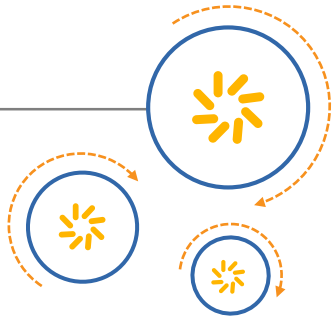
Qualcomm Technologies, Inc.



Qualcomm Android Security Features

80-NU861-1 C

December 8, 2015

Revision history

Revision	Date	Description
A	May 2015	Initial release.

Contents

1 Verified boot

Verified Boot is Google's defined mechanism for the Android boot loader to verify the integrity of Android boot and recovery images before bootup. To enable verified boot, enable device-mapper-verity (dm-verity) in the build system. See Chapter 11 for details on dm-verity.

1.1 Generation of signed images

Generate a signed boot.img and recovery.img.

The Android build system takes care of signing boot and recovery image. The key set by PRODUCT_VERITY_SIGNING_KEY is used to sign these images.

1.2 Generation of public key

To generate a public key:

1. Generate OEM's RSA_PUBLIC_KEY_PEM with openssl:

   ```
   openssl pkcs8 -inform DER -nocrypt -in <PRODUCT_VERITY_SIGNING_KEY>
   -out <RSA_PUBLIC_KEY_PEM>
   ```

2. Generate OEM's RSA_PUBLIC_KEY_DER with openssl:

   ```
   openssl rsa –in <RSA_PUBLIC_KEY_PEM> -pubout –outform DER
   –out <RSA_PUBLIC_KEY_DER>
   ```

3. Generate an OEM keystore.img by running the following command at the top of the Android build tree:

   ```
   keystore_signer <PRODUCT_VERITY_SIGNING_KEY> <verity.x509.pem>
   <KEYSTORE_IMG> <RSA_PUBLIC_KEY_DER>
   ```

   <RSA_PUBLIC_KEY_DER> is the RSA public key in DER format corresponding to the key used to sign boot.img and recovery.img. The Android build system uses a private key set in PRODUCT_VERITY_SIGNING_KEY to sign boot.img and recovery.img. So, the OEM's RSA_PUBLIC_KEY_DER must also be generated with PRODUCT_VERITY_SIGNING_KEY.

4. Generate oem_keystore.h for the Android boot loader (LK):

   LK needs a header file (oem_keystore.h) that was generated with the contents of the OEM keystore.img generated in step 2.

   a. Run the following code to generate this header file:

```
function generate_oem_keystore_h()
{
    echo \#ifndef __OEM_KEYSTORE_H
    echo \#define __OEM_KEYSTORE_H
    xxd -i $1 | sed -e 's/unsigned char .* = {/const unsigned char
OEM_KEYSTORE[] = {/g' -e 's/unsigned int .* =.*;//g'
    echo \#endif
}
generate_oem_keystore_h KEYSTORE_IMG > oem_keystore.h
```

   b. Copy the oem_keystore.h file to location bootable/bootloader/lk/platform/msm_shared/include.

5. Compile the Android boot loader.

NOTE: LK compiles the verified boot code if dm-verity is enabled in the Android build system.

The Device is marked as UNLOCKED by the Android boot loader for userdebug variant builds. After flashing all images to test the verified boot feature on a userdebug build, you must lock the device with the following command:

```
fastboot oem lock
```

1.3 Partitions used for Android verified boot

The partitions that are added for the Android verified boot feature in LK are as follows:

- DEVINFO

  If secure boot is not enabled, this partition is used to store device information such as locked state, verified state, tampered state, bootloader version, etc. The device_info structure is as follows:

```
struct device_info
{
        unsigned char magic[DEVICE_MAGIC_SIZE];
        bool is_unlocked;
        bool is_tampered;
        bool is_verified;
        bool charger_screen_enabled;
        char display_panel[MAX_PANEL_ID_LEN];
        char bootloader_version[MAX_VERSION_LEN];
        char radio_version[MAX_VERSION_LEN];
};
```

  If this partition is not present, device information is stored in a boot partition at the end of the partition.

- RPMB

  This is a secure partition for storing device information and can be accessed only through TZ.

  If USE_RPMB_FOR_DEVINFO is enabled (defined in LK makefile), the device information is stored in this partition only if secure boot is enabled. If secure boot is not enabled, the DEVINFO partition is used for storing this information.

  RPMB key is only generated in TZ. This requires LK to load LKSecapp, which is a secure TZ application, and RPMB reads and writes are requested via this secure application.

  The file is located at bootable/bootloader/lk/app/aboot/aboot.c. The code is as follows:

```
#if USE_RPMB_FOR_DEVINFO
    if (is_secure_boot_enable())
      write_device_info_rpmb((void*) info, mmc_get_device_blocksize());
    else
      write_device_info_mmc(info);
#else
      write_device_info_mmc(info);
#endif
```

- LKSECAPP

  The secure TZ application LKSecapp, is stored in this partition.

- CMNLIB

  This partition stores the commonlib TZ application, which provides APIs for the QSEECOM libraries and common services such as heap and secure file system. RPMB requests made by LK are sent to TZ through this framework.