Google LLC.

Android Kernel Cryptographic Module

Software Version: 5.10.66-android12-9-00072-g143ac63130f0-ab7955824

FIPS 140-3 Non-Proprietary Security Policy

Document Version: 0.4

Last Update Date: 6-27-2024

Table of Contents

1. General

This document is the non-proprietary FIPS 140-3 Security Policy for the Android Kernel Cryptographic Module from Google LLC. It contains a specification of the rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-3 (Federal Information Processing Standards Publication 140-3) for a Security Level 1 Software cryptographic module. This security policy is for the validation of the Android Kernel Cryptographic Module.

In this document, the terms “Android Kernel Cryptographic Module”, “AKC”, “cryptographic module” or “module” are used interchangeably to refer to the module’s software version 5.10.66-android12-9-00072-g143ac63130f0-ab7955824.

Below is a table indicating the individual clause levels.

ISO/IEC 24759 Section 6 [Number Below]	FIPS 140-3 Section Title	Security Level
1	General	1
2	Cryptographic module specification	1
3	Cryptographic module interfaces	1
4	Roles, services, and authentication	1
5	Software/Firmware security	1
6	Operational environment	1
7	Physical security	N/A
8	Non-invasive security	N/A
9	Sensitive security parameter management	1
10	Self-tests	1
11	Life-cycle assurance	1
12	Mitigation of other attacks	N/A

Table 1 - Security Levels

The module is designed to meet an overall security level of 1.

2. Cryptographic module specification

The Android Kernel Cryptographic Module is a multi-chip standalone software only module designed to provide encryption services for the Linux kernel of the device. The module is implemented as a self-contained Linux Kernel Module (LKM).

The module has been tested on the following platforms:

#	Operating System	Hardware Platform	Processor	PAA/Acceleration
1	Linux kernel 5.10	Google Pixel 6	Google Tensor processor	with PAA
2	Linux kernel 5.10	Google Pixel 6	Google Tensor processor	without PAA

Table 2 - Tested Operational Environments

The CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when ported to an operational environment which is not listed on the validation certificate.

Mode of operation

When the module starts up successfully, after passing the pre-operational self-test and the cryptographic algorithms self-tests (CASTs), the module is operating in the approved mode of operation by default and can only be transitioned into the non-approved mode by calling one of the non-approved services listed in Table 9. Section 4 provides details on the service indicator is implemented by the module. The service indicator identifies when an approved service is called. The Crypto Officer shall not configure the use of non-approved algorithms while the module is operating in an approved mode. If a non-Approved algorithm is used, then the module is operating in a non-Approved mode. Prior to using any of the non-approved services, the Crypto Officer shall zeroize all CSPs which places the module into the non-approved mode of operation.

Table 3 below lists all Approved or Vendor-affirmed security functions of the module, including specific key size(s) -in bits unless otherwise noted- employed for approved services, and implemented modes of operation. There are algorithms, modes, and key moduli sizes that have been CAVP-tested but are not used by any approved services of the module. Only the algorithms, modes/methods, and key lengths/curves/moduli shown in the tables below are used by an approved service of the module.

CAVP Cert	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
#A2268	AES [FIPS 197, SP 800 38A]	AES-CBC	128, 192, 256 bits	Data encryption/decryption
#A2268	AES [FIPS 197, SP 800 38A]	AES-ECB	128, 192, 256 bits	Data encryption/decryption
#A2268	AES [FIPS 197, SP 800 38A]	AES-CBC-CS3	128, 192, 256 bits	Data encryption/decryption
#A2268	AES [FIPS 197, SP 800 38A]	AES-CTR	128, 192, 256 bits	Data encryption/decryption
#A2268	AES [FIPS 197, SP 800 38A]	AES-XTS	128, 192, 256 bits	Data encryption/decryption
#A2268	AES [FIPS 197, SP 800 38B]	AES-CMAC	128, 192, 256 bits	MAC generation and verification
N/A	ENT (NP) [SP800-90B]	N/A	N/A	Non-physical entropy source used for seeding DRBG
#A2268	DRBG [SP800-90Ar1]	HMAC-DRBG (HMAC-SHA-1)	N/A	Random number generation
#A2268	DRBG [SP800-90Ar1]	HMAC-DRBG (HMAC-SHA2-256)	N/A	Random number generation
#A2268	DRBG [SP800-90Ar1]	HMAC-DRBG (HMAC-SHA2-384)	N/A	Random number generation
#A2268	DRBG [SP800-90Ar1]	HMAC-DRBG (HMAC-SHA2-512)	N/A	Random number generation
#A2268	HMAC [FIPS 198-1]	HMAC-SHA-1	112 bits or greater	Message authentication

CAVP Cert	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
#A2268	HMAC [FIPS 198-1]	HMAC-SHA2-224	112 bits or greater	Message authentication
#A2268	HMAC [FIPS 198-1]	HMAC-SHA2-256	112 bits or greater	Message authentication
#A2268	HMAC [FIPS 198-1]	HMAC-SHA2-384	112 bits or greater	Message authentication
#A2268	HMAC [FIPS 198-1]	HMAC-SHA2-512	112 bits or greater	Message authentication
#A2268	SHS [FIPS 180-4]	SHA-1 Message Length: 0-65528 Increment 8	N/A	Message Digest Note: SHA-1 is not used for digital signature generation
#A2268	SHS [FIPS 180-4]	SHA2-224 Message Length: 0-65528 Increment 8	N/A	Message digest
#A2268	SHS [FIPS 180-4]	SHA2-256 Message Length: 0-65528 Increment 8	N/A	Message digest
#A2268	SHS [FIPS 180-4]	SHA2-384 Message Length: 0-65528 Increment 8	N/A	Message digest
#A2268	SHS [FIPS 180-4]	SHA2-512 Message Length: 0-65528 Increment 8	N/A	Message digest

Table 3 - Approved Algorithms

Table 4 below lists all non-Approved algorithms not allowed in the approved mode of operation implemented by the module.

Algorithm/Function	Use/Function
AES-GCM	Authenticated Encryption/Decryption
AES-XCBC	Message Authentication Code
AES-CBC-MAC	Message Authentication Code
ESSIV-CBC-AES	Salt Generation

Table 4 - Non-Approved Algorithms Not Allowed in the Approved Mode of Operation

In addition, the module does not implement Non-Approved Algorithms Allowed in Approved Mode of Operation and Non-Approved Algorithms Allowed in Approved Mode of Operation with No Security Claimed.

Cryptographic boundary

Figure 1 below depicts the module's Block Diagram. Please note that the bold RED rectangle in the block diagram represents the Tested Operational Environment's Physical Perimeter (TOEPP) containing the Module (the thin RED rectangle).

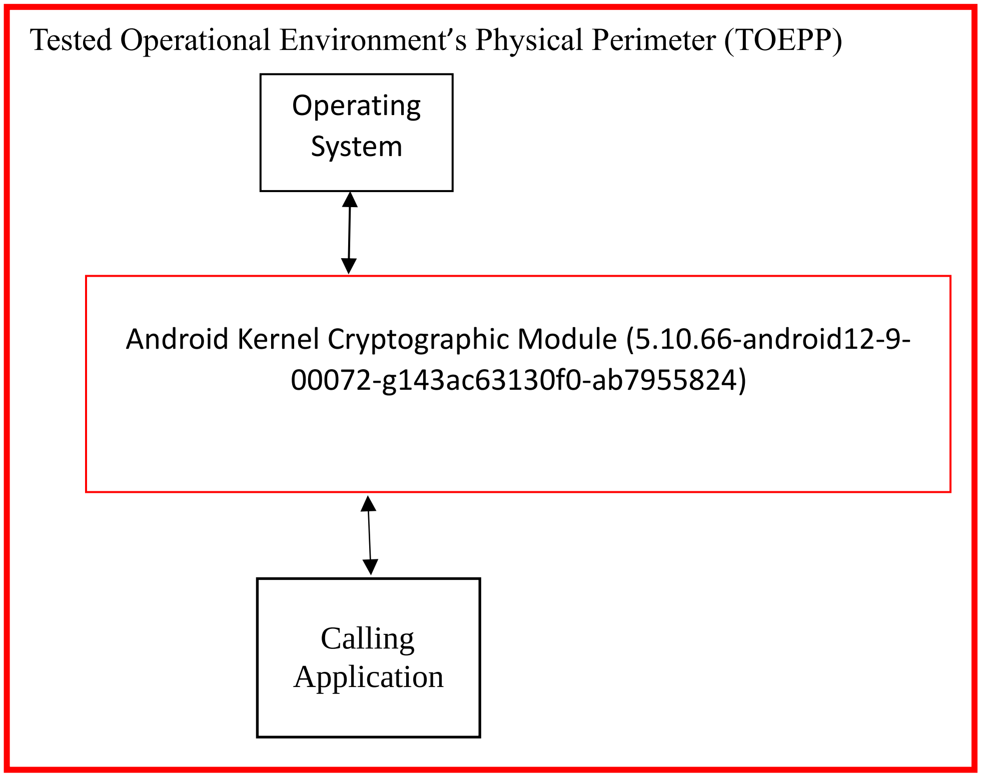
Tested Operational Environment's Physical Perimeter (TOEPP)
Operating System
Android Kernel Cryptographic Module (5.10.66-android12-9-00072-g143ac63130f0-ab7955824)
Calling Application

Figure 1 - Cryptographic Boundary

The module is a single object module (fips140.ko) which is implemented as a Linux Kernel Module and bundled into the tested device's boot image (vendor_boot.img).

3. Cryptographic module interfaces

Physical port	Logical interface	Data that passes over port/interface
N/A	Data Input Interface	Arguments for an API call that provide the data to be used or processed by the module
N/A	Data Output Interface	Output data returned to calling function
N/A	Control Input Interface	Arguments for an API call used to control and configure module operation
N/A	Status Output Interface	Return values from the Module's API used to obtain information on the status of the module. The Status Output Interface also includes the log file where the module messages are output
N/A	Control Output Interface	N/A

Table 5 - Ports and Interfaces

4. Roles, services, and authentication

Role	Service	Input	Output
Crypto Officer (CO)	Initialization	Command to start the initialization	Module initialization status
Crypto Officer (CO)	Symmetric Encryption and Decryption	Command to conduct the encryption and decryption operation	Encrypted or Decrypted message
Crypto Officer (CO)	Message Authentication Code (MAC)	Command to conduct the HMAC or AES-CMAC operation	MAC value
Crypto Officer (CO)	Message Digest	Command to conduct the Message Digest operation	Hashed message
Crypto Officer (CO)	Random Number Generation	Command to conduct the HMAC_DRBG generation	Random value
Crypto Officer (CO)	Perform Self-Test (Pre-operational Self-Tests and Conditional Self-Tests)	Command to conduct the self-tests	Status of self-tests
Crypto Officer (CO)	Show Status	Command to check the status	Module's current status
Crypto Officer (CO)	Show Version	Command to get module's software version	Module's name/ID and versions
Crypto Officer (CO)	Zeroization	Command to zeroize all SSPs	Status of zeroization completion

Table 6 - Roles, Service Commands, Input and Output

No authentication is required at security level 1 and the assumption of the role is implicit by the service being performed.

Role	Authentication Method	Authentication Strength
Crypto Officer (CO)	N/A	N/A

Table 7 - Roles and Authentication

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator
Initialization	Conduct module's initialization	N/A	N/A	Crypto Officer (CO)	N/A	N/A

Symmetric Encryption and Decryption	Conduct Symmetric Encryption and Decryption	AES-CBC;
		AES-ECB;
		AES-CBC-CS3;
		AES-CTR;
		AES-XTS
			AES Key	Crypto Officer (CO)	R, W, E	Approved service execution return value "true"

Message authentication code (MAC)	Conduct Key Hash operation	AES-CMAC;
		HMAC-SHA-1;
		HMAC-SHA2-224;
		HMAC-SHA2-256;
		HMAC-SHA2-384;
		HMAC-SHA2-512
			AES Key;
			HMAC Key
				Crypto Officer (CO)	R, W, E	Approved service execution return value "true"

Message Digest	Conduct Message	SHA-1;
		SHA2-224;
		SHA2-256;
			N/A	Crypto Officer (CO)	N/A	Approved service execution

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator
	digest operation	SHA2-384; SHA2-512				return value "true"

Random Number Generation	Conduct HMAC_DRBG generation	HMAC_DRBG	Entropy Input;
			DRBG Seed;
			DRBG V;
			DRBG Key
				Crypto Officer (CO)	R, W, E	Approved service execution return value "true"

Self-Test (Pre-operational Self-Tests and Conditional Self-Tests)	Run Pre-operational Self-Test and Conditional Algorithm Self-Tests	AES-CBC;
		AES-ECB;
		AES-CBC-CS3;
		AES-CTR;
		AES-XTS;
		AES-CMAC;
		DRBG;
		HMAC-SHA-1;
		HMAC-SHA2-224;
		HMAC-SHA2-256;
		HMAC-SHA2-384;
		HMAC-SHA2-512
		SHA-1;
		SHA2-224;
		SHA2-256;
		SHA2-384;
		SHA2-512
			Software Integrity Key (non-SSP)	Crypto Officer (CO)	R, E	Self-test completion message

Show Status	Provides the module's current status	N/A	N/A	Crypto Officer (CO)	N/A	N/A
Show Version	Provides the module's name/ID and versions	N/A	N/A	Crypto Officer (CO)	N/A	N/A
Zeroization	Zeroize the SSPs stored in the module	N/A	All SSPs	Crypto Officer (CO)	Z	Zeroize completion message

Table 8 – Approved Services

G = Generate: The module generates or derives the SSP.

R = Read: The SSP is read from the module (e.g. the SSP is output).

W = Write: The SSP is updated, imported, or written to the module.

E = Execute: The module uses the SSP in performing a cryptographic operation.

Z = Zeroize: The module zeroizes the SSP.

Service	Description	Algorithms Accessed	Role	Indicator
Authenticated Encryption/Decryption	Conduct Authenticated Encryption/Decryption	AES-GCM	N/A	Non-approved service execution return value “false”
Message Digest using AES-XCBC,	Conduct Message digest operation	AES-XCBC	N/A	Non-approved service execution return value “false”
Message Digest using AES-CBC-MAC	Conduct Message digest operation	AES-CBC-MAC	N/A	Non-approved service execution return value “false”
Salt Generation	Conduct Salt Generation operation	ESSIV-CBC-SHA256	N/A	Non-approved service execution return value “false”

Table 9 – Non-Approved Services

5. Software security

Integrity techniques

The module is software-only and performs self-tests during the loading of the module to ensure the integrity of the module and its services. To ensure software security, the module is protected by an HMAC-SHA2-256 (HMAC Cert. #A2268) algorithm. The software integrity test key (non-SSP) was preloaded to the module’s binary at the factory and used only for the pre-operational software integrity self-test. During initialization of the module, the integrity of the runtime executable is verified using an HMAC-SHA2-256 which is compared to a value computed at build time. If at load time the MAC does not match the stored, known MAC value, the module enters a critical error state where all crypto functionality is inhibited. The module must be reloaded to attempt the integrity test again.

Integrity test on-demand

The integrity test is performed as part of the Pre-Operational Self-Tests. It is automatically executed at power-on. The operator can power-cycle or reboot the tested platform to initiate the integrity test on-demand.

6. Operational environment

The module is operated in a modifiable operational environment per the definition in FIPS 140-3. The operation system shall be restricted to a single operator mode of operation (i.e., concurrent operators are explicitly excluded). The external application calling the module is a single user of the cryptographic module, even when the application is serving multiple clients.

7. Physical security

The module is software-only and so does not claim any physical security.

8. Non-invasive security

The module claims no non-invasive security techniques.

9. Sensitive security parameters management

Key/ SSP Name/ Type	Strength	Security Function and Cert. #	Generation	Import /Export	Establish -ment	Storage	Zero-isation	Use & related keys
AES Key	128, 192 or 256 bits	AES Cert. # A2268	N/A	Import: Electronic Entry (EE) via the API; Export: No	MD/EE	Stored in RAM memory of the tested platform	Automatic zeroization when the tested platform is powered down	Used for Symmetric Encryption, Decryption, and AES-CMAC generation and verification
HMAC Key	112 or greater	HMAC Cert. # A2268	N/A	Import: Electronic Entry (EE) via the API; Export: No	MD/EE	Stored in RAM memory of the tested platform	Automatic zeroization when the tested platform is powered down	Used for HMAC generation and verification
Entropy Input	At least 256 bits of security strength	DRBG Cert. # A2268	N/A	Import: Electronic Entry (EE) via the API; Export: No	MD/EE	Stored in RAM memory of the tested platform	Automatic zeroization when the tested platform is powered down	Used for DRBG Generation
DRBG Seed	256 bits	DRBG Cert. # A2268	N/A	Import: No; Export: No	N/A	Stored in RAM memory of the tested platform	Automatic zeroization when the tested platform is powered down	Used for DRBG Generation
DRBG V	256 bits	DRBG Cert. # A2268	N/A	Import: No; Export: No	N/A	Stored in RAM memory of the tested platform	Automatic zeroization when the tested platform is powered down	Used for DRBG Generation
DRBG Key	256 bits	DRBG Cert. # A2268	N/A	Import: No; Export: No	N/A	Stored in RAM memory of the tested platform	Automatic zeroization when the tested platform is powered down	Used for DRBG Generation

Table 10 – SSPs

Random number generation
The module implements an Approved SP 800-90Ar1 DRBG (HMAC-DRBG Cert. #A2268), which shall be called by the Application Client for key generation.

Key generation
The module does not provide any key generation service or perform key generation for any of its Approved algorithms. Keys/CSPs are passed in from calling applications via the algorithm API parameters.

Key entry and output
The module does not support manual key entry or key output. Keys/CSPs can only be exchanged between the module and the calling application via the appropriate algorithm API calls within the TOEPP.

Key storage
The SSPs are not stored inside the module. A pointer to a plaintext key is passed to the module through the algorithm APIs. Intermediate key/CSP storage locations are immediately zeroized in memory after use.

Key zeroization
The module is passed keys as part of a function call from a calling application and does not store keys persistently. All SSPs can be zeroized by powering down the tested platform.

RBG entropy source

Entropy sources	Minimum number of bits of entropy	Details
CPU Jitter Random Number Generator (Jitter Entropy Library v2.2.0)	0.908 bits/sample bit	CPU Jitter Random Number Generator (Jitter Entropy Library v2.2.0) from Stephen Muller provides at least 256 bits 256-bits of security strength entropy. Please see https://www.chronox.de/jent/doc/CPU-Jitter-NPTRNG.pdf for more information. The entropy source falls into IG 9.3.A, Scenario #1(b): A software module that contains an approved DRBG, that is seeded exclusively from one or more known entropy sources, located within the physical perimeter of the operational environment.

Table 11- Non-Deterministic Random Number Generation Specification

10. Self-tests

When the module is loaded or instantiated (after being powered off, rebooted, etc.), the module runs pre-operational self-tests. The operating system is responsible for the initialization process and loading of the library. The module is designed with a default entry point (DEP) which ensures that the self-tests are initiated automatically when the module is loaded. Prior to the module providing any data output via the data output interface, the module would perform and pass the pre-operational self-tests. Following the successful pre-operational self-tests, the module would execute the Conditional Cryptographic Algorithm Self-tests (CASTs).

The self-test success or failure is output as a return value of the library load API call, which is functioning as the self-test status indicator. If one of the self-tests fails, the module transitions into the error state (there is only one error state if any one of self-tests fails) and outputs the error message via the module's status output interface. While the module is in the error state, all data through the data output interface and all cryptographic operations are disabled. The error state can only be cleared by reloading the module. All self-tests must be completed successfully before the module transitions to the operational state.

Below are the details of the self-tests conducted by the module.

Pre-operational self-test

- HMAC-SHA2-256 KAT
- Software Integrity Test (using HMAC-SHA256)

During the build-time the hash value is calculated over the module (the original code and read-only data of the fips140.ko file). To calculate the hash of the loaded module, the segments of the code are concatenated to each other with the relocations unapplied. The hash of this code is then calculated and compared to the build-time hash value. If the value does not match the module will enter the error state.

Conditional algorithm self-tests

After the successful Pre-operational self-tests, the module will perform Conditional algorithm self-tests (CASTs). The following Known Answer Tests (KATs) are performed, automatically and without any intervention. Any failure of a KAT will result in the module entering the error state.

- AES-ECB Encryption KAT (128 bits), PAA and non-PAA tested separately
- AES-ECB Decryption KAT (128 bits), PAA and non-PAA tested separately
- AES-CCB Encryption KAT (128 bits), PAA and non-PAA tested separately
- AES-CBC Decryption KAT (128 bits), PAA and non-PAA tested separately
- AES-XTS Encryption KAT (128 bits), PAA and non-PAA tested separately
- AES-XTS Decryption KAT (128 bits), PAA and non-PAA tested separately
- AES-CMAC Encryption KAT (128 bits), PAA and non-PAA tested separately
- AES-CMAC Decryption KAT (128 bits), PAA and non-PAA tested separately
- DRBG Instantiate KAT, PAA and non-PAA tested separately
- DRBG Generate KAT, PAA and non-PAA tested separately
- DRBG Reseed KAT, PAA and non-PAA tested separately
- Note: DRBG Health Tests as specified in SP800-90Arev1 Section 11.3 are performed)
- HMAC-SHA-1 KAT, PAA and non-PAA tested separately
- HMAC-SHA2-256 KAT, PAA and non-PAA tested separately
- HMAC-SHA2-512 KAT, PAA and non-PAA tested separately
- SHA-1 KAT, PAA and non-PAA tested separately
- SHA2-256 KAT, PAA and non-PAA tested separately
- SHA2-512 KAT, PAA and non-PAA tested separately

In addition, the module's entropy source also conducts the following entropy health tests:

Entropy source health tests

1. ENT (NP) SP800-90B start-up health tests:
   - Repetition Count Test (RCT)
   - Adaptive Proportion Test (APT)

Note: Please refer to SP800-90B, sections 4.4.1 and 4.4.2 for more information about the RCT and APT.

2. ENT (NP) SP800-90B continuous health tests:
   - Repetition Count Test (RCT)
   - Adaptive Proportion Test (APT)

Periodic/on-demand self-test

The module performs on-demand self-tests initiated by the operator, by power cycling or rebooting the tested platform. The full suite of self-tests is then executed. The same procedure may be employed by the operator to perform periodic self-tests.

It is recommended that the User perform periodic testing of the module's on-demand self-tests every 60 days to ensure all components are functioning correctly.

Error handling

If any one of the above-mentioned self-tests fails, including pre-operational self-test, CASTs, entropy source health tests, the module enters the Error state (there is only one error state). In the case of the module entering the Error State, the module sends a kernel panic signal to the kernel causing a reboot, preventing access to any non cryptographic services or data output from the module. The only method to recover from the error state is to reboot the module (triggered automatically by the kernel panic), and then pass the self-tests, including the pre-operational software integrity test and the conditional CASTs again. The module will only enter into the operational state after successfully passing the pre-operational software integrity test and the conditional CASTs.

11. Life-cycle assurance

Secure initialization and startup

The module is initialized during the loading of the module before any cryptographic functionality is available. The operating system is responsible for the initialization and loading processes of the module. The module is designed with constructor (default entry point of the module) which ensures that the Cryptographic Algorithm Self-Tests (CASTs) and Pre-operational Self-Test are initiated automatically when the module is loaded.

Secure operation

The module is provided directly to solution developers and is not intended for direct download by the general public. The module is installed on an operating system (Linux kernel) specified in Section 2.

Additional Rules of Operation:

1. The writable memory areas of the module (data and stack segments) are accessible only by the kernel itself so that the operating system is in "single user" mode. The module does not support concurrent operators.
2. The operating system is responsible for multitasking operations so that other processes cannot access the address space of the process containing the module.
3. Only the approved services defined in Section 4 above shall be used in approved mode of operation.
4. The length of a single data unit encrypted or decrypted with the AES-XTS shall not exceed 2^{20} AES blocks; that is, 16 MB of data per AES-XTS instance. An XTS instance is defined in section 4 of NIST SP 800-38E.
5. The AES-XTS mode shall only be used for the cryptographic protection of data on storage devices. The AES-XTS shall not be used for other purposes, such as the encryption of data in transit. The module implements the check to ensure that the two AES keys used in the XTS-AES algorithm are not identical

The module is not distributed as a standalone library as an Android developer must integrate the module into the Android system image.

The end user of the operating system is also responsible for zeroizing SSPs via powering down the tested platform that the module executes on to zeroize all SSPs used by the module.

Configuration management

The source code for the module is maintained in a git repository. While in process work on the code is maintained internally, code is eventually released to https://ci.android.com where it can be accessed by anyone. The source code manifest includes a build configuration providing a list of the specific tools needed to reproduce the module. The version number is generated by git based on the commit automatically.

Documentation related to the module is maintained in Google Docs. All documents (whether spreadsheets, documents, presentations or anything else) are automatically version tracked along with the owner. Like git, Docs uses access control lists to control access to the design documentation for the module.

Delivery and operation

The module is released as source code for other users to utilize as needed, but the module is only tested on the specific devices listed in section 2. Google follows the same build steps to produce the binary version of the module as laid out in the configuration management solution when creating the software that will be installed on the device. The build process is strictly controlled within Google, including access controls on the generation and certification of the builds to be used on a device for production.

The module is a built-in component of the overall Linux kernel image that is installed on the devices under test. The binary image produced from the build process is then installed onto the mobile devices at the factory, or updated via the device update process (OTA, or Over-the-Air updates). OTA updates follow the same build requirements as a factory installed image (which includes the module), but can be delivered after the device is purchased and in customer hands. All builds are signed and checked by the device (using checks burned into one-time fuses in the processor) to ensure that unauthorized binaries cannot be installed.

12. Mitigation of other attacks

The module does not claim mitigation of other attacks.