Android KeyMaster|GateKeeper

Broadcom Proprietary and Confidential

Release Note: This document can be shared with customers.

CONTENTS

akgi - Android KeyMaster|GateKeeper Integration

This document presents the information necessary on steps to be taken to setup a device properly for keymaster|gatekeeper support.

Prior to android P|9.0, the keymaster and the gatekeeper Android HAL's implementation have been using the 'software' hal variants provided in AOSP. Starting with Android P|9.0, the Broadcom integration offers a fully featured hardware backed keymaster and gatekeeper hal implementations.

In order for the hardware backed hal to work properly however, the device needs to be setup with proper configuration, such configuration may not have been necessary in the past (pre-P codebase) and therefore may not be available to the device you are working with.

The document will help you make sure the device you are using is properly setup for those new features.

Note: In the document we use the term "keymaster" to actually describe both the keymaster and gatekeeper hal's unless otherwise specified. The reason being that they both share the same

requirements in terms of integration with the secure engine, they both rely on the same trustlet adpater and they are synchronized on boot up|service launch.

Requirements for KeyMaster|GateKeeper

There are two requirements on the device configuration to consider.

In order to properly run the hals, both requirements must be met, failing either or both requirements may lead to the device being unable to properly function:

1. **[hard]** The device must have a keybox configuration fragment which contains the https://developer.android.com/training/articles/security-key-attestation (aka a keymaster drm).
    - a valid certificate should be used for production devices.
    - a test certificate is acceptable for development devices, typically the google test certificate.
2. **[soft]** The device must have a valid rpmb key programmed.
    - if the device has never run android verified boot, chances are the rpmb key was never programmed.
    - devices with already programmed rpmb keys that may have been programmed outside of an android verified boot path may therefore be unable to run the full feature set for those hals since the rpmb key can only be setup once per device lifetime (explained in details in the Android Verified Boot Document).

Failing to fulfill the configuration requirements

One important point is probably to understand what would happen to the device if either (or both) of the requirements for the device configuration are not met.

1. A failure to provide attestation certificate as part of a valid keymaster drm is a **fatal event** for the keymaster.
    - keymaster would look for such on startup (or any first operation) and failure to find any would result in it being unable to provide the services it is expected to provide, therefore it will bail early on error.
    - **it is important to note that android will fail to boot up if the keymaster module is present on the device but cannot be properly started.** there is no fallback, android will continue attempting to load up the same keymaster module which will continue to fail.
    - for development purposes, mitigation step are being taken (see end of this document).
2. Failing to configure RPMB.
    - a failure to configure rpmb is not fatal for the hal's; however some functionality would be unable to be run properly, in particular rollback resistant keys support would not be provided.

- the keymaster and gatekeeper hal would still provide most services and a device can be used for android in most cases.

Setting up the device

Programming RPMB key

The programming of the RPMB key is a per device (one time only) configuration.

This step may be considered optional, but is necessary for a proper full featured keymaster|gatekeeper integration.

To configure the rpmb key on a device, you must run a special (secure) bootloader version of the device bootloader. the device does not need to run in secure mode, but it does need to run the bootloader version which contains the necessary security steps allowing to setup the RPMB.

Configuring the attestation certificate

For Broadcom reference devices (zd|zb), this step can be skipped provided the mitigation plan outlined at the end of the document is in place (which it is on the P branch).

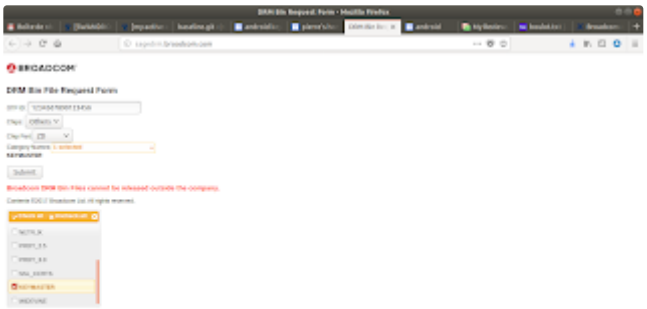
The attestation certificate is a per chip|device configuration. For internal Broadcom developers, requesting a drm.bin can be issued via this link.

This step is mandatory for keymaster|gatekeeper integration to work properly.

To configure the attestation certificate, you must request it from the sage drm form for a given chip otp id. the request is therefore per chip.

There are two ways you can go about this, either ways will require that a new|updated hwcfg image be flashed to the device.

The image below illustrates requesting the KEYMASTER portion of the drm binary from the sage drm request site.



You may proceed in either modes:

- request only the KEYMASTER portion of the drm binary associated with your chip otp identifier. save this drm to a file called "**km.bin**" which will become a fragment drm for keymaster of a fully construct drm.bin and can be placed alongside the existing drm.bin on an already configured device keybox (i.e. /dev/hwcfg).
- request the full "**drm.bin**" configuration module, selecting KEYMASTER as one option among all the options necessary for a given device and replace the entire drm.bin content on an existing device with the newly generated one.

The last step for this setting is to replace the existing "hwcfg" content with the new one containing either the standalone km.bin alongside a prior drm.bin or the new drm.bin.

For devices that are bp3 aware and which are therefore using a vfat hwcfg partitions, it should be possible to write the module directly to the partition after becoming root.

For all devices, including non bp3 aware devices, flashing a new version of the hwcfg.img with the modified content would suffice.

Mitigation for lack of Keymaster keybox

Note: this mitigation strategy does not work for zeus 5 based platforms (e.g. 7278) which are considered new platforms and which are required to be properly configured completely; zeus 4 devices should all be covered under this, which includes all the bcm reference platforms based on 7452S|72604|7268|7271.

In general, a device should always aim at being properly configured in order to run android integration, However in some circumstances it may not be possible or practical to do so, for those use cases, a mitigation plan is in place which allows the device to work.

If the keymaster is not able to startup properly due to the lack of its specific drm fragment availability on the device, the system will attempt to prepare and configure a default keymaster drm fragment using a canned certificate chain present on the vendor.img (the same certificate chain provided by google for testing of the keymaster feature set).

This canned certificate is a sage "type 1" generic certificate, which means it is not wrapped with the device unique otp identifier and can work on any devices of the same type (this it is important to make sure this is a dummy certificate with no consequences in case of a leakage); the keymaster hal processing will produce a device bound "type 3" certificate from this which would allow the keymaster TA to boot properly and therefore function as intended.

Note that this mechanism is only to provide a default keymaster keybox, the system still is required to provide the base keybox for android (the drm.bin). It is expected that most devices are already properly configured with a keybox drm.bin.

This mechanism should *not* be used for real production bound devices since it would expose a production certificate in a "clear" (type 1) form which could potentially be used on third party devices to mis-represent the keymaster device identity.

If however a device is not configured with a valid drm.bin initially, the mitigation plan would fail anyways and the device would fail to boot android. In such case a brute force approach is still possible by removing the hal's modules from the device:

"/vendor/lib/hw/keystore.<device>.so"
"/vendor/lib/hw/gatekeeper.<device>.so"

This would cause android to fall back to a similar state as the pre-P integration point (i.e use the software only variants of the hals).

Recommendations for integration

The following recommendations are being done for proper keymaster|gatekeeper integration.

1. **New devices in P**: for all new devices in android P, that is devices for which the first shipping api level is P, the device must be properly configured with a full featured drm.bin which includes the keymaster fragment (i.e. attestation certificate for the device as delivered by google). There is no exception to this rule, there is no other way to ensure the device will provide the proper services.
2. **Devices upgrading to P**: for devices upgrading to android P from a prior android version, typically would be android O or N-MR1 as example via an OTA mechanism. The following needs to be considered:
    1. **the device drm can be securely updated**: if the device drm can be securely updated in the field, whether this is because the device can do drm type conversion securely or the device can accept an update to its hwcfg configuration seamlessly - both of which would be specific to how a customer has integrated the reference provided by broadcom - **then**: such device can and should enable keymaster support for compliance with android P.
    2. **the device drm cannot be securely updated**: if the device drm cannot be securely updated, which means there is a potential for leakage of the attestation certificate in the case where it would be attempted to be delivered as a type 1 only to the device (refer to the mitigation plan for broadcom reference device above) - **then**: such device should *not* enable the keymaster support and should continue to use the android software variant of such, as per pre android P integration.