



Unisoc Confidential For kxdwww

# Android Dm-Verity 适配指导手册

文档版本            V2.2  
发布日期            2021-12-30

**版权所有 © 紫光展锐（上海）科技有限公司。保留一切权利。**

本文件所含数据和信息都属于紫光展锐（上海）科技有限公司（以下简称紫光展锐）所有的机密信息，紫光展锐保留所有相关权利。本文件仅为信息参考之目的提供，不包含任何明示或默示的知识产权许可，也不表示有任何明示或默示的保证，包括但不限于满足任何特殊目的、不侵权或性能。当您接受这份文件时，即表示您同意本文件中内容和信息属于紫光展锐机密信息，且同意在未获得紫光展锐书面同意前，不使用或复制本文件的整体或部分，也不向任何其他方披露本文件内容。紫光展锐有权在未经事先通知的情况下，在任何时候对本文件做任何修改。紫光展锐对本文件所含数据和信息不做任何保证，在任何情况下，紫光展锐均不负任何与本文件相关的直接或间接的、任何伤害或损失。

请参照交付物中说明文档对紫光展锐交付物进行使用，任何人对紫光展锐交付物的修改、定制化或违反说明文档的指引对紫光展锐交付物进行使用造成的任何损失由其自行承担。紫光展锐交付物中的性能指标、测试结果和参数等，均为在紫光展锐内部研发和测试系统中获得的，仅供参考，若任何人需要对交付物进行商用或量产，需要结合自身的软硬件测试环境进行全面的测试和调试。

Unisoc Confidential For kxdwww

# 紫光展锐（上海）科技有限公司



# 前言

## 概述

本文档主要介绍了 AVB 过程中大镜像分区（比如 system 分区）的校验原理以及实现方式。

## 读者对象




适用于 AVB 和 Dm-Verity 相关研发及测试人员。

## 缩略语

缩略语	英文全名	中文解释
AVB	Android Verified Boot	Android 启动验证

## 符号约定

在本文中可能出现下列标志，它所代表的含义如下。

符号	说明
 <b>说明</b>	用于突出重要/关键信息、补充信息和小窍门等。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害。
 <b>注意</b>	用于强调段落中或句子需要读者重视的信息。
 <b>警告</b>	用于可能无法恢复的失误操作。 “警告”不是危险警示信息，不涉及人身及环境伤害。

## 变更信息

文档版本	发布日期	修改说明
V2.2	2021/12/30	<ul style="list-style-type: none"><li>新增 <b>2.7 Android 12.0</b>。</li><li>调整结构，优化表达。</li></ul>
V2.1	2020/08/18	<ul style="list-style-type: none"><li>新增 <b>2.6 Android 11.0</b>。</li></ul>

文档版本	发布日期	修改说明
		<ul style="list-style-type: none"><li>文档名称由《Android Dm-Verity 适配指导文档》更改为《Android Dm-Verity 适配指导手册》。</li></ul>
V2.0	2019/11/08	增加 Android 10.0 相关内容 <b>2.5 Android 10.0</b> 。
V1.0	2015/02/20	第一次正式发布。

## 关键字

Dm-Verity、AVB

Unisoc Confidential For kxdwww

# 目 录

1 Dm-Verity 概述.....	1
1.1 AVB .....	1
1.2 Dm-Verity 实现 .....	1
2 Dm-Verity 在各 Android 平台适配 .....	3
2.1 Android 6.0.....	3
2.1.1 功能适配.....	3
2.1.2 定制 key.....	6
2.1.3 功能验证 .....	7
2.1.4 功能关闭 .....	7
2.2 Android 7.0.....	7
2.2.1 功能适配.....	8
2.2.2 定制 key.....	8
2.2.3 功能验证 .....	8
2.2.4 功能关闭 .....	9
2.3 Android 8.1.....	9
2.3.1 功能适配.....	9
2.3.2 定制 key.....	10
2.3.3 功能验证 .....	11
2.3.4 功能关闭 .....	11
2.4 Android 9.0.....	11
2.4.1 功能适配.....	12
2.4.2 定制 key.....	14
2.4.3 功能验证 .....	14
2.4.4 功能关闭 .....	15
2.5 Android 10.0.....	15
2.5.1 功能适配.....	16
2.5.2 定制 key.....	17
2.5.3 功能验证 .....	18
2.5.4 功能关闭 .....	19
2.6 Android 11.0.....	19
2.6.1 功能适配.....	20
2.6.2 定制 key.....	21
2.6.3 功能验证 .....	21
2.6.4 功能关闭 .....	23
2.7 Android 12.0.....	23
2.7.1 功能适配.....	23

---

2.7.2 定制 key .....	25
2.7.3 功能验证 .....	25
2.7.4 功能关闭 .....	27
3 常见问题.....	28
3.1 性能影响 .....	28
3.2 OTA 影响 .....	28
3.3 Android 9.0 userdebug 版本 remount 失败.....	28
3.4 功能异常排查指引 .....	29
4 参考文档.....	30

Unisoc Confidential For kxdwww

## 图目录

图 1-1 哈希树结构图 .....	2
图 2-1 Android 6.0 Kernel 功能配置 .....	3
图 2-2 Android 9.0 Kernel 功能配置 .....	12
图 2-3 Android 10.0 Kernel 功能配置 .....	16
图 2-4 Android 11.0 Kernel 功能配置 .....	20
图 2-5 Android 12.0 Kernel 功能配置 .....	24

Unisoc Confidential For kxdwww

## 表目录

表 2-1 Android 11.0 动态分区与 Vbmeta 分区对应关系.....	19
表 2-2 Android 12.0 动态分区与 Vbmeta 分区对应关系.....	23

Unisoc Confidential For kxdwww



# 1 Dm-Verity 概述

Dm-Verity 是一项内核的功能，可以提供透明的对块设备的完整性校验，可以用于防止恶意程序对只读系统分区（如 system、vendor 等）的篡改。在介绍 Dm-Verity 之前，要先了解 Verified Boot（启动验证），包含如下两个版本

- Verified Boot 1.0: Google 从 Android 4.4 开始引入内核 Dm-Verity 功能，展锐平台从 Android 6.0 开始支持 Dm-Verity。
- Verified Boot 2.0: Google 从 Android 8.x 开始支持 AVB。

## 1.1 AVB

对于要启动的 Android 版本中包含的所有可执行代码和数据，启动前均要求对其进行验证，包括内核（从 boot 分区加载）、设备树（从 dtbo 分区加载）、system 分区和 vendor 分区等。

AVB 根据分区大小将分区分成两类：

- 仅读取一次的小分区（如 boot 和 dtbo）：首先将整个内容加载到内存中，再计算相应哈希值来进行验证的，然后比较这个计算出的哈希值与预期哈希值，如果值不一致，则该小分区将无法加载，Android 系统无法启动。
- 内存装不下的较大分区（如 system 和 vendor），使用内核 Dm-Verity 的驱动程序（device-mapper-verity）对块设备进行透明的完整性校验。

Dm-Verity 包括两次校验：

- 启动时校验，验证哈希树（Hash Tree）是否被篡改。
- 运行时校验，校验流程在将数据加载到内存的过程中持续进行。对于运行时校验，Android 会在运行时计算哈希树的根哈希值，并对照预期根哈希值进行检查。如果在某个时间点计算出的根哈希值与预期根哈希值不一致，系统便不会使用相应数据，并提示 Dm-Verity 校验错误

预期哈希值通常存储在每个待验证的分区的末尾或专用分区。最重要的是，这些哈希值是由信任根以直接或间接的方式签名的。

## 1.2 Dm-Verity 实现

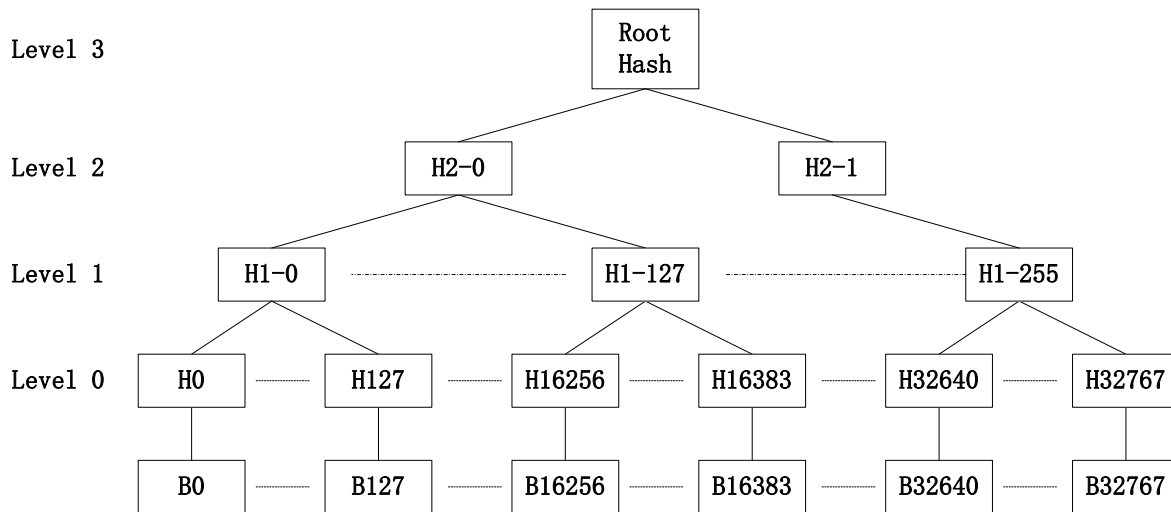
Dm-Verity 虚拟了一个块设备，对块数据进行读取时会先进行哈希计算，并与预先计算的哈希树进行校验，如果匹配则读取成功，否则会生成读取 I/O 错误，以此来达到数据完整性校验的目的。

预先计算的哈希树是对要校验的目标设备的所有块做哈希计算并最终得到一个根哈希的哈希值集合。对目标设备的所有块做哈希计算，每个块（一般是 4 KB）对应一个 SHA 散列（即 32 字节的哈希值），这些 SHA 散列存储在哈希树的叶子节点。对多个叶子节点（如 128 个哈希值大小为 4 KB）做哈希计算 boot 值，则多个叶子节点对应一个 SHA 散列（即 32 字节的哈希值），这样就形成了哈希树的中间节点。通过

多次反复的哈希计算，直到得出唯一哈希值为止，该数值称为根哈希（Root Hash）。基于散列的特性，当一个块有任意的变化，都会导致根哈希的数值变化。

以一个包含 32768 个块的设备为例，哈希树的结构图如图 1-1 所示。

图1-1 哈希树结构图



Unisoc Confidential For kxdwww

# 2 Dm-Verity 在各 Android 平台适配

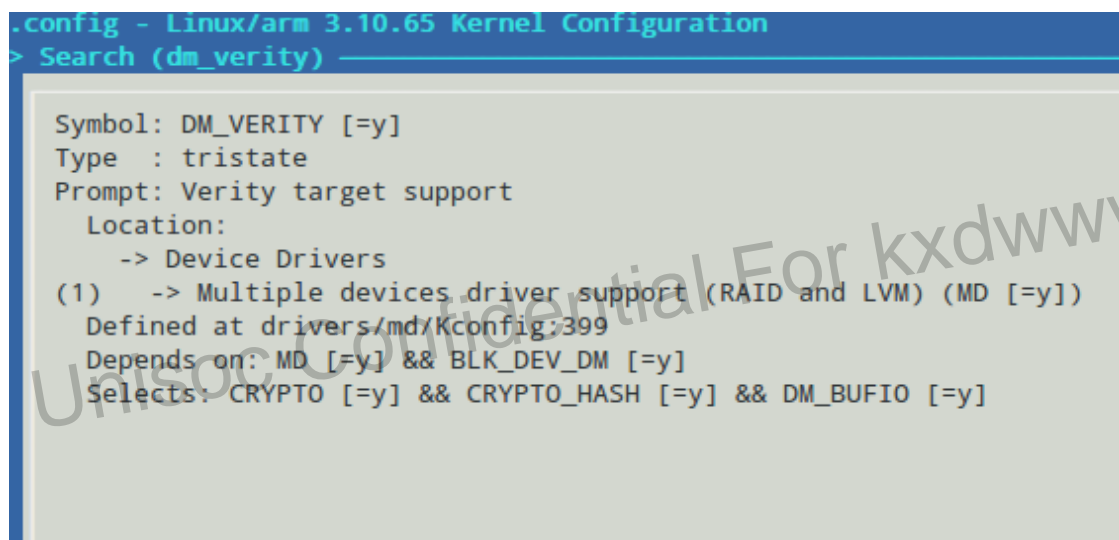
## 2.1 Android 6.0

### 2.1.1 功能适配

#### 1. 配置 kernel config

通过 `kuconfig` 命令配置 kernel config `DM_VERITY`，配置路径如图 2-1 所示。

图2-1 Android 6.0 Kernel 功能配置



修改生效的文件是 `/kernel/arch/arm/arm64/configs/` 下面对应工程的配置文件：

# CONFIG_DM_FLAKY is not set	1254 # CONFIG_DM_FLAKY is not set
# CONFIG_DM_VERITY is not set	1255 CONFIG_DM_BUFIO=y
	1256 CONFIG_DM_VERITY=y
# CONFIG_TARGET_CORE is not set	1257 # CONFIG_TARGET_CORE is not set

#### 2. 修改 Board 相关配置

- `/dev/sprd/scx35/common/device.mk`，增加 Dm-Verity 的编译配置，同时用 `TARGET_DM_VERITY` 宏控制：

- fstab 中对 system 分区增加校验标志:

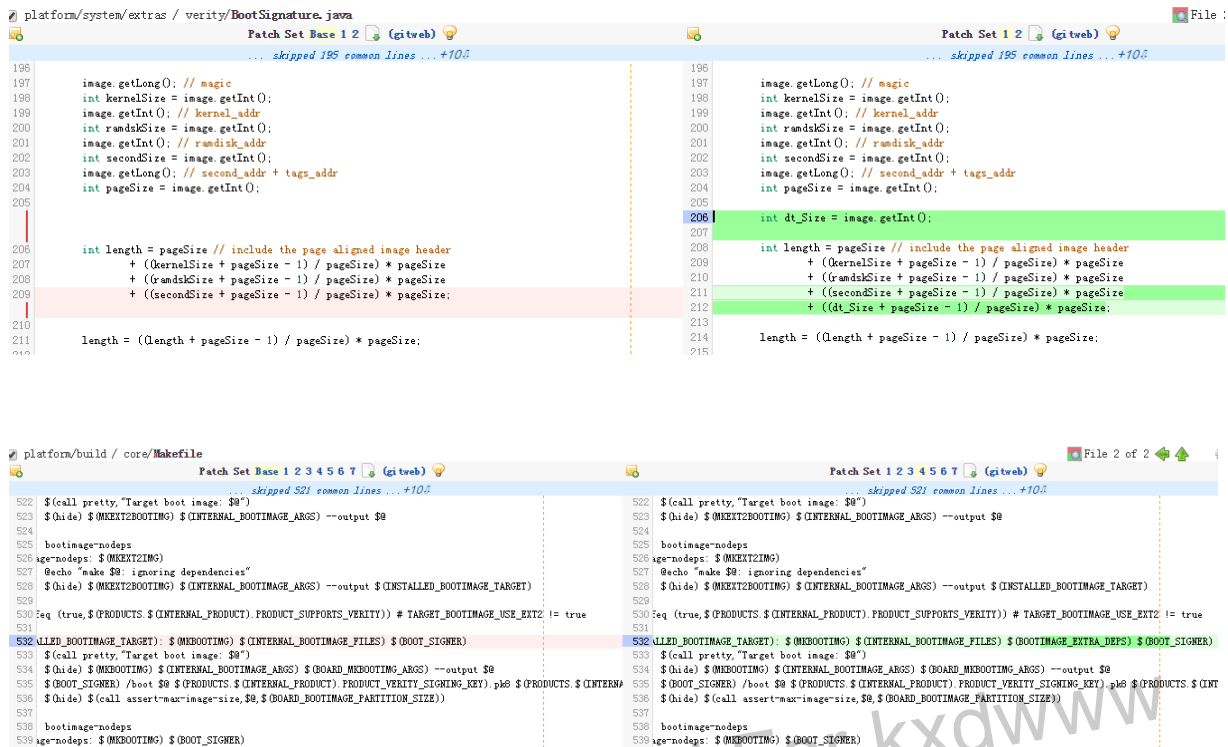
但是为了兼容性, 不建议做直接的修改。改换另一种修改方式如下:

- 1) 增加一个 fstab 文件: **fstab.sc8830.verify**, 该文件拷贝自 **fstab.sc8830**, 并参考上图所展示的差异进行修改。
- 2) 在对应的 Board 中通过 **TARGET\_DM\_VERITY** 宏来控制决定拷贝的 fstab 文件, 参考修改如下:

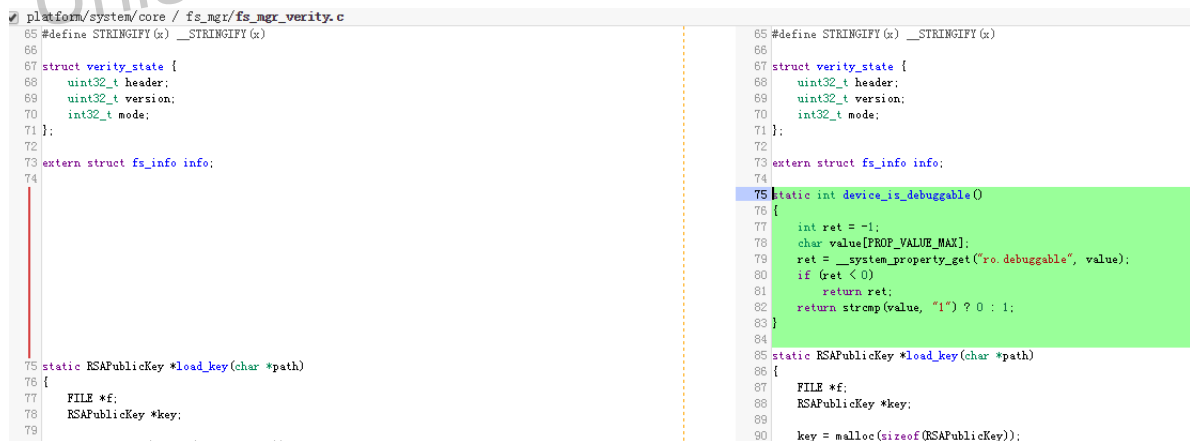
- 3) 在需要打开此功能的 Board 将 **TARGET\_DM\_VERITY** 置为 true, 参考修改如下:

### 3. 修改签名程序及编译脚本

展锐的 kernel 中增加了 device tree 功能，因此需要修改对应的签名程序以及编译脚本，增加 dt 的相关信息。



#### 4. Userdebug 版本关闭该功能，以方便其他功能调试





```

platform/system/core / fs_mgr/fs_mgr_verity.c
205 }
206
207 static int read_verity_metadata(uint64_t device_size, char *block_device, char **signature,
208 char **table)
209 {
210     unsigned magic_number;
211     unsigned table_length;
212     int protocol_version;
213     int device;
214     int retval = FS_MGR_SETUP_VERITY_FAIL;
215
216     *signature = NULL;
217
218     if (table) {
219         *table = NULL;
220     }
221
222     device = TEMP_FAILURE_RETRY(open(block_device, O_RDONLY | O_CLOEXEC));
223     if (device == -1) {
224         ERROR("Could not open block device %s (%s).\n", block_device, strerror(errno));
225         goto out;
226     }
227
215 }
216
217 static int read_verity_metadata(uint64_t device_size, char *block_device, char **signature,
218 char **table)
219 {
220     unsigned magic_number;
221     unsigned table_length;
222     int protocol_version;
223     int device;
224     int retval = FS_MGR_SETUP_VERITY_FAIL;
225
226     *signature = NULL;
227
228     if (table) {
229         *table = NULL;
230     }
231
232     device = TEMP_FAILURE_RETRY(open(block_device, O_RDONLY | O_CLOEXEC));
233     if (device == -1) {
234         ERROR("Could not open block device %s (%s).\n", block_device, strerror(errno));
235         goto out;
236     }
237
238     if(device_is_debuggable()) {
239         retval = FS_MGR_SETUP_VERITY_DISABLED;
240         goto out;
241     }
242

```

完成上述配置后，需要做一次完整编译，并重新烧录 boot.img、system.img、userdata.img，即可在 userdebug 版本关闭 Dm-Verity 功能。

## 2.1.2 定制 key

### 1. Key 文件作用说明

Dm-Verity 包含三个 key 文件，路径位于：build/target/product/security/，具体作用为：

- verity.pk8：这是一个私钥，用于给 boot.img 和 system.img 签名。
- verity.x509.pem：这是一个证书，此证书内包含有公钥信息。
- verity\_key：这是一个公钥，用于 system.img 的 Dm-Verity 完整性校验。

### 2. 替换原生 key

替换方法如下：

#### a 生成 verity 相关的三个 key 文件：verity.pk8、verity.x509.pem、verity\_key：

- 1) 在 Linux 系统中，确保所安装 openssl 版本不低于 1.0（Openssl 版本号可在 Ubuntu 终端输入“openssl version”查看，比如：OpenSSL 1.0.1f 6 Jan 2014，openssl 版本 1.0 是符合要求的）。
- 2) 终端切换到 IDH 代码的根目录下，输入如下命令（直接回车，不需要输入密码），会在当前根目录生成 verity.pk8 和 verity.x509.pem：

```
development/tools/make_key verity 'C=US/ST=California/L=Mountain View/O=Android/OU=Android/CN=Android/emailAddress=android@android.com'
```

- 3) 在终端中执行 source build/envsetup.sh、lunch 选择对应的工程、kheader 后，再输入 make generate\_verity\_key 或者 mmm system/extras/verity/，就会生成 generate\_verity\_key。
- 4) 在终端继续输入如下命令，将会在 idh.code 根目录生成 verity\_key.pub：

```
out/host/linux-x86/bin/generate_verity_key -convert verity.x509.pem verity_key
```

- 5) 将 verity\_key.pub 重命名为 verity\_key。

#### b 将上述生成的三个 key 替换到 build/target/product/security/目录下。

#### c 重新进行完整版本编译。

### 3. 验证 key 是否替换成功的方法

#### a verity\_key 对比验证



verity\_key 最后会被打包进 boot.img 中，以 SC9832E 为例，编译时生成的目录位于：  
out/target/product/sp9832e\_1h10/root。对比 build/target/product/security/目录下的 verity\_key 和前述  
用命令生成的 verity\_key，如果两者一致则说明替换成功。

#### b 交叉验证

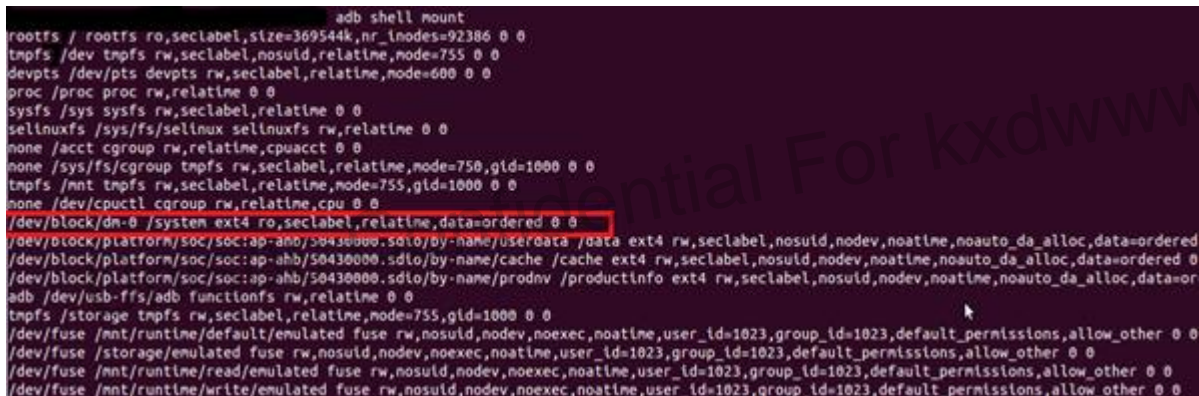
基于同一软件版本，分别使用 A 和 B 两组不同的 key 编译得到 A 和 B 两个完整版本（确保仅  
verity 的 key 不同）。验证步骤如下：

- 1) 下载版本 A 至手机中，开机并按照“验证 Dm-Verity 功能”说明，确认 Dm-Verity 功能开启  
正常。预期结果：手机开机正常，system 分区按照 dm-0 的方式挂载。
- 2) 下载替换版本 B 的 system.img，开机验证。预期结果：手机无法开机。
- 3) 继续下载替换版本 B 的 boot.img，开机验证。预期结果：手机开机正常，system 分区按照  
dm-0 的方式挂载。

### 2.1.3 功能验证

验证此功能需要在 user 版本上验证，因为 userdebug 版本上该功能是关闭的。

验证功能是否成功开启的办法：查看 system 分区的挂载方式，当挂载方式为 dm-x 时，表明功能开启成  
功，反之则是未开启。



```
adb shell mount
rootfs / rootfs ro,seclabel,size=369544k,nr_inodes=92386 0 0
tmpfs /dev tmpfs rw,seclabel,nosuid,relatime,nodev=755 0 0
devpts /dev/pts devpts rw,seclabel,relatime,nodev=600 0 0
proc /proc proc rw,relatime 0 0
sysfs /sys sysfs rw,seclabel,relatime 0 0
selinuxfs /sys/fs/selinux selinuxfs rw,relatime 0 0
none /acct cgroup rw,relatime,cpuacct 0 0
none /sys/fs/cgroup tmpfs rw,seclabel,relatime,nodev=750,gid=1000 0 0
tmpfs /mnt tmpfs rw,seclabel,relatime,nodev=755,gid=1000 0 0
none /dev/cpuctl cgroup rw,relatime,cpu 0 0
/dev/block/dm-0 /system ext4 ro,seclabel,relatime,data=ordered 0 0
/dev/block/platform/soc/soc:ap-ahb/50430000.sdio/by-name/userdata /data ext4 rw,seclabel,nosuid,nodev,noatime,noauto_da_alloc,data=ordered 0
/dev/block/platform/soc/soc:ap-ahb/50430000.sdio/by-name/cache /cache ext4 rw,seclabel,nosuid,nodev,noatime,noauto_da_alloc,data=ordered 0
adb /dev/usb-ffs/adb functionfs rw,relatime 0 0
tmpfs /storage tmpfs rw,seclabel,relatime,nodev=755,gid=1000 0 0
/dev/fuse /mnt/runtime/default/emulated fuse rw,nosuid,nodev,noexec,noatime,user_id=1023,group_id=1023,default_permissions,allow_other 0 0
/dev/fuse /storage/emulated fuse rw,nosuid,nodev,noexec,noatime,user_id=1023,group_id=1023,default_permissions,allow_other 0 0
/dev/fuse /mnt/runtime/read/emulated fuse rw,nosuid,nodev,noexec,noatime,user_id=1023,group_id=1023,default_permissions,allow_other 0 0
/dev/fuse /mnt/runtime/write/emulated fuse rw,nosuid,nodev,noexec,noatime,user_id=1023,group_id=1023,default_permissions,allow_other 0 0
```

### 2.1.4 功能关闭

通过将 TARGET\_DM\_VERITY 置为 false 来关闭。

## 2.2 Android 7.0

Android 7.0 中针对 Dm-Verity 增加了一项新的子项功能 DM\_VERITY\_FEC，即 Verity forward error  
correction。由于此功能的引入，导致 system.img 格式发生变化，从而校验方式也发生变化。要求  
system.img 的大小和 xml 中定义 system 分区的大小需要保持一致，否则就会由于找不到校验数据，导致  
system 分区无法挂载，出现无法开机的情况。

为了满足上述新功能需求，平台做了以下功能性修改：

1. 增加 System 分区自适应功能。通过 SYSTEM\_IMAGE\_SIZE\_ADAPT 宏来控制。当  
SYSTEM\_IMAGE\_SIZE\_ADAPT 为 false 的时候，功能关闭，否则功能开启。

2. 针对 System 分区特殊大小（比如 1600MB），进行分区大小再调整的修改。具体方案为：当 System 分区自适应功能打开时，在编译脚本中自动进行分区调整；当 system 分区自适应功能关闭，在编译阶段报错，并主动提示用户需要配置 BOARD\_SYSTEMIMAGE\_PARTITION\_SIZE 的大小。

**注意**

特殊大小包含两个含义：

- 当 system 分区为 1600MB 时，无法分配出合适的 system 文件系统（以 A 表示）和校验数据的大小（以 B 表示），因为 B 的大小是跟随 A 的大小而变化的。当 A 为 1651507200，B 计算出的大小为 26210304，总大小为 1677717504（刚好比 1600MB 少一个 block 大小，也就是 4096）；当 A 为 1651511296，B 计算出的大小为 26214400。总大小为 1677725696（刚好比 1600MB 多一个 block，也就是 4096）其中 B 包含三部分：B1（GetVerityTreeSize）、B2（GetVerityMetadataSize）、B3（GetVerityFECSize）。
- 文件系统的大小要求规则：假设传进来的镜像大小是 X，X 与 128MB 求余，余数为 Y，如果  $0 < Y < 4MB$ ，则文件系统会把 Y 丢掉，如果大于等于 4MB，则文件系统不会丢 block。

平台默认开启 system 分区大小自适应功能。如需关闭该功能，则对 system 分区的大小进行调整时，需要同时修改 xml 中的 system 分区大小并保证和 BOARD\_SYSTEMIMAGE\_PARTITION\_SIZE 一致。同时，配置分区大小的数值需要为 50MB 的倍数，并且排除 1600MB 这个特殊值。

## 2.2.1 功能适配

配置 kernel config:

```
kernel/common / arch/arm/configs/sp9832a_2h11_volte_defconfig
Patch Set Base 1 2 3 4 (gitweb)

1281 # CONFIG_BLK_DEV_MD is not set
1282 # CONFIG_BCACHE is not set
1283 CONFIG_BLK_DEV_DM_BUILTIN=y
1284 CONFIG_BLK_DEV_DM=y
1285 # CONFIG_DM_DEBUG is not set
1286 CONFIG_DM_CRYPT=y
1287 # CONFIG_DM_SNAPSHOT is not set
1288 # CONFIG_DM_THIN_PROVISIONING is not set
1289 # CONFIG_DM_CACHE is not set
1290 # CONFIG_DM_MIRROR is not set
1291 # CONFIG_DM_RAID is not set
1292 # CONFIG_DM_ZERO is not set
1293 # CONFIG_DM_MULTIPATH is not set
1294 # CONFIG_DM_DELAY is not set
1295 # CONFIG_DM_UEVENT is not set
1296 # CONFIG_DM_FLAKEY is not set
1297 # CONFIG_DM_VERITY is not set
1298 # CONFIG_TARGET_CORE is not set
1299 CONFIG_NETDEVICES=y
1300 CONFIG_NET_CORE=y
1301 # CONFIG_BONDING is not set

Patch Set 1 2 3 4 (gitweb)
1281 # CONFIG_BLK_DEV_MD is not set
1282 # CONFIG_BCACHE is not set
1283 CONFIG_BLK_DEV_DM_BUILTIN=y
1284 CONFIG_BLK_DEV_DM=y
1285 # CONFIG_DM_DEBUG is not set
1286 CONFIG_DM_CRYPT=y
1287 # CONFIG_DM_SNAPSHOT is not set
1288 # CONFIG_DM_THIN_PROVISIONING is not set
1289 # CONFIG_DM_CACHE is not set
1290 # CONFIG_DM_MIRROR is not set
1291 # CONFIG_DM_RAID is not set
1292 # CONFIG_DM_ZERO is not set
1293 # CONFIG_DM_MULTIPATH is not set
1294 # CONFIG_DM_DELAY is not set
1295 # CONFIG_DM_UEVENT is not set
1296 CONFIG_DM_FLAKEY=y
1297 CONFIG_DM_VERITY=y
1298 CONFIG_DM_VERITY_FEC=y
1299 # CONFIG_TARGET_CORE is not set
1300 CONFIG_NETDEVICES=y
1301 CONFIG_NET_CORE=y
1302 CONFIG_NET_CORE=y
1303 # CONFIG_BONDING is not set
```

其余修改请参考 [2.1.1 功能适配](#)。

## 2.2.2 定制 key

Android 7.0 定制 key 的方式与 Android 6.0 相同，请参考 [2.1.2 定制 key](#)。

## 2.2.3 功能验证

验证此功能需要在 user 版本上验证，因为 userdebug 版本上该功能是关闭的。

验证功能是否成功开启的办法：查看 system 分区的挂载方式，当挂载方式为 dm-x 时，表明功能开启成功，如下图所示。反之则是未开启。



```
adb shell mount
rootfs / rootfs ro,seclabel,size=369544k,nr_inodes=92386 0 0
tmpfs /dev tmpfs rw,seclabel,nosuid,relatime,nodev=755 0 0
devpts /dev/pts devpts rw,seclabel,relatime,nodev=600 0 0
proc /proc proc rw,relatime 0 0
sysfs /sys sysfs rw,seclabel,relatime 0 0
selinuxfs /sys/fs/selinux selinuxfs rw,relatime 0 0
none /acct cgroup rw,relatime,cpuacct 0 0
none /sys/fs/cgroup tmpfs rw,seclabel,relatime,nodev=750,gid=1000 0 0
tmpfs /mnt tmpfs rw,seclabel,relatime,nodev=755,gid=1000 0 0
none /dev/cpuctl cgroup rw,relatime,cpu 0 0
/dev/block/dm-0 /system ext4 ro,seclabel,relatime,data=ordered 0 0
/dev/block/platform/soc/soc:ap-ahb/50430000.sdlo/by-name/userdata /data ext4 rw,seclabel,nosuid,nodev,noatime,noauto_da_alloc,data=ordered 0 0
/dev/block/platform/soc/soc:ap-ahb/50430000.sdlo/by-name/cache /cache ext4 rw,seclabel,nosuid,nodev,noatime,noauto_da_alloc,data=ordered 0 0
adb /dev/usb-ffs/adb functionfs rw,relatime 0 0
tmpfs /storage tmpfs rw,seclabel,relatime,nodev=755,gid=1000 0 0
/dev/fuse /mnt/runtime/default/emulated fuse rw,nosuid,nodev,noexec,noatime,user_id=1023,group_id=1023,default_permissions,allow_other 0 0
/dev/fuse /storage/emulated fuse rw,nosuid,nodev,noexec,noatime,user_id=1023,group_id=1023,default_permissions,allow_other 0 0
/dev/fuse /mnt/runtime/read/emulated fuse rw,nosuid,nodev,noexec,noatime,user_id=1023,group_id=1023,default_permissions,allow_other 0 0
/dev/fuse /mnt/runtime/write/emulated fuse rw,nosuid,nodev,noexec,noatime,user_id=1023,group_id=1023,default_permissions,allow_other 0 0
```

## 2.2.4 功能关闭

通过将 TARGET\_DM\_VERITY 置为 false 来关闭。

## 2.3 Android 8.1

在 Android 8.1 上适配 Dm-Verity 功能与之前的 Android 版本有所不同。

- Verify Boot 在 Android 8.0 采用了 AVB2.0 的方案，Android 6.0 及 Android 7.0 版本都是采用的 AVB1.0 的方案。因 Dm-Verity 作为 Verify Boot 功能的一部分，也采用了 AVB2.0 的方案。
- Android 8.1 上 system 和 vendor 分区的挂载逻辑提前到 DoFirstStageMount() 函数中完成，故校验 flag 的配置需要在 dts 文件中定义。

### 2.3.1 功能适配

#### 1. 配置 kernel config

kernel/common / arch/arm/configs/sp9832a_2h11_volte_defconfig	Patch Set 1 2 3 4 (gitweb)
1281 # CONFIG_BLK_DEV_MD is not set	1281 # CONFIG_BLK_DEV_MD is not set
1282 # CONFIG_BCACHE is not set	1282 # CONFIG_BCACHE is not set
1283 CONFIG_BLK_DEV_DM_BUILTIN=y	1283 CONFIG_BLK_DEV_DM_BUILTIN=y
1284 CONFIG_BLK_DEV_DM=y	1284 CONFIG_BLK_DEV_DM=y
1285 # CONFIG_DM_DEBUG is not set	1285 # CONFIG_DM_DEBUG is not set
1286 CONFIG_DM_CRYPT=y	1286 CONFIG_DM_CRYPT=y
1287 # CONFIG_DM_SNAPSHOT is not set	1287 # CONFIG_DM_SNAPSHOT is not set
1288 # CONFIG_DM_THIN_PROVISIONING is not set	1288 # CONFIG_DM_THIN_PROVISIONING is not set
1289 # CONFIG_DM_CACHE is not set	1289 # CONFIG_DM_CACHE is not set
1290 # CONFIG_DM_MIRROR is not set	1290 # CONFIG_DM_MIRROR is not set
1291 # CONFIG_DM_RAID is not set	1291 # CONFIG_DM_RAID is not set
1292 # CONFIG_DM_ZERO is not set	1292 # CONFIG_DM_ZERO is not set
1293 # CONFIG_DM_MULTIPATH is not set	1293 # CONFIG_DM_MULTIPATH is not set
1294 # CONFIG_DM_DELAY is not set	1294 # CONFIG_DM_DELAY is not set
1295 # CONFIG_DM_UEVENT is not set	1295 # CONFIG_DM_UEVENT is not set
1296 # CONFIG_DM_FLAKY is not set	1296 # CONFIG_DM_FLAKY is not set
1297 # CONFIG_DM_VERITY is not set	1297 CONFIG_DM_VERITY=y
1298 # CONFIG_TARGET_CORE is not set	1298 CONFIG_DM_VERITY_FEC=y
1299 CONFIG_NETDEVICES=y	1300 # CONFIG_TARGET_CORE is not set
1300 CONFIG_NET_CORE=y	1301 CONFIG_NETDEVICES=y
1301 # CONFIG_BONDING is not set	1302 CONFIG_NET_CORE=y
	1303 # CONFIG_BONDING is not set

#### 2. 修改 dts 文件，开启 Dm-Verity flag

适配 system 和 vendor 分区 Dm-Verity 挂载方式，需要修改 dts 文件。在 dts 文件中添加 avb flag，修改方案有两种：

- system 和 vendor 的 fsmgr\_flags 从 “wait” 改成 “wait,avb”，修改文件 kernel/arch/arm/boot/dts/sp9832e-common.dtsi 如下如所示。

```

firmware {
    android {
        compatible = "android,firmware";

        vbmeta {
            compatible = "android,vbmeta";
            parts = "vbmeta,boot,recovery,system,vendor";
        };

        fstab {
            compatible = "android,fstab";
            fs_system: system {
                compatible = "android,system";
                dev = "/dev/block/platform/soc/soc:ap-ahb/20600000.sdio/by-name/system";
                type = "ext4";
                mnt_flags = "ro,barrier=1";
                fsmgr_flags = "wait";
            };
            fs_vendor: vendor {
                compatible = "android,vendor";
                dev = "/dev/block/platform/soc/soc:ap-ahb/20600000.sdio/by-name/vendor";
                type = "ext4";
                mnt_flags = "ro,barrier=1";
                fsmgr_flags = "wait";
            };
        };
    };
};

```

- b 新增一个文件包含对应的 dts 文件，然后再覆盖 fsmgr\_flags 属性。如新增一个 dts 文件：sp9832e-1h10-gofu-avb.dts，详细修改参考下图。

```

1 /*
2  * Spreadtrum SP9832E 1H10 GO FULL board DTS file
3  *
4  * Copyright (C) 2016-2017, Spreadtrum Communications Inc.
5  *
6  * This file is licensed under a dual GPLv2 or X11 license.
7  */
8 #include "sp9832e-1h10-gofu.dts"
9
10 &fs_system {
11     fsmgr_flags = "wait,avb";
12 };
13
14 &fs_vendor {
15     fsmgr_flags = "wait,avb";
16 };

```

### 3. 开启 Secure Boot

因 Dm-Verity 功能是 Secure Boot 功能的部分实现，故该功能和 Secure Boot 用同一个宏开关控制，即 BOARD\_SECBOOT\_CONFIG := true 功能才会开启生效。

#### 说明

在 Android 8.1 的 go 版本上 Dm-Verity 和 Secure Boot 默认都是关闭的（出于对性能影响的考虑，Google 建议关闭此功能），如需要此功能，需要重新适配。

非 go 版本默认是开启的。

## 2.3.2 定制 key

Android 8.1 上 system 和 vendor 采用两个 key 分别签名。都在目录 vendor/sprd/proprieties-source/packimage\_scripts/signimage/sprd/config/下：

```
rsa4096_system.pem & rsa4096_system_pub.bin
```

```
rsa4096_vendor.pem & rsa4096_vendor_pub.bin
```

生成新的 key 的方式:

步骤 1 进入目录: `cd vendor/sprd/proprieties-source/packimage_scripts/signimage/sprd/config`

步骤 2 执行 `genkey.sh` 脚本, 生成对应分区的 key, 具体如下:

- 生成 system 分区的 key 执行命令: `/genkey.sh system`。
- 生成 vendor 分区的 key 执行命令: `/genkey.sh vendor`。

步骤 3 执行上述 `genkey.sh` 脚本后, 每个分区都会生成 `rsa4096_xxx.pem`、`rsa4096_xxx_pub.bin` 两个文件。

----结束

## 2.3.3 功能验证

验证 Dm-Verity 功能需要在 user 版本上验证, `userdebug` 版本上该功能是关闭的。

验证功能是否成功开启的办法: 查看 system 和 vendor 分区的挂载方式, 当挂载方式为 `dm-x` 时, 表明功能开启成功, 如下图所示。反之则是未开启。

```
sp9832e_1h10:/ $ mount
rootfs on / type rootfs (ro,seclabel)
tmpfs on /dev type tmpfs (rw,seclabel,nosuid,relatime,mode=755)
devpts on /dev/pts type devpts (rw,seclabel,relatime,mode=600)
proc on /proc type proc (rw,relatime,gid=3009,hidepid=2)
sysfs on /sys type sysfs (rw,seclabel,relatime)
selinuxfs on /sys/fs/selinux type selinuxfs (rw,relatime)
/dev/block/dm-0 on /system type ext4 (ro,seclabel,relatime,data=ordered)
/dev/block/dm-1 on /vendor type ext4 (ro,seclabel,relatime,data=ordered)
none on /acct type cgroup (rw,relatime,cgroup)
debugfs on /sys/kernel/debug type debugfs (rw,seclabel,relatime)
tmpfs on /mnt type tmpfs (rw,seclabel,relatime,mode=755,gid=1000)
none on /config type configs (rw,relatime)
```

## 2.3.4 功能关闭

修改 dts 文件中 system 和 vendor 分区对应的 `fsmgr_flags` 值, 去掉 “,avb”。

## 2.4 Android 9.0

Dm-Verity 在 Android 9.0 采用 AVB2.0 的方案。

Android 9.0 上要求必须支持 `System_as_root` 功能, 此功能要求 `ramdisk.img` 打包进 `system.img`。一旦 `system.img` 开启 Dm-Verity, system 分区的挂载将在 kernel 中进行, 且挂载成 dm 设备的参数需要 uboot 解析, 然后以 `cmdline` 的方式传递给 kernel。

Android 9.0 上增加了 product 分区, 此分区也需要用 Dm-Verity 功能保护。

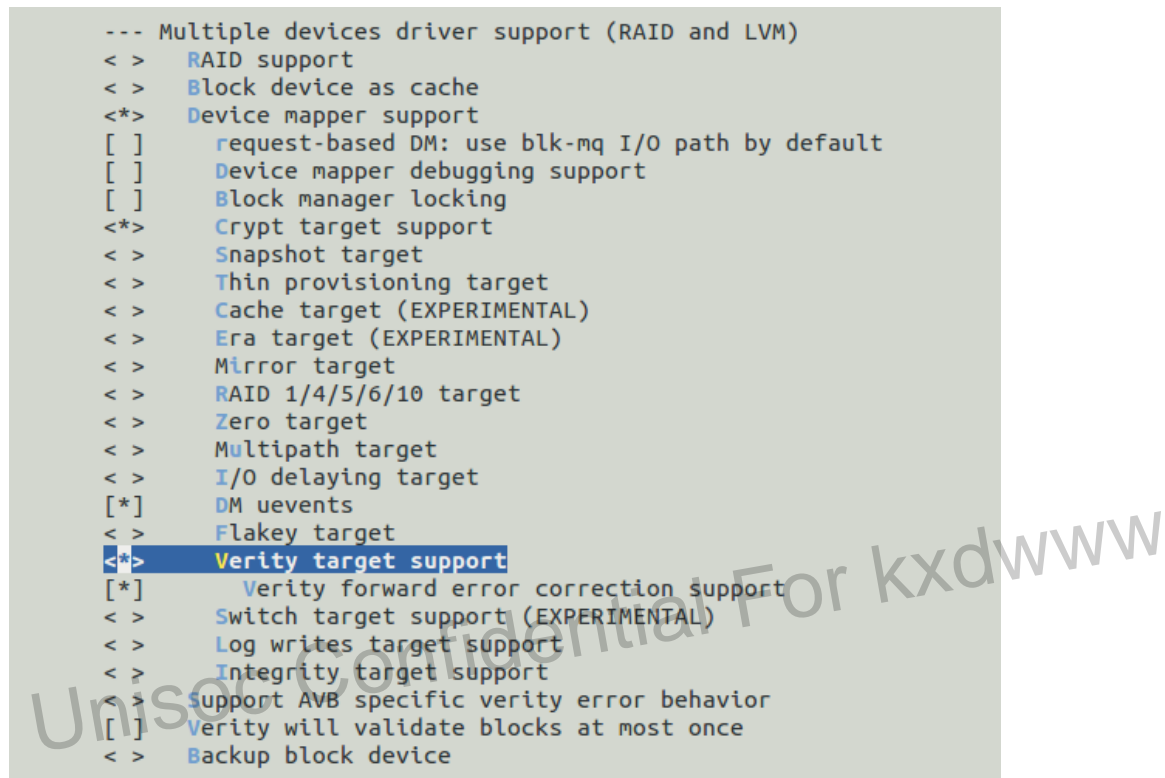
综上, system 分区将会通过解析 `cmdline` 的方式挂载, vendor 和 product 分区的挂载方式保持和 Android 8.1 一致, 由 dts 文件控制。

## 2.4.1 功能适配

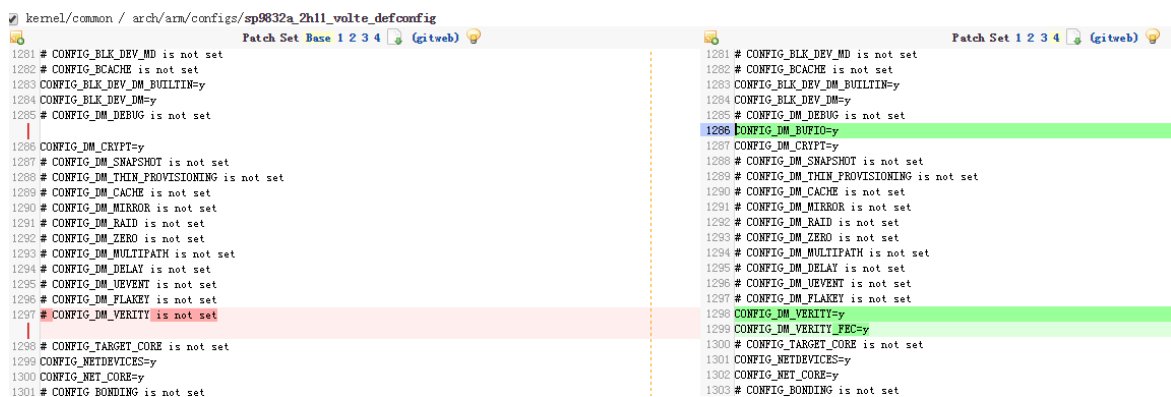
### 1. 配置 kernel config

Kuconfig 工具配置 Verity target support 和 Verity forward error correction support 两个功能，如图 2-2 所示。

图2-2 Android 9.0 Kernel 功能配置



对应 deconfig 文件修改如下：



### 2. 修改 dts 文件，开启 Dm-Verity flag

修改 dts 文件，适配 vendor 和 product 分区的挂载，并添加开启 Dm-Verity 的 flag。修改方案有两种：

- system 和 vendor 的 fsmgr\_flags 从 “wait” 改成 “wait,avb”。

```

9/ {
10     firmware {
11         android {
12             compatible = "android,firmware";
13
14             vbmeta {
15                 compatible = "android,vbmeta";
16                 parts = "vbmeta,boot,recovery,system,vendor,product";
17             };
18             fstab {
19                 compatible = "android,fstab";
20                 vendor {
21                     compatible = "android,vendor";
22                     dev = "/dev/block/platform/soc/soc:ap-ahb/20600000.sdio/by-name/vendor";
23                     type = "ext4";
24                     mnt_flags = "ro,barrier=1";
25                     fsmgr_flags = "wait,avb";
26                 };
27                 product {
28                     compatible = "android,product";
29                     dev = "/dev/block/platform/soc/soc:ap-ahb/20600000.sdio/by-name/product";
30                     type = "ext4";
31                     mnt_flags = "ro,barrier=1";
32                     fsmgr_flags = "wait,avb";
33                 };
34             };
35         };
36     };
37 };

```

- 新增一个文件包含对应的 dts 文件，然后再覆盖 fsmgr\_flags 属性。

- 1) 原 dtsi 文件增加 fs\_vendor、fs\_product 标签。

<pre> 39     fstab { 40         compatible = "android,fstab"; 41         vendor { 42             compatible = "android,vendor"; 43             dev = "/dev/block/platform/soc/soc:ap-ahb/20600000.sdio/by-name/vendor"; 44             type = "ext4"; 45             mnt_flags = "ro,barrier=1"; 46             fsmgr_flags = "wait"; 47         }; 48         product { 49             compatible = "android,product"; 50             dev = "/dev/block/platform/soc/soc:ap-ahb/20600000.sdio/by-name/product"; 51             type = "ext4"; 52             mnt_flags = "ro,barrier=1"; 53             fsmgr_flags = "wait"; 54         }; 55     }; </pre>	<pre> 39     fstab { 40         compatible = "android,fstab"; 41         fs_vendor: vendor { 42             compatible = "android,vendor"; 43             dev = "/dev/block/platform/soc/soc:ap-ahb/20600000.sdio/by-name/vendor"; 44             type = "ext4"; 45             mnt_flags = "ro,barrier=1"; 46             fsmgr_flags = "wait"; 47         }; 48         fs_product: product { 49             compatible = "android,product"; 50             dev = "/dev/block/platform/soc/soc:ap-ahb/20600000.sdio/by-name/product"; 51             type = "ext4"; 52             mnt_flags = "ro,barrier=1"; 53             fsmgr_flags = "wait"; 54         }; 55     }; </pre>
---	--

- 2) 新增一个 dts 文件，引用原 dtsi 文件定义的标签，覆盖 fsmgr\_flags 属性。

```

1 /*
2  * Spreadtrum SP9832E 1H10 GO FULL board DTS file
3  *
4  * Copyright (C) 2016-2017, Spreadtrum Communications Inc.
5  *
6  * This file is licensed under a dual GPLv2 or X11 license.
7  */
8 #include "sp9832e-1h10-gofu.dts"
9
10 &fs_system {
11     fsmgr_flags = "wait, avb";
12 };
13
14 &fs_vendor {
15     fsmgr_flags = "wait, avb";
16 };

```

### 3. 开启 Secure Boot

因 Dm-Verity 功能是 Secure Boot 功能的部分实现，故该功能和 Secure Boot 用同一个宏开关控制，即 BOARD\_SECBOOT\_CONFIG := true 功能才会开启生效。

#### 说明

在 Android 9.0 的 go 版本上 Dm-Verity 功能默认是关闭的（出于对性能影响的考虑，Google 建议关闭此功能），如需要此功能，需要重新适配。

非 go 版本默认是开启的。

## 2.4.2 定制 key

Android 9.0 上 key 的配置和 Android 8.1 原理上一致。

Android 9.0 上增加了 product 分区，即需要增加 product 分区的 key。生成新的 key 的操作如下：

步骤 1 进入目录：cd vendor/sprd/proprieties-source/packimage\_scripts/signimage/sprd/config。

步骤 2 执行 genkey.sh 脚本，生成对应分区的 key，具体如下：

- 生成 system 分区的 key 执行命令：/genkey.sh system
- 生成 vendor 分区的 key 执行命令：/genkey.sh vendor。
- 生成 product 分区的 key 执行命令：/genkey.sh product。

步骤 3 执行上述 genkey.sh 脚本后，每个分区都会生成 rsa4096\_xxx.pem、rsa4096\_xxx\_pub.bin 两个文件。

----结束

## 2.4.3 功能验证

验证功能是否成功开启的办法：查看 system、vendor、product 分区的挂载方式。

通过 mount 命令查看分区挂载，vendor 和 product 分别挂载成 dm-2 和 dm-1，说明 vendor 和 product 已经挂载成功，如下图所示。



```
s9863aih10:/ # mount
/dev/root on / type ext4 (ro,seclabel,relatime,block_validity,delalloc,barrier,user_xattr)
devtmpfs on /dev type devtmpfs (rw,seclabel,relatime,size=928860k,nr_inodes=232215,mode=755)
tmpfs on /dev type tmpfs (rw,seclabel,nosuid,relatime,mode=755)
devpts on /dev/pts type devpts (rw,seclabel,relatime,mode=600)
proc on /proc type proc (rw,relatime,gid=3009,hidepid=2)
sysfs on /sys type sysfs (rw,seclabel,relatime)
selinuxfs on /sys/fs/selinux type selinuxfs (rw,relatime)
tmpfs on /mnt type tmpfs (rw,seclabel,nosuid,nodev,noexec,relatime,mode=755,gid=1000)
/dev/block/dm-1 on /product type ext4 (ro,seclabel,relatime,block_validity,delalloc,barrier,user_xattr)
/dev/block/dm-2 on /vendor type ext4 (ro,seclabel,relatime,block_validity,delalloc,barrier,user_xattr)
none on /acct type cgroup (rw,nosuid,nodev,noexec,relatime,cpuacct)
debugfs on /sys/kernel/debug type debugfs (rw,seclabel,relatime)
none on /dev/stune type cgroup (rw,nosuid,nodev,noexec,relatime,schedtune)
none on /config type configfs (rw,nosuid,nodev,noexec,relatime)
none on /dev/cpuctl type cgroup (rw,nosuid,nodev,noexec,relatime,cpu)
none on /dev/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset,noprefix,release_agent=/sbin/cpuset_release_agent)
pstore on /sys/fs/pstore type pstore (rw,seclabel,nosuid,nodev,noexec,relatime)
adb on /dev/usb-lun0 type functionfs (rw,relatime)
tracefs on /sys/kernel/debug/tracing type tracefs (rw,seclabel,relatime)
/dev/block/mmcblk0p31 on /cache type ext4 (rw,seclabel,nosuid,nodev,noatime,noauto_da_alloc,data=ordered)
/dev/block/mmcblk0p1 on /mnt/vendor type ext4 (rw,seclabel,nosuid,nodev,noatime,noauto_da_alloc,data=ordered)
tmpfs on /storage type tmpfs (rw,seclabel,nosuid,nodev,noexec,relatime,mode=755,gid=1000)
cgroup_root on /sys/fs/cgroup type tmpfs (rw,seclabel,relatime)
cgroup_root on /sys/fs/cgroup type tmpfs (rw,seclabel,relatime)
```

System 分区的挂载无法通过 mount 命令查看，可查看到/dev/block/dm-0，说明 system 已经挂载 dm 设备成功，如下图所示。

```
s9863aih10:/dev/block # ls -l
total 0
drwxr-xr-x 2 root root 760 2018-11-14 19:06 by-name
brw----- 1 root root 253, 0 2018-11-14 19:06 dm-0
brw----- 1 root root 253, 1 2018-11-14 19:06 dm-1
brw----- 1 root root 253, 2 2018-11-14 19:06 dm-2
brw----- 1 root root 253, 3 2018-11-14 19:07 dm-3
```

综上，3 个分区都挂载成功时，Dm-Verity 功能开启成功。

## 2.4.4 功能关闭

从 Android 9.0 开始，在 user/userdebug 版本上都开启了 Dm-Verity 功能。

在 Secure Boot 功能开启的情况下（即 BOARD\_SECBOOT\_CONFIG := true），单独关闭 Dm-Verity 功能的方法如下。

- vendor 和 product 分区关闭 Dm-Verity 的方法：修改 dts 文件中分区对应的 fsmgr\_flags 值，去掉“avb”。
- system 分区关闭 Dm-Verity 的方法：PRODUCT\_DMVERITY\_DISABLE := true。

## 2.5 Android 10.0

Android 10.0 较之前版本存在以下两点变化：

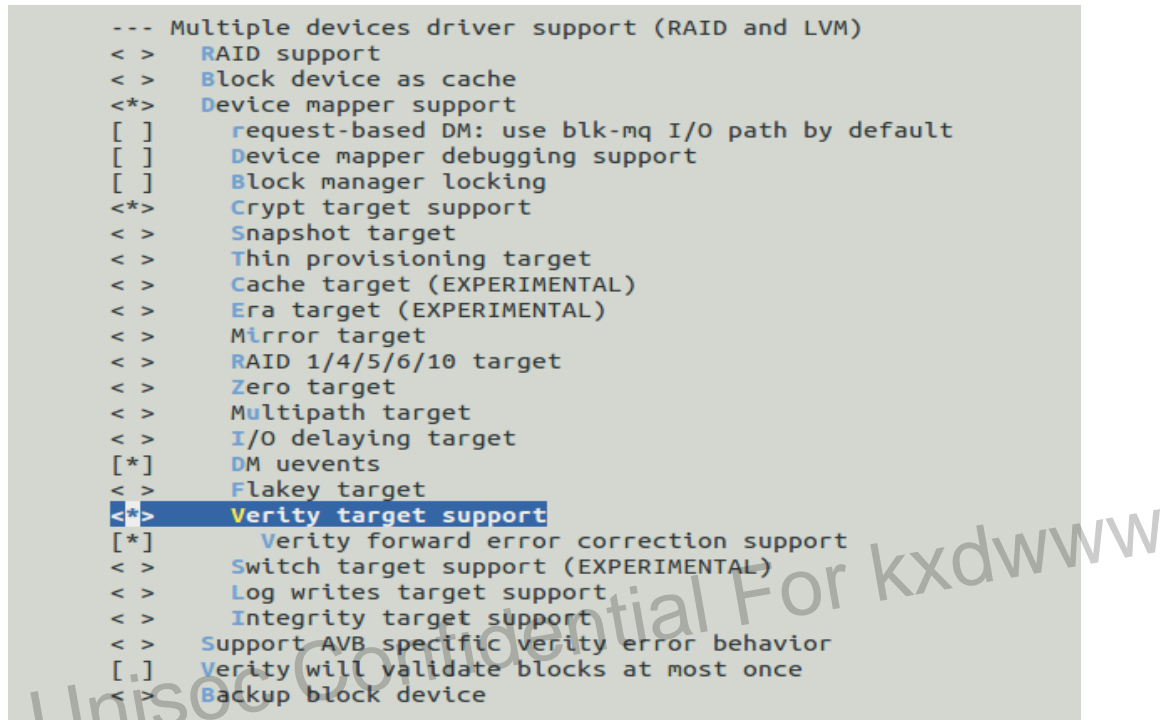
1. 新增了动态分区功能，将 system、vendor 和 product 分区打包进 super 分区，挂载的时候以逻辑分区的形式挂载，因此开机启动 bootloader 阶段无法读取用户空间动态分区的 vbmeta footer 信息。在 Android 10.0 以前的版本上，该信息存储在分区镜像文件中。在 Android 10.0 上，这部分信息被移到对应的 vbmeta 分区中。所以 Android 10.0 新增了两个 vbmeta 分区：
  - vbmeta\_system 存储 system、product 分区的 vbmeta 信息。
  - vbmeta\_vendor 存储 vendor 分区的 vbmeta 信息。
2. BSP 独立编译新增了 socko、odmko 两个分区，这两个分区也需要进行 Dm-Verity 校验。

## 2.5.1 功能适配

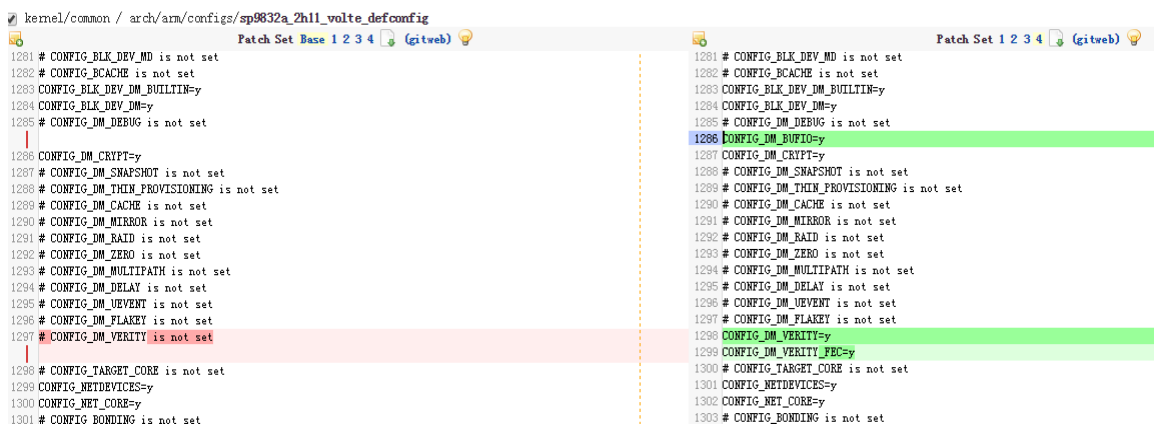
### 1. 配置 kernel config

Kuconfig 工具配置 Verity target support 和 Verity forward error correction support 两个功能，如图 2-3 所示。

图2-3 Android 10.0 Kernel 功能配置



对应 deconfig 文件修改如下：



### 2. fstab 增加 avb flag

a system/vendor/product 分区开启 Dm-Verity 方法



为了适配动态分区和 SYSTEM AS ROOT 功能，Android 10.0 新增了 fstab.ramdisk，该 fstab 主要负责内存文件系统挂载、动态分解析析和挂载，system、vendor 和 product 分区 Dm-Verity 校验标志也在该文件中配置。

由于新增了 vbmeta\_system、vbmeta\_vendor 两个分区，“avb=” flag 需要指明 vbmeta 信息存储在哪个分区。展锐方案中 vbmeta\_system 存储 system、product 分区的 vbmeta 信息，vbmeta\_vendor 存储 vendor 分区的 vbmeta 信息。具体修改如下。

<pre>1 #Dynamic partitions fstab file 2 #&lt;dev&gt; &lt;mnt_point&gt; &lt;type&gt; &lt;mnt_flags options&gt; &lt;fs_mgr_flags&gt; 3 4 system /system ext4 ro,barrier=1 wait,logical,first_stage_mount 5 vendor /vendor ext4 ro,barrier=1 wait,logical,first_stage_mount 6 product /product ext4 ro,barrier=1 wait,logical,first_stage_mount 7 /dev/block/platform/soc/soc:ap-ahb/20600000.sdio/by-name/metadata /metadata ext4 noatime, nosuid, errors=panic wait,formattable,first_stage_mount</pre>	<pre>1 #Dynamic partitions fstab file 2 #&lt;dev&gt; &lt;mnt_point&gt; &lt;type&gt; &lt;mnt_flags options&gt; &lt;fs_mgr_flags&gt; 3 4 system /system ext4 ro,barrier=1 wait,avb=vbmeta_system,logical,first_stage_mount 5 vendor /vendor ext4 ro,barrier=1 wait,avb=vbmeta_vendor,logical,first_stage_mount 6 product /product ext4 ro,barrier=1 wait,avb=vbmeta,logical,first_stage_mount 7 /dev/block/platform/soc/soc:ap-ahb/20600000.sdio/by-name/metadata /metadata ext4 noatime, nosuid, errors=panic wait,formattable,first_stage_mount</pre>
--	---

## b socko/odmko 分区开启 Dm-Verity 方法

由于 socko、odmko 并非动态分区，其 vbmeta 信息仍存储在对应分区中，“avb=” flag 指向对应分区即可，具体修改如下。

<pre>23 /mnt/vendor/socko ext4 ro, noatime, nosuid, nodev, noblk_io_submit, noauto_da_alloc wait, check 24 /mnt/vendor/odmko ext4 ro, noatime, nosuid, nodev, noblk_io_submit, noauto_da_alloc wait, check</pre>	<pre>22 /mnt/vendor/socko ext4 ro, noatime, nosuid, nodev, noblk_io_submit, noauto_da_alloc wait, avb=socko, c 23 /mnt/vendor/odmko ext4 ro, noatime, nosuid, nodev, noblk_io_submit, noauto_da_alloc wait, avb=odmko, c</pre>
--	--

## 3. fstab 增加 avb\_keys flag

对于 Android 10.0 GSI 版本，Google 原生的 system 镜像也需要做 Dm-Verity 校验。配置方法是在 fstab 中增加“avb\_keys=”的 flag，该 flag 指明 GSI system 镜像 key 的存储路径。若不配置，则 GSI system 镜像校验失败，导致 system 分区无法挂载，致使系统无法开机。

avb\_keys=/avb/q-gsi.avbpubkey:/avb/r-gsi.avbpubkey:/avb/s-gsi.avbpubkey

具体修改如下：

```
#Dynamic partitions fstab file
#<dev> <mnt_point> <type> <mnt_flags options> <fs_mgr_flags>

system /system ext4 ro,barrier=1 wait,avb=vbmeta_system,logical,first_stage_mount,avb_keys=/avb/q-gsi.avbpubkey:/avb/r-gsi.avbpubkey:/avb/s-gsi.avbpubkey
vendor /vendor ext4 ro,barrier=1 wait,avb=vbmeta_vendor,logical,first_stage_mount
product /product ext4 ro,barrier=1 wait,avb=vbmeta,logical,first_stage_mount
/dev/block/platform/soc/soc:ap-ahb/20600000.sdio/by-name/metadata /metadata ext4 noatime, nosuid, errors=panic wait,formattable,first_stage_mount
```

## 2.5.2 定制 key

Android 10.0 上 key 的配置方法和 Android 9.0 保持一致。Android 10.0 共有 system、vendor、product、socko 和 odmko 五个分区开启了 Dm-Verity 校验，每个分区都需要生成对应的 key，具体方法如下：

步骤 1 进入目录：cd bsp/tools/secureboot\_key/config/。

步骤 2 执行 genkey.sh 脚本，生成对应分区的 key，具体如下：

- 生成 system 分区的 key 执行命令：/genkey.sh system。
- 生成 product 分区的 key 执行命令：/genkey.sh product。
- 生成 vendor 分区的 key 执行命令：/genkey.sh vendor。
- 生成 socko 分区的 key 执行命令：/genkey.sh socko。
- 生成 odmko 分区的 key 执行命令：/genkey.sh odmko。

步骤 3 执行上述 genkey.sh 脚本后，每个分区都会生成 rsa4096\_xxx.pem、rsa4096\_xxx\_pub.bin 两个文件。

----结束

## 2.5.3 功能验证

### 1. 确认 system/vendor/product 分区是否开启 Dm-Verity 校验

由于 Android 10.0 新增了动态分区功能，该功能基于 Linux 内核 Dm-Linear，不开启 Dm-Verity 情况下，system、vendor 和 product 分区也会挂载为 dm 设备，因此这三个分区无法通过查看设备挂载状态确认。可以通过查看串口 log，分区挂载前是否构建 Dm-Verity 映射表确认是否开启 Dm-Verity，具体 log 信息如下：

#### - system 分区

```
init: [libfs_avb]Built verity table: '1 /dev/block/dm-0 /dev/block/dm-0 4096 4096 313614 313614 sha1
4838d910705aeabeba55416e26863500bc7eec0b e31af2081be98ea257df30db080d570b0cad5b5 10
use_fec_from_device /dev/block/dm-0 fec_roots 2 fec_blocks 316086 fec_start 316086
restart_on_corruption ignore_zero_blocks'
device-mapper: verity: sha1 using implementation "sha1-ce"
init: [libfs_mgr]__mount(source=/dev/block/dm-3,target=/system,type=ext4)=0: Success
device-mapper: verity: sha1 using implementation "sha1-ce"
```

#### - vendor 分区

```
init: [libfs_avb]Built verity table: '1 /dev/block/dm-1 /dev/block/dm-1 4096 4096 82470 82470 sha1
230d7d714a21f62c84f7306ff7a467533c761243 d9a9ce03416b28207ac29e50a9bbd3546bbfe154 10
use_fec_from_device /dev/block/dm-1 fec_roots 2 fec_blocks 83122 fec_start 83122
restart_on_corruption ignore_zero_blocks'
device-mapper: verity: sha1 using implementation "sha1-ce"
init: [libfs_mgr]__mount(source=/dev/block/dm-4,target=/vendor,type=ext4)=0: Success
```

#### - product 分区

```
init: [libfs_avb]Built verity table: '1 /dev/block/dm-2 /dev/block/dm-2 4096 4096 360010 360010 sha1
5f28681e9ae5c356dbbcc7f4c632a8301fcf612c 0fe99a5dfc2297cd2294f7c880124b87c4bc042a 10
use_fec_from_device /dev/block/dm-2 fec_roots 2 fec_blocks 362846 fec_start 362846
restart_on_corruption ignore_zero_blocks'
device-mapper: verity: sha1 using implementation "sha1-ce"
init: [libfs_mgr]__mount(source=/dev/block/dm-5,target=/product,type=ext4)=0: Success
```

### 2. 确认 socko/odmko 分区是否开启 Dm-Verity 校验

新增的 socko 和 odmko 分区由于不是动态分区，仍可通过是否挂载为 dm 设备判断是否开启 Dm-Verity。执行 mount 命令，socko 和 odmko 分区挂载状态如下：

```
/dev/block/dm-6 on /mnt/vendor/socko type ext4 (ro,seclabel,nosuid,nodev,noatime
,noauto_da_alloc)
/dev/block/dm-7 on /mnt/vendor/odmko type ext4 (ro,seclabel,nosuid,nodev,noatime
,noauto_da_alloc)
```

### 3. 确认使用的哈希算法

哈希运算使用 SHA 算法，对于不支持 ARM-CE 的芯片，使用 sha1-neon 实现，串口 log 打印如下：

```
device-mapper: verity: sha1 using implementation "sha1-neon"
```

对于支持 ARM-CE 的芯片，使用 sha1-ce 实现，串口 log 打印如下：

```
device-mapper: verity: sha1 using implementation "sha1-ce"
```

展锐目前支持 Dm-verity 的量产芯片中，除 SC7731E 不支持 ARM-CE 外，其余芯片均支持。需根据如上串口 log 确认使用的算法实现，对于支持 ARM-CE 的芯片，sha1-ce 性能优于 sha1-neon。

## 2.5.4 功能关闭

### 1. 量产版本关闭 Dm-Verity 校验

Google CDD 文档要求开启 Verified Boot 功能，所以对于需要过 Google 认证的项目，都需要开启 Dm-Verity 校验。但如果有关闭该功能的需求，也支持通过以下方式关闭：

- 找到对应 board 的 fstab 文件：fstab.ramdisk、fstab.xxx、fstab.xxx.f2fs...
- 删除所有“avb=vbmeta\_system”、“avb=vbmeta\_vendor”、“avb=vbmeta” flag
- 删除 system 分区的 avb\_keys=/avb/q-gsi.avbpubkey:/avb/r-gsi.avbpubkey:/avb/s-gsi.avbpubkey
- 重新编译版本。

### 2. 临时关闭 Dm-Verity 校验

如调试需要临时关闭或需要执行 remount 操作，可先根据《Android 10.0 设备解锁指导手册》文档解锁设备后执行以下命令。

```
$ adb roo
$ adb disable-verity
$ adb reboot
$ adb wait-for-device
$ adb root
$ adb remount
```

## 2.6 Android 11.0

Android 11.0 新增了一个 system\_ext 动态分区，与 system、system\_ext、vendor 以及 product 分区一起打包进 super 分区，挂载的时候以逻辑分区的形式挂载。与 Android 10.0 相同，由于开机启动 bootloader 阶段无法读取用户空间动态分区，需要将这几个动态分区的 vbmeta 信息存储在单独的 vbmeta 分区。Android 11.0 上，每个动态分区分别对应一个 vbmeta 分区，具体如表 2-1 所示。

表2-1 Android 11.0 动态分区与 Vbmeta 分区对应关系

动态分区	vbmeta 分区
system	system_vbmeta
system_ext	system_ext_vbmeta
vendor	vendor_vbmeta
product	product_vbmeta

通过下载工具 ResearchDownload 重新下载 super 分区，或者通过 fastbootd 命令重新下载动态分区后，需要同步更新相应的 vbmeta 分区。

## 2.6.1 功能适配

### 1. 配置 kernel config

kuconfig 工具配置 Verity target support、Verity forward error correction support 和 Support AVB specific verity error behavior 三个功能，如图 2-4 所示。

图2-4 Android 11.0 Kernel 功能配置

```

--- Multiple devices driver support (RAID and LVM)
< > RAID support
< > Block device as cache
<*> Device mapper support
[ ] request-based DM: use blk-mq I/O path by default
[ ] Device mapper debugging support
[ ] Block manager locking
<*> Crypt target support
<*> Default-key target support
<*> Snapshot target
< > Thin provisioning target
< > Cache target (EXPERIMENTAL)
< > Era target (EXPERIMENTAL)
< > Mirror target
< > RAID 1/4/5/6/10 target
< > Zero target
< > Multipath target
< > I/O delaying target
[*] DM uevents
< > Flakey target
<*> Verity target support
[*] Verity forward error correction support
< > Switch target support (EXPERIMENTAL)
< > Log writes target support
< > Integrity target support
<*> Support AVB specific verity error behavior
[ ] Android verity target support
[ ] Verity will validate blocks at most once
<*> Backup block device

```

### 2. fstab 增加 avb flag

#### a system/system\_ext/vendor/product 分区开启 Dm-Verity 方法

与 Android 10.0 相同，为了适配动态分区和 SYSTEM AS ROOT 功能，fstab.ramdisk 主要负责内存文件系统挂载、动态分区解析和挂载，system、system\_ext、vendor 和 product 分区的 Dm-Verity 校验标志也在该文件中配置。同时“avb=”标签需要指明动态分区的 vbmeta 信息存储在哪个 vbmeta 分区。以 UMS512 为例，可以参考 IDH 包中的文件配置如下：

```
/sprdroidr_trunk/device/sprd/mpool/module/partition/msoc/sharkl5Pro/fstab.ramdisk
```

#### b socko/odmko 分区开启 Dm-Verity 方法

由于 socko、odmko 并非动态分区，其 vbmeta 信息仍存储在对应分区中，“avb=”标签指向对应分区即可。以 UMS512 为例，可以参考 IDH 包中的文件配置如下：

```
/sprdroidr_trunk/device/sprd/mpool/module/partition/msoc/sharkl5Pro/fstab
```

### 3、fstab 增加 avb\_keys flag

Android 11.0 GSI 版本，Google 原生 system 镜像也会做 Dm-Verity 校验，需要增加“avb\_keys=”标签，指明 GSI system 镜像 key 的存储路径，否则 GSI 版本会由于挂载 system 分区失败导致无法开机。具体修改需要在 fstab.ramdisk 中，system 分区那一行增加以下 flag：

```
avb_keys=/avb/q-gsi.avbpubkey:/avb/r-gsi.avbpubkey:/avb/s-gsi.avbpubkey
```

## 2.6.2 定制 key

Android 11.0 上 key 的配置方法和 Android 8.1、Android 9.0 以及 Android 10.0 保持一致。Android 11.0 共有 system、system\_ext、vendor、product、socko 和 odmko 六个分区开启了 Dm-Verity 校验，每个分区都需要生成对应的 key，具体方法如下：

步骤 1 进入目录：cd bsp/tools/secureboot\_key/config/。

步骤 2 执行 genkey.sh 脚本，生成对应分区的 key，具体如下：

- 生成 system 分区的 key 执行命令：/genkey.sh system。
- 生成 system\_ext 分区的 key 执行命令：/genkey.sh system\_ext。
- 生成 product 分区的 key 执行命令：/genkey.sh product。
- 生成 vendor 分区的 key 执行命令：/genkey.sh vendor。
- 生成 socko 分区的 key 执行命令：/genkey.sh socko。
- 生成 odmko 分区的 key 执行命令：/genkey.sh odmko。

步骤 3 执行上述 genkey.sh 脚本后，每个分区都会生成 rsa4096\_XXX.pem、rsa4096\_XXX\_pub.bin 两个文件，将这些文件预置在如下目录：bsp/tools/secureboot\_key/config/。

----结束

## 2.6.3 功能验证

### 1. 确认 system/system\_ext/vendor/product 分区是否开启 Dm-Verity 校验

由于启用了动态分区功能，该功能基于 Linux 内核 Dm-Linear，不开启 Dm-Verity 情况下，system、system\_ext、vendor 和 product 分区也会挂载为 dm 设备，因此这五个分区无法通过查看设备挂载状态确认。可以通过 dmctl 命令获取 Dm-Verity 映射表确认：

- system 分区

```
$ adb root
$ adb shell dmctl table system-verity
```

输出：

```
Targets in the device-mapper table for system-verity:
0-1100832: verity, 1 252:0 252:0 4096 4096 137604 137604 sha1
586ff4c1db6b0b435718090bf03c3e7de1dc707a 79ee3646176a38b29018a0aaf7dd1d92d0502d1d 11
restart_on_corruption ignore_zero_blocks check_at_most_once use_fec_from_device 252:0 fec_blocks
138690 fec_start 138690 fec_roots 2
```

- system\_ext 分区

```
$ adb root
```

```
$ adb shell dmctl table system_ext-verity
```

输出:

Targets in the device-mapper table for system\_ext-verity:

```
0-550464: verity, 1 252:1 252:1 4096 4096 68808 68808 sha1
da34fff41964cfbdc53811fb00ad315708657385 d2dded84053f72116b559c7e358cc0278a41026a 11
restart_on_corruption ignore_zero_blocks check_at_most_once use_fec_from_device 252:1 fec_blocks
69352 fec_start 69352 fec_roots 2
```

#### - vendor 分区

```
$ adb root
```

```
$ adb shell dmctl table vendor-verity
```

输出:

Targets in the device-mapper table for vendor-verity:

```
0-368312: verity, 1 252:2 252:2 4096 4096 46039 46039 sha1
c21d791ac53181d198588d992d9250828453cfb6 69d1c37b2bc63759af2b0b306ce0feba59e147e8 11
restart_on_corruption ignore_zero_blocks check_at_most_once use_fec_from_device 252:2 fec_blocks
46403 fec_start 46403 fec_roots 2
```

#### - product 分区

```
$ adb root
```

```
$ adb shell dmctl table product-verity
```

输出:

Targets in the device-mapper table for product-verity:

```
0-1415920: verity, 1 252:3 252:3 4096 4096 176990 176990 sha1
f4b29d17766b015e94ce03bb2baa75ee769db5e7 35dc3648baad4450b60b45a4bbff96a9c9c354da 11
restart_on_corruption ignore_zero_blocks check_at_most_once use_fec_from_device 252:3 fec_blocks
178385 fec_start 178385 fec_roots 2
```

### 说明

当输出信息中有“Targets in the device-mapper table for product-verity:”字样，表示开启校验，下文情况相同。

## 2. 确认 socko/odmko 分区是否开启 Dm-Verity 校验

新增的 socko 和 odmko 分区由于不是动态分区，仍可通过是否挂载为 dm 设备判断是否开启 Dm-Verity。执行 mount 命令，socko 和 odmko 分区挂载状态如下：

```
/dev/block/dm-8 on /mnt/vendor/socko type ext4 (ro,seclabel,nosuid,nodev,noatime,noauto_da_alloc)
/dev/block/dm-9 on /mnt/vendor/odmko type ext4 (ro,seclabel,nosuid,nodev,noatime,noauto_da_alloc)
```

## 3. 确认使用的哈希算法

哈希运算使用 SHA 算法，对于不支持 ARM-CE 的芯片，使用 sha1-neon 实现，串口 log 打印如下：

```
device-mapper: verity: sha1 using implementation "sha1-neon"
```

对于支持 ARM-CE 的芯片，使用 sha1-ce 实现，串口 log 打印如下：

```
device-mapper: verity: sha1 using implementation "sha1-ce"
```



展锐目前支持 Dm-Verity 的量产芯片中，除 SC7731E 不支持 ARM-CE 外，其余芯片均支持。需根据如上串口 log 确认使用的算法实现，对于支持 ARM-CE 的芯片，sha1-ce 性能优于 sha1-neon。

## 2.6.4 功能关闭

### 1. 量产版本关闭 Dm-Verity 校验

Google CDD 文档要求开启 Verified Boot 功能，所以对于需要过 Google 认证的项目，都需要开启 Dm-Verity 校验。但如果有关闭该功能的需求，也支持通过以下方式关闭：

- a 找到对应 board 的 fstab 文件：fstab.ramdisk、fstab.xxx、fstab.xxx.f2fs...
- b 删除所有“avb=vbmata\_system”、“avb=vbmata\_vendor”、“avb=vbmata” flag
- c 删除 system 分区的“avb\_keys=/avb/q-gsi.avbpubkey:/avb/r-gsi.avbpubkey:/avb/s-gsi.avbpubkey”
- d 重新编译版本

### 2. 临时关闭 Dm-Verity 校验

与 Android 10.0 不同，Android 11.0 如调试需要执行 remount 操作，在解锁 bootloader 后，执行以下命令即可，不再需要单独执行 adb disable-verity，但需要重启手机才能生效。

```
$ adb root
$ adb remount
$ adb reboot
```

## 2.7 Android 12.0

和 Android 11.0 相比，Android 12.0 新增了一个 vendor\_dlkm 动态分区，并移除了 socko 和 odmko 分区。system、system\_ext、vendor、product 和 vendor\_dlkm 分区一起打包进 super 分区，挂载的时候以逻辑分区的形式挂载。与 Android 11.0 相同，由于开机启动 bootloader 阶段无法读取用户空间动态分区，需要将这几个动态分区的 vbmeta 信息存储在单独的 vbmeta 分区。Android 12.0 上，system、system\_ext 以及 product 这三个动态分区分别对应一个 vbmeta 分区，vendor 及 vendor\_dlkm 共用一个 vbmeta\_vendor 分区，具体如下表 2-2 所示。

表2-2 Android 12.0 动态分区与 Vbmeta 分区对应关系

动态分区	vbmeta 分区
system	system_vbmeta
system_ext	system_ext_vbmeta
vendor	vendor_vbmeta
product	product_vbmeta
vendor_dlkm	vbmeta_vendor

通过下载工具 ResearchDownload 重新下载 super 分区，或者通过 fastbootd 命令重新下载动态分区后，需要同步更新相应的 vbmeta 分区。

### 2.7.1 功能适配

#### 1. 配置 kernel config

kuconfig 工具配置 Verity target support 和 Verity forward error correction support 两个功能。

图2-5 Android 12.0 Kernel 功能配置

```

--- Multiple devices driver support (RAID and LVM)
< > RAID support
< > Block device as cache
< * > Device mapper support
[ ] Device mapper debugging support
[ ] Block manager locking
< > Unstriped target
< * > Crypt target support
< * > Default-key target support
< * > Snapshot target
< > Thin provisioning target
< > Cache target (EXPERIMENTAL)
< > Writecache target
< > Era target (EXPERIMENTAL)
< > Clone target (EXPERIMENTAL)
< > Mirror target
< > RAID 1/4/5/6/10 target
< > Zero target
< > Multipath target
< > I/O delaying target
< > Bad sector simulation target
[ ] DM "dm-mod.create=" parameter support
[ * ] DM uevents
< > Flakey target
< * > Verity target support
[ ] Verity data device root hash signature verification support
[ * ] Verity forward error correction support
< > Switch target support (EXPERIMENTAL)
< > Log writes target support
< > Integrity target support
< * > Backup block device
< * > Block device in userspace

```

对应 deconfig 文件修改如下：

kernel/common / arch/arm/configs/sp9832a_2h11_volte_defconfig	Patch Set 1 2 3 4 (gitweb)
1281 # CONFIG_BLK_DEV_MD is not set	1281 # CONFIG_BLK_DEV_MD is not set
1282 # CONFIG_BCACHE is not set	1282 # CONFIG_BCACHE is not set
1283 CONFIG_BLK_DEV_DM_BUILTIN=y	1283 CONFIG_BLK_DEV_DM_BUILTIN=y
1284 CONFIG_BLK_DEV_DM=y	1284 CONFIG_BLK_DEV_DM=y
1285 # CONFIG_DM_DEBUG is not set	1285 # CONFIG_DM_DEBUG is not set
1286 CONFIG_DM_CRYPT=y	1286 CONFIG_DM_CRYPT=y
1287 # CONFIG_DM_SNAPSHOT is not set	1287 # CONFIG_DM_SNAPSHOT is not set
1288 # CONFIG_DM_THIN_PROVISIONING is not set	1288 # CONFIG_DM_THIN_PROVISIONING is not set
1289 # CONFIG_DM_CACHE is not set	1289 # CONFIG_DM_CACHE is not set
1290 # CONFIG_DM_MIRROR is not set	1290 # CONFIG_DM_MIRROR is not set
1291 # CONFIG_DM_RAID is not set	1291 # CONFIG_DM_RAID is not set
1292 # CONFIG_DM_ZERO is not set	1292 # CONFIG_DM_ZERO is not set
1293 # CONFIG_DM_MULTIPATH is not set	1293 # CONFIG_DM_MULTIPATH is not set
1294 # CONFIG_DM_DELAY is not set	1294 # CONFIG_DM_DELAY is not set
1295 # CONFIG_DM_UEVENT is not set	1295 # CONFIG_DM_UEVENT is not set
1296 # CONFIG_DM_FLAKEY is not set	1296 # CONFIG_DM_FLAKEY is not set
1297 # CONFIG_DM_VERITY is not set	1297 # CONFIG_DM_VERITY=y
1298 # CONFIG_TARGET_CORE is not set	1298 # CONFIG_TARGET_CORE is not set
1299 CONFIG_NETDEVICES=y	1299 CONFIG_NETDEVICES=y
1300 CONFIG_NET_CORE=y	1300 CONFIG_NET_CORE=y
1301 # CONFIG_BONDING is not set	1301 # CONFIG_BONDING is not set

## 2. fstab 增加 avb flag

system/system\_ext/vendor/product/vendor\_dlkm 分区开启 Dm-Verity 方法如下：

与 Android 11.0 相同，为了适配动态分区和 SYSTEM AS ROOT 功能，fstab.ramdisk 主要负责内存文件系统挂载、动态分区分析和挂载，system、system\_ext、vendor、product 和 vendor\_dlkm 分区的



Dm-Verity 校验标志也在该文件中配置。同时“avb=”标签需要指明动态分区的 vbmeta 信息存储在哪个 vbmeta 分区。以 SC9863A 为例，可以参考 IDH 包中的以下文件配置：

```
/sprdroid12_trunk/device/sprd/mpool/module/partition/msoc/shark15Pro/mfeature/kernel/kernel5.4/fstab.ramdisk
```

### 3. fstab 增加 avb\_keys flag

Android 12.0 GSI 版本，Google 原生 system 镜像也会做 Dm-Verity 校验，需要增加“avb\_keys=”标签，指明 GSI system 镜像 key 的存储路径，否则 GSI 版本会由于挂载 system 分区失败导致无法开机。具体修改需要在 fstab.ramdisk 中，system 分区那一行增加以下 flag：

```
avb_keys=/avb/q-gsi.avbpubkey:/avb/r-gsi.avbpubkey:/avb/s-gsi.avbpubkey
```

具体修改如下：

```
1 #Dynamic partitions fstab file.For V-A/B Feature.
2 #<dev> <mnt_point> <type> <mnt_flags options> <fs_mgr_flags>
3
4 system /system ext4 ro,barrier=1 wait,avb=vbmeta_system,logical,first_stage_mount,avb_keys=/avb/q-gsi.avbpubkey:/avb/r-gsi.avbpubkey:/avb/s-gsi.avbpubkey,slotselect
5 system_ext /system_ext ext4 ro,barrier=1 wait,avb=vbmeta_system_ext,logical,first_stage_mount,slotselect
6 vendor /vendor ext4 ro,barrier=1 wait,avb=vbmeta_vendor,logical,first_stage_mount,slotselect
7 product /product ext4 ro,barrier=1 wait,avb=vbmeta_product,logical,first_stage_mount,slotselect
8 vendor_dlkm /vendor_dlkm ext4 noatime,ro,errors=panic wait,avb=vbmeta_vendor,logical,first_stage_mount,slotselect
9 /dev/block/by-name/metadata /metadata ext4 nodev,noatime,nosuid,errors=panic wait,formattable,first_stage_mount,check
10
```

## 2.7.2 定制 key

Android 12.0 上 key 的配置方法和 Android 11.0 保持一致。

Android 12.0 共有 system、system\_ext、vendor、product 和 vendor\_dlkm 五个分区开启了 Dm-Verity 校验，每个分区都需要生成对应的 key，具体方法如下：

步骤 1 进入目录：cd bsp/tools/secureboot\_key/config。

步骤 2 执行 genkey.sh 脚本，生成对应分区的 key，具体如下：

- 生成 system 分区的 key 执行命令：/genkey.sh system。
- 生成 system\_ext 分区的 key 执行命令：/genkey.sh system\_ext。
- 生成 vendor 分区的 key 执行命令：/genkey.sh vendor。
- 生成 product 分区的 key 执行命令：/genkey.sh product。
- 生成 vendor\_dlkm 分区的 key 执行命令：/genkey.sh vendor。

步骤 3 执行上述 genkey.sh 脚本后，每个分区都会生成 rsa4096\_xxx.pem、rsa4096\_xxx\_pub.bin 两个文件，将这些文件预置在如下目录：bsp/tools/secureboot\_key/config/。

----结束

## 2.7.3 功能验证

### 1. 确认 system/system\_ext/vendor/vendor\_dlkm/product 分区是否开启 Dm-Verity 校验

由于启用了动态分区功能，该功能基于 Linux 内核 Dm-Linear，不开启 Dm-Verity 情况下，system、system\_ext、vendor 和 product 分区也会挂载为 dm 设备，因此这五个分区无法通过查看设备挂载状态确认。可以通过 dmctl 命令获取 Dm-Verity 映射表确认：

- system 分区

```
$ adb root
```

```
$ adb shell dmctl table system-verity
```

输出:

Targets in the device-mapper table for system-verity:

```
0-1264208: verity, 1 253:0 253:0 4096 4096 158026 158026 sha1
e8b5bac3a431f7407ec6618a1ab45c06768e4a5c 36e056125216a97e63f55ca2b86424319cae0714 10
restart_on_corruption ignore_zero_blocks use_fec_from_device 253:0 fec_blocks 159272 fec_start
159272 fec_roots 2
```

– system\_ext 分区

```
$ adb root
```

```
$ adb shell dmctl table system_ext-verity
```

输出:

Targets in the device-mapper table for system\_ext-verity:

```
0-860720: verity, 1 253:1 253:1 4096 4096 107590 107590 sha1
1b316a2828beef6d8386f530d411dbfe5d912cdc 5808b30c9a9215c4896db9a05384033005c17d5a 10
restart_on_corruption ignore_zero_blocks use_fec_from_device 253:1 fec_blocks 108439 fec_start
108439 fec_roots 2
```

– vendor 分区

```
$ adb root
```

```
$ adb shell dmctl table vendor-verity
```

输出:

Targets in the device-mapper table for vendor-verity:

```
0-731712: verity, 1 253:2 253:2 4096 4096 91464 91464 sha1
4114f99def682bbb6a218b17f9fbb6abc61afd02 ece8d0d8ec4b2d3e151b2406de310c922a0a15f2 10
restart_on_corruption ignore_zero_blocks use_fec_from_device 253:2 fec_blocks 92186 fec_start 92186
fec_roots 2
```

– product 分区

```
$ adb root
```

```
$ adb shell dmctl table product-verity
```

输出:

Targets in the device-mapper table for product-verity:

```
0-473528: verity, 1 253:3 253:3 4096 4096 59191 59191 sha1
f7ca7905de75e2e3b1af7290a07cf4d659fcab1a 88cdc2d74ab78a51f4f42a8f78011bea595b60cf 10
restart_on_corruption ignore_zero_blocks use_fec_from_device 253:3 fec_blocks 59659 fec_start 59659
fec_roots 2
```

– vendor\_dlkm 分区

```
$ adb root
```

```
$ adb shell dmctl table vendor_dlkm-verity
```

输出:

Targets in the device-mapper table for vendor\_dlkm-verity:

```
0-13712: verity, 1 253:4 253:4 4096 4096 1714 1714 sha1
083c3d91940748885d8a6f82ce598c0cd410cef1 b31acd5ab19d95e4a238ce8886ac1c4ebc2a8490 10
restart_on_corruption ignore_zero_blocks use_fec_from_device 253:4 fec_blocks 1729 fec_start 1729
fec_roots 2
```

## 2. 确认使用的哈希算法

哈希运算使用 SHA 算法，对于不支持 ARM-CE 的芯片，使用 sha1-neon 实现，串口 log 打印如下：

```
device-mapper: verity: sha1 using implementation "sha1-neon"
```

对于支持 ARM-CE 的芯片，使用 sha1-ce 实现，串口 log 打印如下：

```
device-mapper: verity: sha1 using implementation "sha1-ce"
```

展锐目前支持 Dm-Verity 的量产芯片中，除 SC7731E 不支持 ARM-CE 外，其余芯片均支持。需根据如上串口 log 确认使用的算法实现，对于支持 ARM-CE 的芯片，sha1-ce 性能优于 sha1-neon。

## 2.7.4 功能关闭

### 1. 量产版本关闭 Dm-Verity 校验

Google CDD 文档要求开启 Verified Boot 功能，所以对于需要过 Google 认证的项目，都需要开启 Dm-Verity 校验。但如果有关闭该功能的需求，也支持通过以下方式关闭：

- 找到对应 board 的 fstab 文件：fstab.ramdisk、fstab.xxx、fstab.xxx.f2fs...
- 删除所有“avb=vbmeta\_system”、“avb=vbmeta\_system\_ext”、“avb=vbmeta\_vendor”、“avb=vbmeta\_product”等 flag
- 删除 system 分区的 avb\_keys=/avb/q-gsi.avbpubkey:/avb/r-gsi.avbpubkey:/avb/s-gsi.avbpubkey
- 重新编译版本

### 2. 临时关闭 Dm-Verity 校验

如调试需要临时关闭或需要执行 remount 操作，可先根据《Android 10.0 设备解锁指导手册》。

文档解锁设备后执行以下命令：

```
$ adb root
$ adb disable-verity
$ adb reboot
$ adb wait-for-device
$ adb root
$ adb remount
```

# 3 常见问题

## 3.1 性能影响

开启 Dm-Verity 功能后，增加了校验逻辑分区 Block 数据流程，会对性能造成一定的影响。根据 Dm-Verity 的工作原理，对性能的影响主要体现在两个方面：

- 开机挂载 system、vendor 和 product 分区前，需要对相应的 vbmeta footer 数据进行校验。挂载分区后，系统启动过程从 system、vendor 和 product 分区读取数据都要进行 Hash 校验，增加耗时预计在 1~3s 左右；
- 系统运行时，从 system、vendor 和 product 等只读分区读取的数据都需要进行 Hash 校验，因此对只读分区的 IO 会有一定影响。

## 3.2 OTA 影响

当使能 Dm-Verity 之后，需要将 OTA 升级方式切换为基于 Block 的升级方式。即 OTA 更新要在块设备层操作，并同时更新哈希树和 Verity metadata。

## 3.3 Android 9.0 userdebug 版本 remount 失败

Android 9.0 上在非 go 的项目上都是开启了 Secure Boot 和 Dm-Verity 功能的，在 go 的项目上只开启了 Secure Boot 功能，未开启 Dm-Verity 功能。

因此在非 go 项目的 userdebug 版本上直接使用以下命令会失败：

```
adb root;
adb remount;
```

因为一旦开启 Dm-Verity 功能去保护 system 分区，system 则无法被 remount 成可读写的分区。

### 解决方案

首先需要 unlock bootloader，然后再 remount 手机即可成功。

### 说明

unlock bootloader 的详细操作请参考文档 *Android9.0 Unlock Bootloader Guide*。

### 临时方案 s

为了方便 debug，在 userdebug 版本中对应的 board 中临时定义 `PRODUCT_DMVERITY_DISABLE := true`，可关闭 Dm-Verity 功能，这样直接 remount 可成功。但后续量产版本切记要将上述临时修改回退。

如需要在 userdebug 版本上单独关闭 Dm-Verity 功能，可通过用 userdebug 的判断条件控制 PRODUCT\_DMVERITY\_DISABLE，参考修改如下：

```
ifeq ($(TARGET_BUILD_VARIANT),userdebug)
    PRODUCT_DMVERITY_DISABLE := true
endif
```

## 3.4 功能异常排查指引

Dm-Verity 功能在平台参考 Board 上已做适配，若新增 Board 配置后出现无法开机等异常，可以参照以下流程排查：

- 步骤 1 按照平台版本，检查本文档各项配置是否正确，尤其注意 Board 的适配。
- 步骤 2 检查编译 log，确认是否编译出 build\_verity\_tree 和 build\_verity\_metadata 信息。
- 步骤 3 串口抓取开机 kernel.log，检查 system 和 vendor 分区是否挂载正常，关键字为 fs\_mgr。
- 步骤 4 针对 Android 7.0，需要确认 pac 包中 system.img 的大小和 xml 分区表上 system 分区的大小是否一致。

----结束

Unisoc Confidential For kxdwww

# 4

## 参考文档

---

- 《Android 10.0 设备解锁指导手册》
- *Android9.0 Unlock Bootloader Guide*
- Verified-Boot 官网信息参考 <https://source.android.com/security/verifiedboot/verified-boot>
- Dm-Verity 官网信息参考 <https://source.android.com/security/verifiedboot/dm-verity>

Unisoc Confidential For kxdwww