



CV184X 安全启动使用手册

Version: 1.0.1

Release date: 2025-06-26

©2025 北京晶视智能科技有限公司
本文件所含信息归北京晶视智能科技有限公司所有。
未经授权，严禁全部或部分复制或披露该等信息。

目录

1	声明	2
2	安全启动介绍	3
2.1	镜像结构	3
2.2	安全启动流程	4
3	安全镜像生成	5
3.1	四层保护对照	5
3.2	密钥与工具（速查）	6
4	eFuse 烧写	7
5	FIP 签名与加密及 eFuse	8
5.1	生成 FIP 用的密钥	8
5.2	打开 FSBL 安全启动并编译	8
5.3	对 FIP 做签名与加密	9
5.4	获取 eFuse 要烧写的值（开发机）	9
5.4.1	HASH0_PUBLIC（64 位十六进制）	9
5.4.2	LOADER_EK（32 位十六进制）	9
5.5	烧写 eFuse（U-Boot 中，一次性、不可逆）	9
5.6	烧录镜像（eFuse 烧写完成之后）	10
5.7	流程简表（FIP 侧）	10
5.8	FIP 密钥一览（my_fip_keys/）	10
5.9	与 FIT 的关系	11
6	FIT 镜像安全启动	12
6.1	生成 FIT 验签与加密用的密钥与材料	12
6.2	打开配置（内核与 U-Boot）	12
6.3	编译与产物	13
6.4	FIT 相关密钥一览（ramdisk/keys/）	13
6.5	流程简表（FIT 侧）	13
6.6	与 FIP 的配合	13
7	SquashFS RootFS 签名与内核验签	14
7.1	整体目标	14
7.2	使用与排查建议	14
7.3	相关文件	15
8	DATA 分区（userdata）签名与验签	16
8.1	概述	16
8.2	使用与验证方式	16
8.3	相关文件一览	17

修订记录

Revision	Date	Description
1.0.0	2025-03-04	初稿
1.0.1	2025-06-26	更新启动流程
1.0.2	2026-04-08	增补 FIT/FIP、RootFS、DATA（userdata）签名与加密说明
1.0.3	2026-04-08	FIT 与 FIP 分章；RootFS、DATA 章节顺延
1.0.4	2026-04-08	目录中 FIP 章置于 FIT 章之前

1 声明



法律声明

本数据手册包含北京晶视智能科技有限公司（下称“晶视智能”）的保密信息。未经授权，禁止使用或披露本数据手册中包含的信息。如您未经授权披露全部或部分保密信息，导致晶视智能遭受任何损失或损害，您应对因之产生的损失/损害承担责任。

本文件内信息如有更改，恕不另行通知。晶视智能不对使用或依赖本文件所含信息承担任何责任。本数据手册和本文件所含的所有信息均按“原样”提供，无任何明示、暗示、法定或其他形式的保证。晶视智能特别声明未做任何适销性、非侵权性和特定用途适用性的默示保证，亦对本数据手册所使用、包含或提供的任何第三方的软件不提供任何保证；用户同意仅向该第三方寻求与此相关的任何保证索赔。此外，晶视智能亦不对任何其根据用户规格或符合特定标准或公开讨论而制作的可交付成果承担责任。

联系我们

地址 北京市海淀区丰豪东路 9 号院中关村集成电路设计园（ICPARK）1 号楼

深圳市宝安区福海街道展城社区会展湾云岸广场 T10 栋

电话 +86-10-57590723 +86-10-57590724

邮编 100094（北京）518100（深圳）

官方网站 <https://www.sophgo.com/>

技术论坛 <https://developer.sophgo.com/forum/index.html>

2 安全启动介绍

2.1 镜像结构

图 2.1 是 CV184x 的镜像结构。使用安全启动时，FIP.bin 镜像被签名并加密（可选），开机时由处理器进行校验。

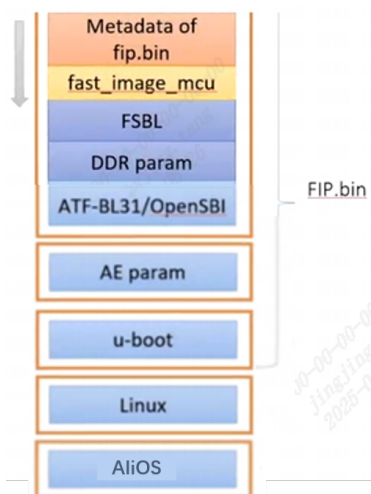
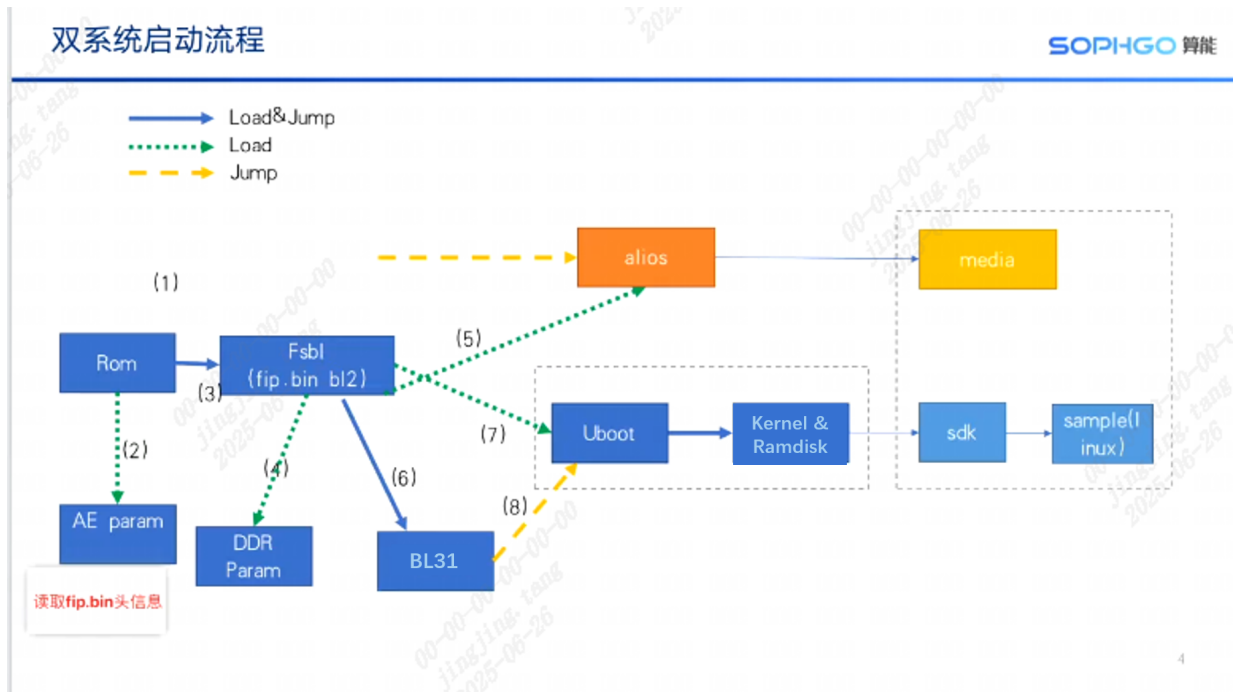


图 2.1: Layout of FIP.bin.

2.2 安全启动流程



3 安全镜像生成

本章概述 CV184x Spinor 安全启动场景下，**FIP**、**FIT** (内核/设备树)、**SquashFS RootFS**、**DATA (userdata)** 四层签名或加密与验签的关系。具体操作、命令与密钥生成请分章阅读：

- [FIP 签名与加密及 eFuse](#) —FIP 签名与加密、eFuse 与 FIP 分区烧录顺序
- [FIT 镜像安全启动](#) —FIT 验签与可选加密 (boot.spinor、U-Boot)
- [SquashFS RootFS 签名与内核验签](#) —RootFS (SquashFS) 签名与内核挂载前验签
- [DATA 分区 \(userdata\) 签名与验签](#) —DATA 分区 (JFFS2) 签名与启动时验签、挂载策略

3.1 四层保护对照

对象	典型镜像/分区	说明
FIP	fip.bin / fip_enc.bin	ROM/BL1 对 FIP 验签与解密（可选）；需 CONFIG_FSBLS_SECURE_BOOT_SUPPORT 等配置，详见 FIP 签名与加密及 eFuse
内核 (FIT)	boot.spinor 内 FIT	U-Boot 对 FIT 验签与可选解密；与 Enable kernel secure boot 、U-Boot Security 等选项相关，详见 FIT 镜像安全启动
RootFS	rootfs.spinor (SquashFS)	构建时追加签名块，内核挂载前验签，详见 SquashFS RootFS 签名与内核验签
userdata (DATA)	data.spinor (JFFS2)	构建时追加 trailer，启动脚本调用 verify_data_sig 通过后挂载，详见 DATA 分区 (userdata) 签名与验签

注解：注意事项

为避免量产密钥被窃，建议量产密钥单独保管，并在安全环境下进行签名与加密操作。

3.2 密钥与工具（速查）

FIP 侧常用私钥与密钥文件包括：`rsa_hash0.pem`、`bl_priv.pem`、`loader_ek.key`、`bl_ek.key`；FIT 与 RootFS、DATA 签名常与 `ramdisk/keys/test_reeos.key` 及 `linux_5.10/certs/cert.pem` 配合使用。`fipsign.py` 与 eFuse 见[FIP 签名与加密及 eFuse](#)；FIT 密钥与菜单项见[FIT 镜像安全启动](#)。FIP 签名工具路径示例：

```
python3 fsbl/plat/cvitek/cv184x/common/fipsign.py sign-enc --help
```

仅签名不加密时可使用 `sign` 子命令（参数见工具 `--help` 输出）。U-Boot 镜像生成仍参考《U-boot 移植应用开发指南》等文档得到原始 `fip.bin` 后再做签名或加密。

4 eFuse 烧写

与安全启动相关的 eFuse 取值、在 U-Boot 中的 `efusew` 烧写顺序，以及 eFuse 完成后的镜像烧录顺序，均见[FIP 签名与加密及 eFuse](#) 中下列小节：

- 获取 eFuse 要烧写的值（开发机）—`HASH0_PUBLIC`、`LOADER_EK` 的生成方式
- 烧写 eFuse（U-Boot 中，一次性、不可逆）—`efusew` 命令序列
- 烧录镜像（eFuse 烧写完成之后）—FIP 与其它分区、先后次序说明

具体 eFuse 位域定义与芯片约束仍以芯片手册及厂商文档为准。

注解：注意事项

eFuse 烧写为不可逆行为，执行前请确认镜像与密钥规划已核对无误。

5 FIP 签名与加密及 eFuse

本章说明在本仓库中对 **fip.bin** 做 **签名与加密**，并配合 **eFuse 烧录**，使 ROM/BL1 只执行受信任的 FIP。若还需 **boot.spinor** 内 FIT 的验签或加密，见[FIT 镜像安全启动](#)。

所有命令均在 **mars 根目录** 执行（U-Boot 内 eFuse 命令除外）；须先依次执行 `source build/envsetup_soc.sh` 与 `defconfig`，将后者参数换为实际板型名。

5.1 生成 FIP 用的密钥

只做一次，密钥妥善保存。

```
KEY_DIR="$(pwd)/my_fip_keys"
mkdir -p "$KEY_DIR"
openssl genrsa -out "${KEY_DIR}/rsa_hash0.pem" -F4 2048
openssl genrsa -out "${KEY_DIR}/bl_priv.pem" -F4 2048
head -c 16 /dev/random > "${KEY_DIR}/loader_ek.key"
head -c 16 /dev/random > "${KEY_DIR}/bl_ek.key"
chmod 600 "${KEY_DIR}"/*.pem "${KEY_DIR}"/*.key 2>/dev/null || true
```

得到：**rsa_hash0.pem**、**bl_priv.pem**、**loader_ek.key**、**bl_ek.key**。

5.2 打开 FSBL 安全启动并编译

- 在 `menuconfig` 中：**FIP setting** → 勾选 **Add secure boot support to FSBL** → 保存退出。

或直接在板级 `defconfig` 中加入：

```
CONFIG_FSBL_SECURE_BOOT_SUPPORT=y
```

执行 `build_all` 完成编译。完成后在 `install/` 下对应板型的 `soc_*` 子目录中可得到 **fip.bin** 等（具体以板级配置为准）。

注解： 若启用内核 FIT 安全启动，还需按[FIT 镜像安全启动](#) 配置内核与 U-Boot，并与本章 `build_all` 在同一套配置下完成。

5.3 对 FIP 做签名与加密

在 mars 根目录执行（将 KEY_DIR、FIP_IN、FIP_OUT 换成实际路径与板子输出目录）：

```
KEY_DIR="$(pwd)/my_fip_keys"
FIP_IN=install/soc_cv1842hp_wevb_0014a_spinor/fip.bin
FIP_OUT=install/soc_cv1842hp_wevb_0014a_spinor/fip_enc.bin
python3 fsbl/plat/cvitek/cv184x/common/fipsign.py sign-enc \
    --root-priv="${KEY_DIR}/rsa_hash0.pem" \
    --bl-priv="${KEY_DIR}/bl_priv.pem" \
    --ldr-ek="${KEY_DIR}/loader_ek.key" \
    --bl-ek="${KEY_DIR}/bl_ek.key" \
    "$FIP_IN" "$FIP_OUT"
```

得到形如 install/soc_XXX/fip_enc.bin 的输出（XXX 为板型目录名）。仅签名不加密时可改用 sign 子命令，参数见 fipsign.py --help。

5.4 获取 eFuse 要烧写的值（开发机）

5.4.1 HASH0_PUBLIC（64 位十六进制）

需安装 PyCryptodome：

```
python3 -c "
from Crypto.PublicKey import RSA
from Crypto.Hash import SHA256
key = RSA.import_key(open('my_fip_keys/rsa_hash0.pem').read())
n = key.n.to_bytes(256, 'big')
print(SHA256.new(n).digest().hex())
"
```

5.4.2 LOADER_EK（32 位十六进制）

```
xxd -p -c 256 my_fip_keys/loader_ek.key | tr -d '\n'
```

5.5 烧写 eFuse（U-Boot 中，一次性、不可逆）

将占位符替换为上一节得到的值后，在 U-Boot 中按顺序执行：

```
efusew HASH0_PUBLIC <64位十六进制>
efusew LOADER_EK <32位十六进制>
efusew LOCK_WRITE_LOADER_EK 01
efusew LOCK_WRITE_HASH0_PUBLIC 01
efusew SECUREBOOT 02
```

顺序以芯片与厂商文档为准；锁与 **SECUREBOOT** 烧写后即生效，无法撤销。

注解：注意事项

eFuse 烧写为不可逆行为，执行前请确认镜像与密钥规划已核对无误。

5.6 烧录镜像（eFuse 烧写完成之后）

- **FIP 分区**：将 **fip_enc.bin** 作为实际烧写内容，按原 **fip.bin** 用法并 **改名为 fip.bin** 后烧录。勿使用未签名的原始 **fip.bin** 或 **fip_spl.bin**。
- **其它分区**：与平时一致，例如 **boot.spinor** → **BOOT**，**rootfs.spinor** → **ROOTFS**，**data.spinor** → **DATA**，**yoc.bin** → **2nd** 等。
- **顺序**：先完成 eFuse 烧录，再烧录镜像及各分区。若先烧加密 FIP 再烧 eFuse，可能无法启动，需用未加密的 **fip.bin** 重烧 FIP 等方式恢复。

5.7 流程简表（FIP 侧）

阶段	步骤	内容
准备	0	（可选）FIT 安全启动见 FIT 镜像安全启动
密钥	1	生成 FIP 密钥目录 my_fip_keys
配置与编译	2	menuconfig 打开 FSBL Add secure boot support ； build_all
签名加密	3	fipsign.py sign-enc 得到 fip_enc.bin
eFuse 取值	4	开发机计算 HASH0_PUBLIC ；读取 LOADER_EK
eFuse 烧录	5	U-Boot 中 efusew (HASH0_PUBLIC 、 LOADER_EK 、 LOCK 、 SECUREBOOT)
分区烧录	6	以 fip_enc.bin 替换原 fip.bin 并仍命名为 fip.bin 后烧 FIP；其它分区照旧

5.8 FIP 密钥一览（my_fip_keys/）

文件	用途
rsa_hash0.pem	签署 FSBL (BL2) 的 RSA 私钥；其公钥哈希烧入 eFuse HASH0_PUBLIC
bl_priv.pem	签署 Monitor/U-Boot 的 RSA 私钥
loader_ek.key	加密 FSBL 的 AES 密钥 (16 字节)；烧入 eFuse LOADER_EK
bl_ek.key	加密 Monitor/U-Boot 的 AES 密钥 (16 字节)

5.9 与 FIT 的关系

- **FIT**: 保护 **boot.spinor**, 由 U-Boot 验签或解密, 见[FIT 镜像安全启动](#)。
- **FIP**: 保护 **fip.bin**, 由 ROM/BL1 验签与解密。

两者可同时使用: 先完成 FIT 侧打包与烧录策略, 再对 **fip.bin** 做 sign-enc; 烧录时用 **fip_enc.bin** 替换并改名为 **fip.bin** 写 FIP 分区, BOOT 分区写已签名或加密的 **boot.spinor**。

6 FIT 镜像安全启动

本章说明 `boot.spinor` 中 FIT 镜像的验签与可选加密：由 **U-Boot** 在加载内核与设备树前完成校验或解密。FIT 与 **FIP** 在 ROM/BL1 侧的职责相互独立；产线常见顺序是先完成 FIT 侧配置与打包，再对 `fip.bin` 做签名与加密，详见[FIP 签名与加密](#)及 [eFuse](#)。

除进入 `ramdisk/keys` 目录后的命令外，其余命令均在 **mars 根目录** 执行；须先依次执行 `source build/envsetup_soc.sh` 与 `defconfig`，将后者参数换为实际板型名。

6.1 生成 FIT 验签与加密用的密钥与材料

只做一次，与 FIP 密钥分开保存。FIT 的 RSA 验签使用 **私钥**与 **公钥证书**；若开启 FIT 加密，还需要 **AES-256 密钥**与 **IV**。

在 **mars 根目录**下进入 `ramdisk/keys` 后执行（与 `ramdisk` 侧打包脚本约定的路径一致）：

```
cd ramdisk/keys
openssl genpkey -algorithm RSA -out test_reeos.key
openssl req -new -x509 -key test_reeos.key -out test_reeos.crt
openssl rand -out test_reeos_aes256.bin 32
openssl rand -out test_reeos_iv.bin 16
```

得到：`test_reeos.key`、`test_reeos.crt`；若启用 FIT 加密，还有 `test_reeos_aes256.bin`、`test_reeos_iv.bin`。ITS、打包脚本、U-Boot 环境变量中的路径须与上述文件名及目录一致。

6.2 打开配置（内核与 U-Boot）

- 执行 `source build/envsetup_soc.sh`、`defconfig`（实际板型名）`后运行 ``menuconfig`，在 **Kernel options** 下勾选 **Enable kernel secure boot**，保存退出。
- （可选）在 `menuconfig` 中勾选菜单项 **Support FIT image encryption/decryption**。勾选后对 FIT 做加密与验签；不勾选则仅验签、不加密。
- 执行 `menuconfig_uboot`，进入 **Security support**，勾选 **Add secure boot support to kernel**，并确保子项已启用：**Enable RSA support**、**Enable hash command**、**Enable libcrypto support** 等，保存退出。

6.3 编译与产物

执行 `build_all` 完成编译。完成后在 `install/` 下对应板型的 `soc_*` 子目录中可得到 **boot.spinor** 等产物（具体以板级配置为准）。烧录时按现有流程将带签名或加密的 **boot.spinor** 写入 BOOT 分区。

6.4 FIT 相关密钥一览 (ramdisk/keys/)

文件	用途
<code>test_reeos.key</code>	FIT RSA 私钥（签名）
<code>test_reeos.crt</code>	FIT 公钥证书
<code>test_reeos_aes256.bin</code>	FIT AES-256 密钥（32 字节，仅当启用 FIT 加密）
<code>test_reeos_iv.bin</code>	FIT 加密 IV（16 字节，仅当启用 FIT 加密）

6.5 流程简表 (FIT 侧)

步骤	内容
1	在 <code>ramdisk/keys</code> 下生成 FIT 密钥与证书；可选 AES/IV
2	<code>menuconfig</code> 打开 Enable kernel secure boot；可选 Support FIT image encryption/decryption
3	<code>menuconfig_uboot</code> 打开 Add secure boot support to kernel 及所需子项
4	执行 <code>build_all</code> ，取 <code>install</code> 下 <code>boot.spinor</code> 等产物并按原流程烧录 BOOT 分区

6.6 与 FIP 的配合

- **FIT**：保护 **boot.spinor** 中的 kernel、FDT 等，由 U-Boot 执行。
- **FIP**：保护 **fip.bin** 镜像，含 U-Boot 与 BL2 等，由 ROM/BL1 验签与解密，见[FIP 签名与加密及 eFuse](#)。

完整安全启动链路中，通常在完成本章配置并得到 **boot.spinor** 后，再按[FIP 签名与加密及 eFuse](#)生成 **fip_enc.bin**、烧 eFuse 并烧录 FIP 分区。

7 SquashFS RootFS 签名与内核验签

7.1 整体目标

- **目的**：保证设备上 mtdblock6 等设备节点对应的 SquashFS rootfs（分区号以板级为准）未被篡改，内核在挂载前对其做签名验证，失败则拒绝挂载。
- **实现方式**：构建时对 mksquashfs 产出的镜像做 **SHA256** 与 **裸 RSA** 签名，并追加自定义签名块；内核挂载前按同一规则验签。

7.2 使用与排查建议

日常使用可归纳为：

1. menuconfig->Kernel options->Enable kernel secure boot。
2. 执行 **build_all**。
3. 烧录生成的镜像，将 install/.../rootfs.spinor（路径随板型变化）写入 ROOTFS 分区。

可选：用 check_rootfs_sig.sh 确认 rootfs 已签名 (sig_len>0)。更换证书时需同时更新 linux_5.10/certs/cert.pem 与构建端 -c / -k 参数。

- **验签成功**：日志出现 VFS: Mounted root (squashfs filesystem) readonly on device 31:6.，随后 /sbin/init 正常启动。
- **error -126**：若日志出现 SquashFS raw signature requires cert with 'SquashFS Root Signing'，则 get_squashfs_x509_key() 为 NULL。通过 fallback 修改后，单证书（如 “Internet Widgits Pty Ltd”）会被用作 SquashFS key，此错误可消除。
- **error -129**（仅旧 PKCS#7 路径）：若日志出现 Message digest doesn't match 或提示 error -129，多为设备上前 bytes_used 字节与签名时 payload 不一致；当前已用自定义 raw 格式，无此 PKCS#7 摘要比较。

7.3 相关文件

用途	路径（相对仓库根）
私钥（签名）	ramdisk/keys/test_reeos.key
证书（验签）	linux_5.10/certs/cert.pem
签名脚本	build/scripts/append_squashfs_sig.py
检查脚本	build/scripts/check_rootfs_sig.sh
内核验签	linux_5.10/fs/squashfs/verification.c
证书加载	linux_5.10/certs/system_keyring.c
SquashFS key 接口	linux_5.10/security/keys/key.c

证书通过 CONFIG_SYSTEM_TRUSTED_KEYS 编入内核；构建时用上述私钥对 rootfs 签名。

8 DATA 分区 (userdata) 签名与验签

8.1 概述

- **目的**: DATA 分区 (JFFS2) 在烧录前做 **SHA256 + RSA** 签名, 设备启动挂载前先 **验签**; 通过则挂载 JFFS2, 失败或未签名则挂载 tmpfs, 避免使用被篡改或未授权数据。
- **算法**: 对 payload 做 SHA256, 再对 32 字节哈希做 PKCS#1 v1.5 裸 RSA 签名 (2048-bit); 验签使用公钥的模数 N 与指数 e, 与签名私钥成对。
- **密钥**: 签名使用 ramdisk/keys/test_reeos.key; 验签公钥由 gen_data_sig_pubkey_h.py 从该私钥 (或对应证书) 提取并生成 C 头文件; 设备端无 OpenSSL, 使用 mbedTLS 做 RSA 验签。

8.2 使用与验证方式

1. menuconfig->Kernel options->Enable kernel secure boot。
2. 执行 **build_all**, 或按需 **make data**、**make rootfs**: 前者生成并签名 **data.spinor**, 后者将 **verify_data_sig** 等装入 rootfs。
3. 烧录时将 **install/.../data.spinor** 整文件写入 DATA (userdata) 分区起始位置; 镜像路径随板型变化, 具体分区名与烧录工具以板级为准。
4. 设备启动后, 对 DATA 分区执行 **verify_data_sig**, 验签通过则挂载 JFFS2, 失败或未签名则挂载 tmpfs。
5. 在设备上执行 **mount | grep /mnt/data**, 若输出中出现 jffs2, 即表示 DATA 验签成功并已挂载 JFFS2; 若为 tmpfs, 则表示验签未通过或未签名。

8.3 相关文件一览

文件	作用
build/scripts/append_image_sig.py	对 data 镜像做 SHA256 + RSA 签名并追加 trailer
build/scripts/gen_data_sig_pubkey_h.py	从密钥或证书生成 verify_data_sig_key.h
build/scripts/verify_data_sig.c	设备端与本机验签程序 (mbedTLS)
build/scripts/verify_data_sig_key.h	公钥 N/e (由 gen_data_sig_pubkey_h.py 生成)
build/tools/common/image_tool/mkjffs2.py	制作 JFFS2 data.spinor 并预留 trailer 空间
build/tools/common/image_tool/create_automount.py	生成挂载脚本 (含 DATA 验签分支)
build/Makefile	编译 verify_data_sig / verify_data_sig_host, 构建 host/ARM 用 mbedTLS