. **ALC_DVS_EXT.1.4C** The development security documentation shall describe all the physical, procedural, personnel, and other	Y

	security measures that are necessary to protect the confidentiality and integrity of the ~~following manufacturing components:~~ data and keys provisioned to the device for a Root of Trust for the device.	
	DEVELOPMENT SECURITY DOCUMENTATION **Target Evaluation:** MDSCert L3 Certification **Control Family:** ALC_DVS (Development Security) **Document Scope:** Security measures and procedures for software signing key management across the development and manufacturing lifecycle. **1. ALC_DVS_EXT.1.1C – Security Measures for Manufacturing Components** **Requirement:** The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of manufacturing components: keys used to sign the publicly released system software and its updates. **1.1 Physical Security Measures** • **Isolated Infrastructure:** Production signing keys are stored exclusively on physically isolated (air-gapped) servers within a secured data facility. • **Continuous Surveillance:** The server room housing the signing infrastructure is under 24/7 continuous video surveillance with tamper-evident recording retention. • **Physical Access Control:** Entry to the secure zone requires authenticated physical security tokens. Access is strictly logged, audited, and limited to explicitly authorized personnel. **1.2 Personnel Security Measures** • **Role-Based Access Control (RBAC):** Remote and local access to signing infrastructure is restricted to a minimal set of authorized security engineers and build administrators on a strict need-to-know basis. • **Authorization & Auditing:** All access requests undergo formal approval. Session logs, command histories, and access attempts are centrally collected and reviewed quarterly for anomaly detection. **1.3 Procedural & Technical Security Measures** • **Key Generation Compliance:** Signing keys are generated strictly following SoC vendor-recommended cryptographic	

	algorithms, parameters, and secure generation procedures. • **Production Artifact Restriction:** Production firmware builds only contain cryptographically signed binaries and the cryptographic hash of the public signing key/certificate. Raw private key material is never embedded in, exported to, or accessible from production devices. • **Secure Distribution:** Signed system software is transmitted to manufacturing facilities exclusively via dedicated secure network lines. All transit artifacts are additionally signed using GPG keys to guarantee end-to-end integrity and confidentiality. • **Key Lifecycle Management:** Key rotation, backup, and destruction follow documented cryptographic lifecycle procedures aligned with industry best practices and SoC vendor guidelines. **2. ALC_DVS_EXT.1.2C – Key Selection and Build Process Procedures** **Requirement:** The development security documentation shall describe the procedures for selecting the proper signing keys used for a device and to ensure the use of the proper keys in the build process. **2.1 Key Generation & Component Assignment** **All cryptographic signing keys are generated in strict compliance with SoC vendor-specified algorithms, parameters, and secure generation procedures. To maintain a verifiable chain of trust and prevent cryptographic misassignment, signing responsibilities are explicitly partitioned as follows:** • **Secure Boot Key: Used exclusively to sign all SoC-specified firmware components mandated by the silicon vendor. This key establishes the hardware-rooted root of trust during the earliest boot stages.** • **AVB (Android Verified Boot) Key: Used to sign all Android-related system partitions and components, including but not limited to dtbo, bootloader, system, vendor, vendor_dlkm, and system_dlkm. This ensures the cryptographic integrity and authenticity of the Android OS stack throughout the verified boot process.**	

2.2 Production vs. Development/Engineering Key Process

A strict cryptographic segregation policy is enforced between production/release keys and engineering/debug keys to prevent unauthorized or accidental signing of production firmware:

Key Type	Development/Engineering Use	Production/Release Use
AVB (Android Verified Boot) Keys	Separate engineering keys are provisioned for internal development, testing, and debugging. These keys are distributed only to developer workstations and internal CI runners with restricted signing capabilities.	Production AVB keys are strictly isolated and accessible **only** within the secure, automated CI/CD pipeline. Standard developers have zero access to production AVB key material.
Secure Boot Keys	A unified key set is utilized across both development and production environments. Procedural controls, build configuration flags, and pipeline isolation ensure consistent application without exposing raw private keys.	Same unified key set is used. Access is restricted to authorized build administrators, and signing operations are executed exclusively in controlled, audited build environments.

2.3 Build Process Integration & Enforcement

- **Automated Key Selection:** The CI/CD pipeline automatically selects the appropriate signing key based on the build configuration (BUILD_TYPE=production vs BUILD_TYPE=development). Manual key selection is prohibited in release branches.
- **Pipeline Isolation:** Production signing operations occur in isolated, ephemeral build runners that fetch credentials from a secure secrets manager at runtime. Credentials are never persisted on disk or exposed to developer consoles.
- **Pre-Release Validation:** Automated cryptographic verification steps run post-build to confirm that:
    - The correct production key was used for

	release artifacts. • No development/debug keys are embedded in production builds. • Signature verification against the embedded public key hash succeeds. • **Audit Trail:** Every signing operation is logged with timestamp, build ID, key identifier, operator/pipeline identity, and cryptographic checksum of the signed output. Logs are retained for the full product lifecycle and are subject to internal and external audit.	

ALC_FLR.3	The developer shall have a process for receiving information about flaws and then provide patches for those flaws to the impacted devices. In addition this includes a defined period and frequency of updates to the device (including both OS updates and regular security patches).	
	The developer needs to provide documentation about the flaw remediation/patching process. This can include public sites/information along with the support information for the device under evaluation. ***Note that the highlighted sections are where these are changes from the PP.***	Supported? (Y/N)
	ALC_FLR.3.1D The developer shall document and provide flaw remediation procedures addressed to TOE manufacturers. **ALC_FLR.3.2D** The developer shall establish a procedure for accepting and acting upon all reports of security flaws and requests for corrections to those flaws. **ALC_FLR.3.3D** The developer shall provide flaw remediation guidance addressed to TOE users. **ALC_FLR.3.4D** The developer shall provide public guidance related to the duration period, frequency and type of updates that will be released to support the TOE. **ALC_FLR.3.5D** The developer shall provide a public vulnerability disclosure program to provide security bulletins about the flaws that have been remediated. **ALC_FLR.3.6D** The developer shall establish procedures for ensuring that no known security vulnerabilities rated as High and Critical (e.g. as classified in public databases) are included in the TOE at public release.	Y

	ALC_FLR.3.1C The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE. **ALC_FLR.3.2C** The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw. **ALC_FLR.3.3C** The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws. **ALC_FLR.3.4C** The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users. ~~The flaw remediation procedures documentation shall also define the planned minimum length of time after release of the TOE that these methods will be used to maintain the TOE.~~ **ALC_FLR.3.5C** The flaw remediation procedures shall describe a means by which the developer receives from TOE users reports and enquiries of suspected security flaws in the TOE. **ALC_FLR.3.6C** The flaw remediation procedures shall include a procedure requiring timely response and the automatic distribution of security flaw reports and the associated corrections to registered users who might be affected by the security flaw. **ALC_FLR.3.7C** The procedures for processing reported security flaws shall ensure that any reported flaws are remediated and the remediation procedures issued to TOE users. **ALC_FLR.3.8C** The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws. **ALC_FLR.3.9C** The flaw remediation guidance shall describe a means by which TOE users report to the developer any suspected security flaws in the TOE. **ALC_FLR.3.10C** The flaw remediation guidance shall describe a means by which TOE users may register with the developer, to be eligible to receive security flaw reports and corrections. **ALC_FLR.3.11C** The flaw remediation guidance shall identify the specific points of contact for all reports and enquiries about security issues involving the TOE. **ALC_FLR.3.12C** The flaw remediation procedures documentation shall define the planned minimum duration after	

	release of the TOE that these methods will be used to maintain the TOE. **ALC_FLR.3.13C** The flaw remediation procedures documentation shall define the types of updates (such as security/maintenance or operating system) and the frequency of these updates being provided for the TOE. **ALC_FLR.3.14C** The flaw remediation procedures documentation shall describe the process for publicly releasing security flaw remediation information, including the location(s) where this will be publicly available. **ALC_FLR.3.15C** The flaw remediation procedures documentation shall describe the process for verifying known security flaws are not propagated into a new (not yet released) TOE. **ALC_FLR.3.1E** The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.	
	Lenovo has a well-managed process to response on privacy issues and vulnerabilities as described in evidence document. Besides, Users can receive security flaw reports and corrections by accessing to the security patch maintenance webpage: Disclosure page : https://support.lenovo.com/gb/en/product_security/home Disclosure Policy : https://support.lenovo.com/us/en/product_security/vulnerability-disclosure-policy	