Android Widevine
开发指南

版本号: 1.0
发布日期: 2024.11.13

版本历史

版本号	日期	制/修订人	内容描述
1.0	2024.11.13	AWA2093	初始版本文档

目　录

1 概述 **1**

1.1 读者对象 1

1.2 Widevine 简介 1

1.2.1 Widevine 整体框架 1

1.2.2 工作原理 3

1.2.3 安全等级 6

2 开发流程 **7**

2.1 Widevine L3 开发流程 7

2.2 Widevine L1 开发流程 9

2.2.1 专业术语 10

2.2.2 开发步骤 10

2.2.3 环境配置 10

2.2.4 widevine provision 工作流对比 11

2.2.4.1 工作流 - provisioning 2.0 11

2.2.4.2 设备工作流 - provisioning 4.0（新） 12

2.2.5 provisioning 4.0 远程密钥处理流程介绍 13

2.2.6 谷歌审批网站注册 widevine L1 步骤 14

2.2.7 使用谷歌原生工具提取 widevine CSR json 文件操作步骤 16

2.2.8 使用全志 RKP 工具提取 widevine CSR json 文件操作步骤 16

2.2.9 使用全志 RKP 工具上传 widevine CSR json 文件操作步骤 22

2.2.10 注意事项 22

3 测试 **23**

3.1 provisioning 流程测试 23

3.1.1 provisioning 4.0 原理 23

3.1.2 provisioning 4.0 测试 25

3.2 GMS 测试 26

3.2.1 GTS 26

3.2.2 VTS 27

3.3 Exoplayer 测试 27

3.4 disney plus 测试 27

3.5 prime video 测试 27

插　图

图 1-1　Android 11 之前的 DRM 框架 2
图 1-2　从 Android 11 开始的 DRM 框架 3
图 1-3　Android 11 之前的 DRM 硬件抽象层 4
图 1-4　从 Android 11 开始的 DRM 硬件抽象层 5
图 1-5　Widevine 插件图 6
图 2-1　provisioning 4.0 交互模型 13
图 2-2　审批设置 1 14
图 2-3　审批设置 2 14
图 2-4　审批设置 3 15
图 2-5　审批设置完成界面 15
图 2-6　DragonSN 设置 1 17
图 2-7　DragonSN 设置 2 18
图 2-8　DragonSN 设置 3 18
图 2-9　DragonSN 设置 4 19
图 2-10　DragonSN 设置 5 19
图 2-11　DragonSN 设置 6 20
图 2-12　DragonSN 提取成功界面 21
图 2-13　提取出来的 csr json 文件 21
图 2-14　包含 keymint 和 widevine 的 CSR 信息 22
图 2-15　只包含 widevine 的 CSR 信息 22
图 3-1　密钥校验流程 24
图 3-2　2 步配置流程 25
图 3-3　provision 成功的日志打印 26
图 3-4　provision 失败的日志打印 26

1 概述

本文档主要介绍全志 Widevine 的工作原理以及开发过程及注意事项。

1.1 读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

1.2 Widevine 简介

Widevine 是美国的一家专门提供流媒体数字版权保护 (DRM) 技术的公司，该公司的 DRM 技术被广泛地应用于数字流媒体领域，例如在线视频、数字电视等等。2010 年 9 月，谷歌收购了此公司，意图拓展自己的数字流媒体电影服务以及获得其 DRM 保护技术，谷歌在 Android 上搭载了 Widevine 的 DRM 技术，已经成为了 GMS(Google Mobile Service) 中必备的内容。

1.2.1 Widevine 整体框架

Widevine 是 DRM 的一种插件，在 Android 中应用，使用的架构是固定的 DRM 架构。

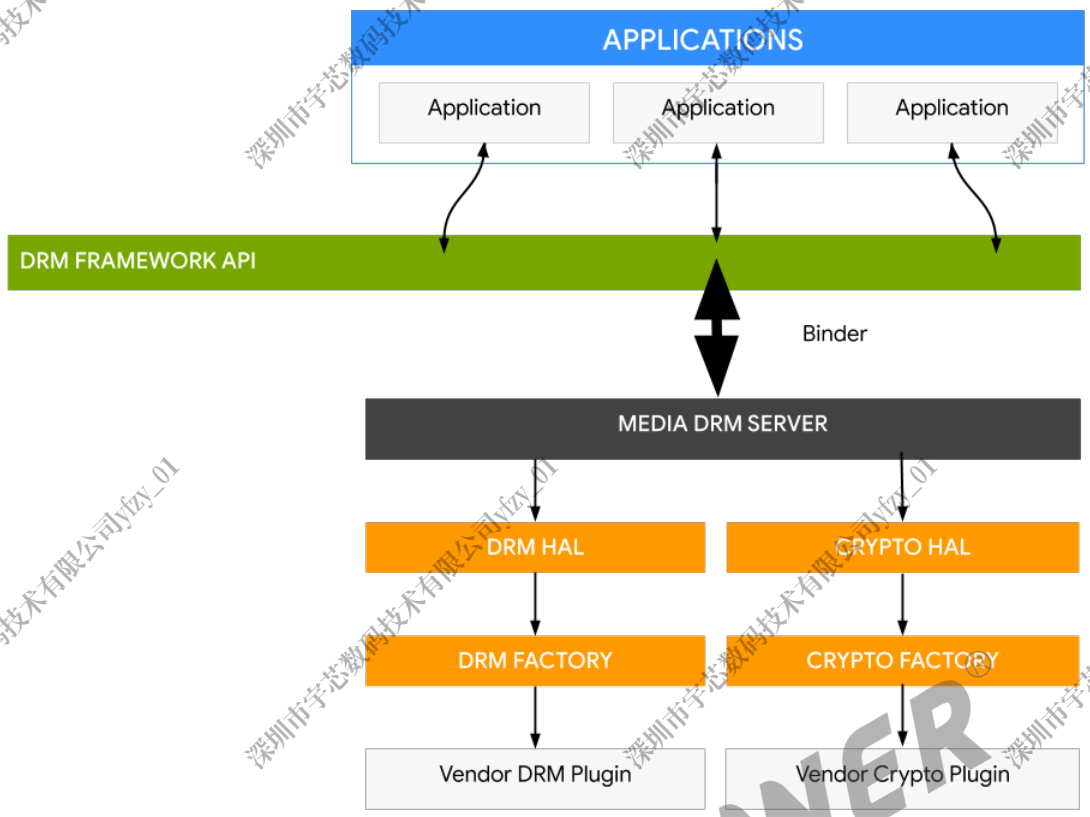

图 1-1: Android 11 之前的 DRM 框架

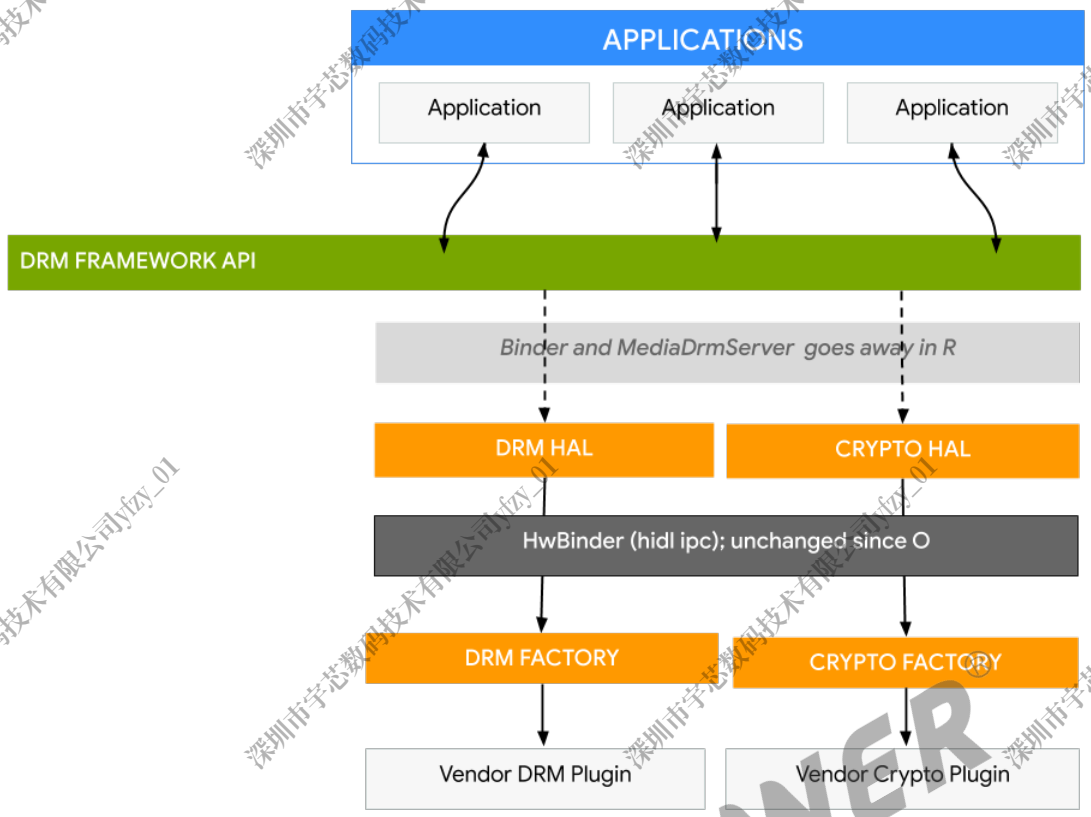

图 1-2: 从 Android 11 开始的 DRM 框架

DRM 框架与实现无关，可在方案特定的 DRM 插件中提取特定 DRM 方案实现的详情。DRM 框架包括可执行以下操作的简单 API: 处理复杂的 DRM 操作，获得许可，配置设备，将 DRM 内容与其许可相关联，以及最终解密 DRM 内容。

Android DRM 是在以下两个架构层中实现的:

• DRM framework API: 通过 Android 应用框架提供给应用。

• 本机代码 DRM 框架：为 DRM 插件（代理）提供接口，以便处理各种 DRM 方案的版权管理和解密工作。

1.2.2 工作原理

Android 平台提供了一个可扩展的 DRM 框架，支持应用根据与受版权保护的内容关联的许可限制条件来管理这些内容。DRM 框架支持多种 DRM 方案；设备具体支持哪些 DRM 方案由设备制造商决定。DRM 框架为应用开发者提供了一个统一接口，并隐藏了 DRM 操作的复杂性。DRM 框架为受保护和不受保护的内容提供了一致的操作模式。DRM 方案可以定义复杂的许可元数据使用模型。DRM 框架提供了 DRM 内容与许可之间的关联，并处理权限管理。这样可以将媒体播放器从受 DRM 保护或不受保护的内容中提取出来。

DRM 插件会与 Android DRM 框架集成在一起，并可使用受硬件支持的保护功能来确保付费内容和用户凭据的安全。

DRM 插件提供的内容保护功能取决于底层硬件平台的安全和内容保护功能。设备的硬件功能应包括硬件安全启动，可建立加密密钥的安全和保护功能的信任链。设备的内容保护功能应包括设备内加密帧的保护和通过可信输出保护机制实现的内容保护。并非所有硬件平台都支持上述所有的安全和内容保护功能。安全功能绝不会在堆栈的单个位置实现，而是依赖于硬件、软件和服务的集成。将硬件安全功能、可信启动机制以及用于处理安全功能的隔离安全操作系统组合使用是保障安全设备的关键。

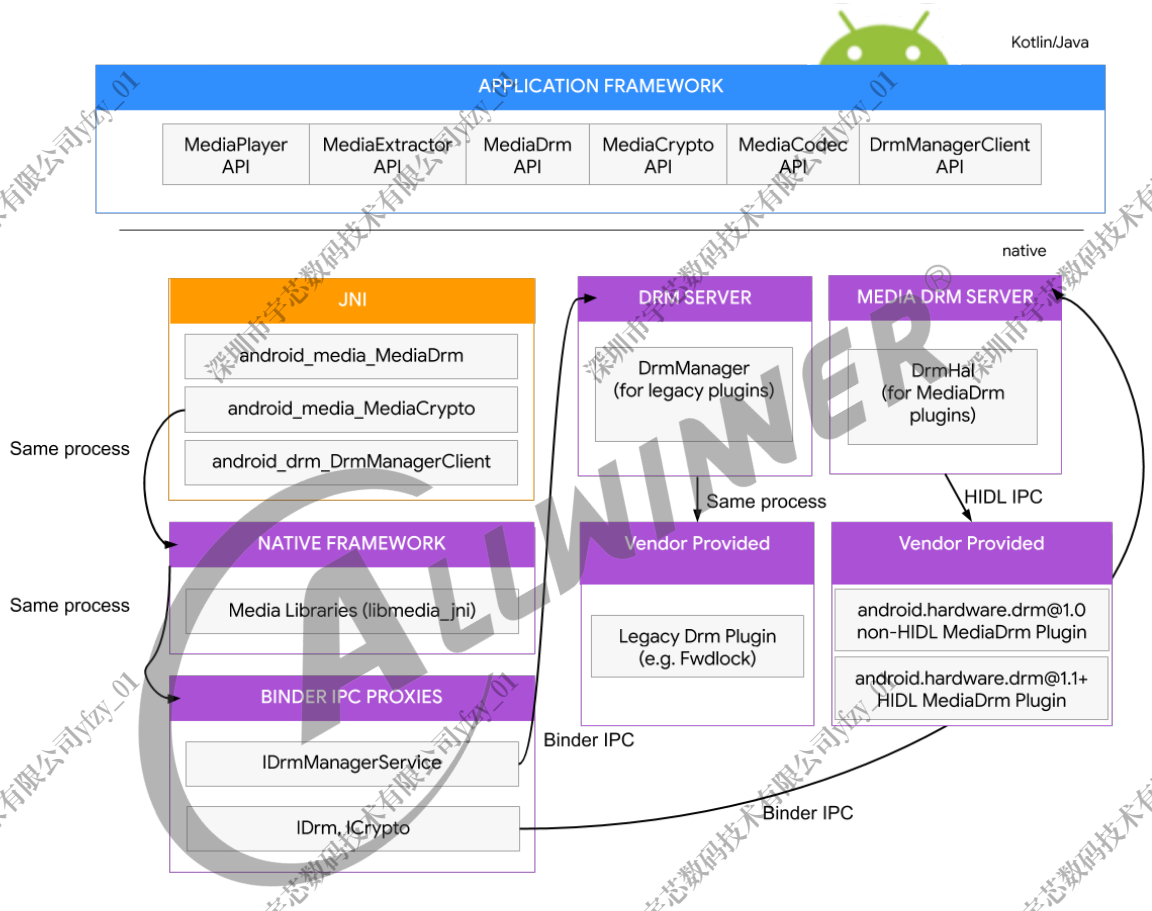

图 1-3: Android 11 之前的 DRM 硬件抽象层

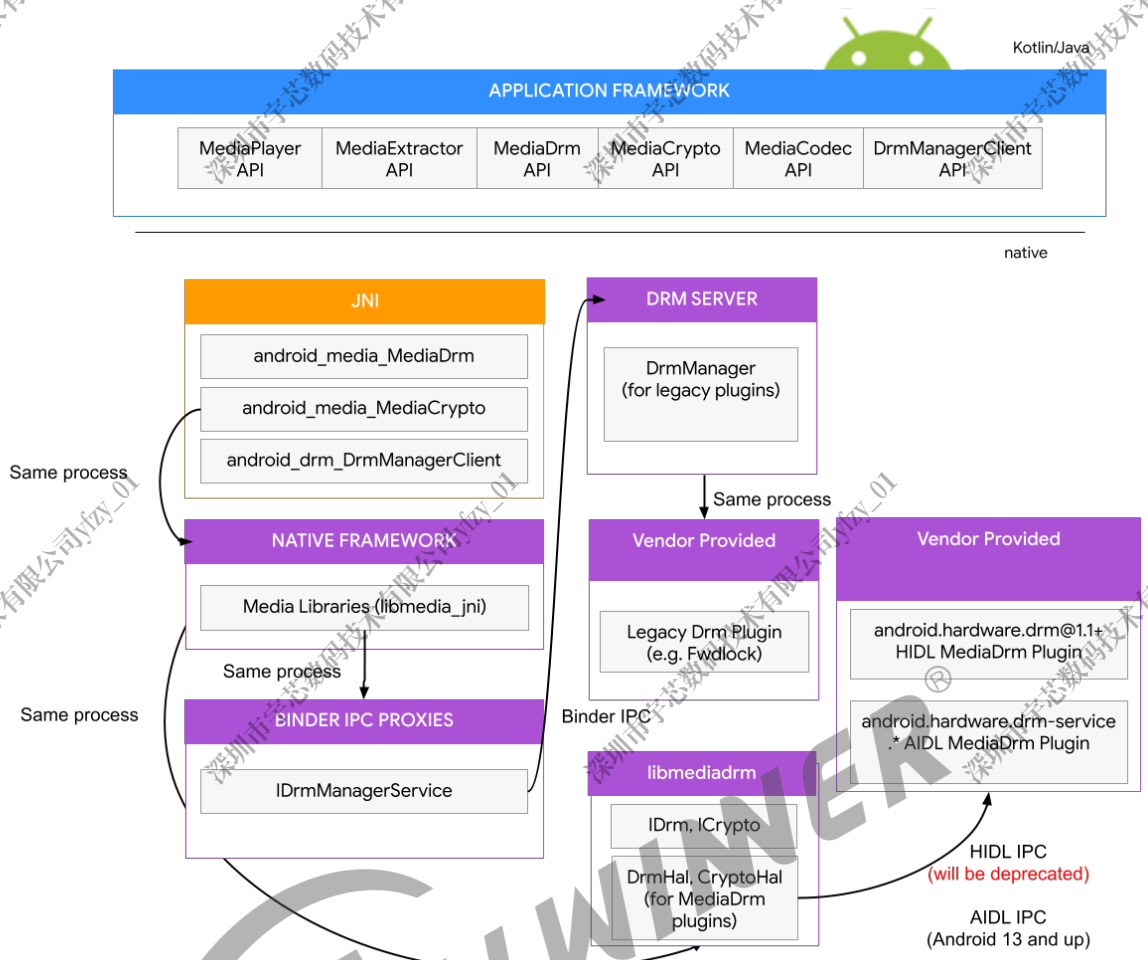

图 1-4: 从 Android 11 开始的 DRM 硬件抽象层

对于 Widevine 插件，在 Android 上同样是基于这个架构来进行。

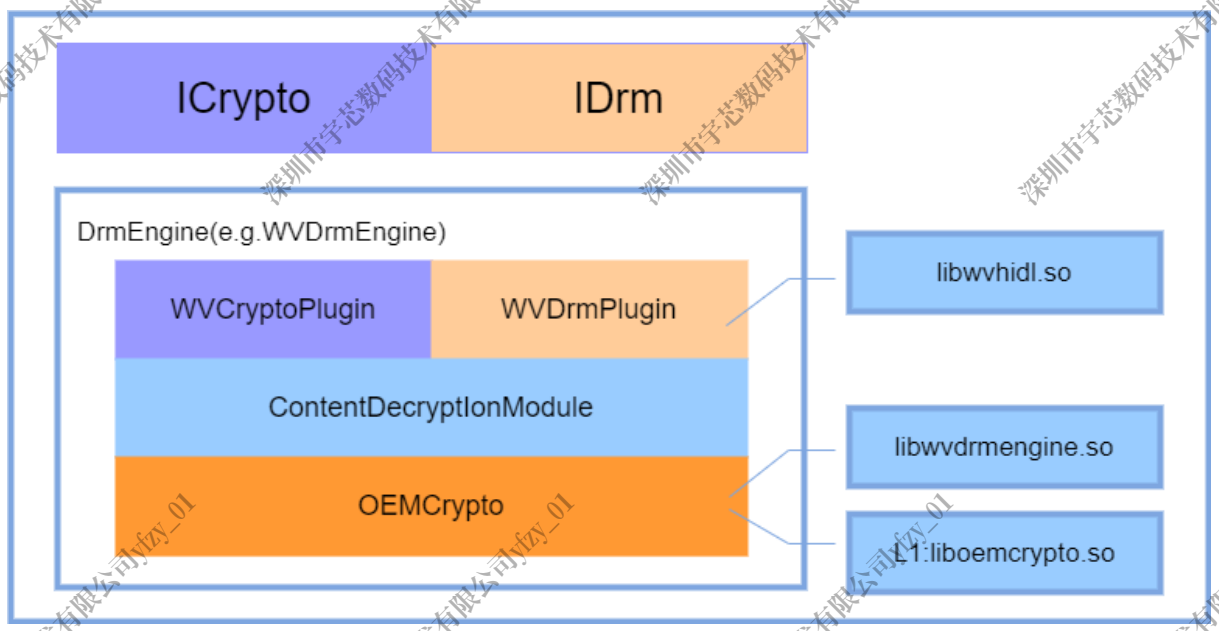

图 1-5: Widevine 插件图

步骤 1：在播放前，WVDrmPlugin 向 DRM Server 申请 license，并获得认证通过。

步骤 2：媒体服务（MediaCodec) 向内容服务器下载加密内容，经过 MediaExtractor 解析将加密内容解析出来，然后调用 WVCryptoPlugin 进行解密，最后送码流到解码器进行解码播放。

1.2.3 安全等级

内容保护需要依赖于平台的安全能力，一个平台的安全能力需要安全硬件的支持，但并非所有的设备都具备该技术所需的硬件设备，故 Widevine 分为了三个安全等级。

表 1-1: 安全等级表

安全等级	安全启动	widevine keybox 装配	安全硬件 TRUSTZONE	KEYBOX 及视频密钥处理	硬件视频路径
Level 1	是	工厂安装	是	密钥不以明文暴露给 CPU	视频流通过硬件保护，输出在 TEE 中
Level 2	是	工厂安装	是	密钥不以明文暴露给 CPU	解码器直接获得明文视频流
Level 3	否	软件安装	否	密钥以明文暴露给 CPU	解码器直接获得明文视频流

2 开发流程

A733 64 位默认支持 Widevine L1，A733 32 位默认支持 Widevine L3。

2.1 Widevine L3 开发流程

Widevine L3 不需要基于设备生成的密钥认证，故 OEM 厂商不需要向 Google 上传 widevine CSR。使用全志的 SDK，一般默认提供 Widevine L3 的功能支持。

全志的 SDK 中，Widevine 相关的库文件放在路径：$android/hardware/aw/widevine 下。

```
.
├── Android.bp
├── android.hardware.drm-service-lazy.widevine
├── android.hardware.drm-service-lazy.widevine.rc
├── android.hardware.drm-service.widevine
├── android.hardware.drm-service.widevine.rc
├── apex
│   ├── allowlist_com.google.android.widevine.xml
│   ├── Android.bp
│   ├── apex_manifest.json
│   ├── com.google.android.widevine.avbpubkey
│   ├── com.google.android.widevine.pem
│   ├── com.google.android.widevine.pk8
│   ├── com.google.android.widevine.rc
│   ├── com.google.android.widevine.x509.pem
│   ├── com.google.android.widevine.xml
│   ├── device
│   │   ├── AndroidProducts.mk
│   │   ├── build.sh
│   │   ├── device.mk
│   │   ├── linker.config.json
│   │   └── widevine_generic.mk
│   ├── file_contexts
│   ├── nonupdatable
│   │   ├── Android.bp
│   │   ├── com.google.android.widevine.nonupdatable.pk8
│   │   ├── com.google.android.widevine.nonupdatable.x509.pem
│   │   └── lazy
│   │       ├── Android.bp
│   │       ├── apex_manifest.json
│   │       ├── com.google.android.widevine.lazy.rc
│   │       └── file_contexts
│   ├── prebuilt
│   │   ├── Android.bp
│   │   ├── com.google.android.widevine.lazy.apks
│   │   ├── com.google.android.widevine.nonupdatable.apks
│   │   └── release
│   │       ├── com.google.android.widevine.lazy-11868851.apks
```

```
│   │   └── com.google.android.widevine.nonupdatable-11868851.apks
│   └── proto
│       ├── Android.bp
│       ├── apex_manifest_minimal.proto
│       ├── widevine_apex_info.cpp
│       └── widevine_apex_info.h
├── CleanSpec.mk
├── demo
│   ├── ExoPlayerDemo.apk
│   └── readme.txt
├── lib32
│   ├── libvtswidevine.so
│   ├── libwvaidl.so
│   └── secure
│       └── liboemcrypto.so
├── lib64
│   ├── libvtswidevine.so
│   ├── libwvaidl.so
│   └── secure
│       └── liboemcrypto.so
├── manifest_android.hardware.drm-service.widevine.xml
├── nativetest
│   ├── base64_test
│   ├── buffer_reader_test
│   ├── cdm_coverage_test
│   ├── cdm_engine_metrics_decorator_unittest
│   ├── cdm_engine_test
│   ├── cdm_extended_duration_test
│   ├── cdm_feature_test
│   ├── cdm_random_unittest
│   ├── cdm_session_unittest
│   ├── cdm_usage_table_unittest
│   ├── certificate_provisioning_unittest
│   ├── core_integration_test
│   ├── counter_metric_unittest
│   ├── crypto_session_unittest
│   ├── device_files_unittest
│   ├── distribution_unittest
│   ├── duration_use_case_test
│   ├── event_metric_unittest
│   ├── file_store_unittest
│   ├── file_utils_unittest
│   ├── generic_crypto_unittest
│   ├── hal_metrics_adapter_unittest
│   ├── http_socket_test
│   ├── initialization_data_unittest
│   ├── keybox_ota_test
│   ├── libwvdrmdrmplugin_hal_test
│   ├── libwvdrmengine_hal_test
│   ├── libwvdrmmediacrypto_hal_test
│   ├── license_keys_unittest
│   ├── license_unittest
│   ├── metrics_collections_unittest
│   ├── oemcrypto_test
│   ├── okp_fallback_policy_test
│   ├── ota_keybox_provisioner_test
│   ├── policy_engine_constraints_unittest
│   ├── policy_engine_unittest
│   ├── policy_integration_test
│   ├── reboot_test
```

```
├── request_license_test
├── rw_lock_test
├── service_certificate_unittest
├── system_id_extractor_unittest
├── timer_unittest
├── value_metric_unittest
├── vendor
│   ├── odk_test
│   │   └── odk_test
│   └── wv_plugin_test
│       └── wv_plugin_test
└── wv_cdm_metrics_test
└── run_all_unit_tests.sh
```

在使用 Widevine L3 时，需要检查设备的配置选项。

配置路径 1：

A733:

$android14_grf_sdk/device/softwinner/jupiter/common/secure/config.mk。

```
# widevine config
ifeq ($(TARGET_ARCH),arm)
BOARD_WIDEVINE_OEMCRYPTO_LEVEL := 3  //支持Widevine L3
else ifeq ($(TARGET_ARCH),arm64)
BOARD_WIDEVINE_OEMCRYPTO_LEVEL := 1  //支持Widevine L1
endif
```

需要过 GMS 认证的设备，若需要支持 widevine，则必须要使用 apex，配置如下：

```
#widevine config use apex
BOARD_WIDEVINE_APEX_ENABLED := true
```

配置路径 2：

A733:

$android14_grf_sdk/device/softwinner/jupiter/common/media/config.mk。

```
ifeq ($(TARGET_ARCH),arm)
WIDEVINE_OEMCRYPTO_LEVEL := 3   //支持Widevine L3
else ifeq ($(TARGET_ARCH),arm64)
WIDEVINE_OEMCRYPTO_LEVEL := 1   //支持Widevine L1
endif
```

2.2 Widevine L1 开发流程

provisioning 4.0 是新一代 Widevine 设备配置。此配置方案与可信计算组设计的模型密切相关。从意图和设计上看，Widevine 配置 4.0 模型与 Android 的全新远程认证密钥配置流程密切相关，也就是谷歌为了进一步加强防破解，从 Android14 开始，widevine L1 的设备需要强制使用 provision4 远程密钥的方式，采用此方式时，密钥会保存在谷歌的远程数据库服务器上，而不是像之前的 keybox 那样，将密钥和设备信息保存烧录在本机设备，这种做法将会大大提高设备的防破解能

力。

Widevine L1 需要针对每一部终端设备提取并上传 widevine CSR 文件到谷歌远程服务器，并且需要安全硬件支持，流程较复杂。

2.2.1 专业术语

- GMS: (Google Mobile Service) 谷歌移动服务。
- RKP: (Remote Key Provisioning) 远程密钥配置。新一代 Google 信用链初始化机制。
- CSR: (Certificate Signing Request) 数字签名请求，从终端设备提取出来的 CSR 是一个 json 类型的文件。
- BCC: (BootCertificateChain) 根证书链，从终端设备生成，每个设备的 BCC 是唯一的。

2.2.2 开发步骤

步骤 1：OEM 厂商向谷歌签订协议。

步骤 2：OEM 厂商获得全志的 SDK 及安全相关的 patch。

步骤 3：OEM 厂商与全志集成开发。

步骤 4：OEM 厂商使用全志 RKP 提取工具，批量提取每个机器唯一的 widevine CSR json 文件。

步骤 5：OEM 厂商使用全志 RKP 上传工具，将提取到的 widevine CSR json 文件批量上传到谷歌远程服务器。

步骤 6：完成测试，发布软件。

步骤 7：OEM 产线生产和出货。

2.2.3 环境配置

1. Widevine L1 配置

配置路径 1：

A733:

$android14_grf_sdk/device/softwinner/jupiter/common/secure/config.mk。

```
# widevine config
ifeq ($(TARGET_ARCH),arm)
BOARD_WIDEVINE_OEMCRYPTO_LEVEL := 3 //支持Widevine L3
else ifeq ($(TARGET_ARCH),arm64)
```

```
BOARD_WIDEVINE_OEMCRYPTO_LEVEL := 1 //支持Widevine L1
endif
```

需要过 GMS 认证的设备，若需要支持 widevine，则必须要使用 apex，配置如下：

```
#widevine config use apex
BOARD_WIDEVINE_APEX_ENABLED := true
```

配置路径 2：

A733:

$android14_grf_sdk/device/softwinner/jupiter/common/media/config.mk。

```
ifeq ($(TARGET_ARCH),arm)
WIDEVINE_OEMCRYPTO_LEVEL := 3   //支持Widevine L3
else ifeq ($(TARGET_ARCH),arm64)
WIDEVINE_OEMCRYPTO_LEVEL := 1   //支持Widevine L1
endif
```

Widevine L1 需要安全固件，安全启动支持。

2. 安全 buffer 大小设置，文件中默认配置预留内存 150M（64 位默认打开，32 位需手动打开），如果要播放大分辨率的视频，需要将此 buffer 增大。

A733 pro3 机型 64 位路径：

$android14_grf_sdk/longan/device/config/chips/a733/configs/pro3/sys_config.fex。

文件中具体配置如下。

```
[secure]
dram_region_mbytes = 150
```

⚠ 注意

注意：支持 1080P 至少需要占用 120M 预留内存，若需要 L1 功能，建议使用内存为 3GB 以上的方案。

3. mediacodec 配置，可以确认 media_codecs_allwinner_video.xml（路径：$android/softwinner/jupiter/common/media/codec/xml_for_l1) 是否有配置 secure 解码器，默认已配置。
4. 镜像打包与烧写，需要安全固件（略）。

2.2.4 widevine provision 工作流对比

2.2.4.1 工作流 - provisioning 2.0

下表介绍了使用 keybox（provisioning 2.0）为 Widevine 集成设备的流程。

表 2-1: keybox 配置模型

#	说明	执行者
1	SOC 开发 Widevine OEMCrypto TA。	SOC
2	OEM 向 Widevine（或通过 APFE 向 Android 注册）设备。	OEM
3	OEM 从 Widevine 或 Android（通过 APFE）请求 Widevine 密钥箱。	OEM
4	SOC 为 OEM 提供了用于按设备安装密钥箱的工具。	SOC
5	OEM 制造设备。	OEM

2.2.4.2 设备工作流 - provisioning 4.0（新）

下表介绍了 A733 当前使用的 provisioning 4.0 实现 Widevine 的新设备集成流程。

⚠ 注意

注意：需要 OEMCrypto v18 或更高版本。

表 2-2: provisioning 4.0 配置模型

#	说明	执行者
1	SOC 开发 Widevine OEMCrypto TA。	SOC
2	SoC 会通过网页界面向 Widevine 注册芯片组。	SOC
3	OEM 向 Widevine（或通过 APFE 向 Android 注册）设备。	OEM
4	OEM 使用每个芯片上现有的唯一设备密钥 (UDS) 创建信任根。	OEM
5	OEM 制造设备。	OEM

2.2.5 provisioning 4.0 远程密钥处理流程介绍

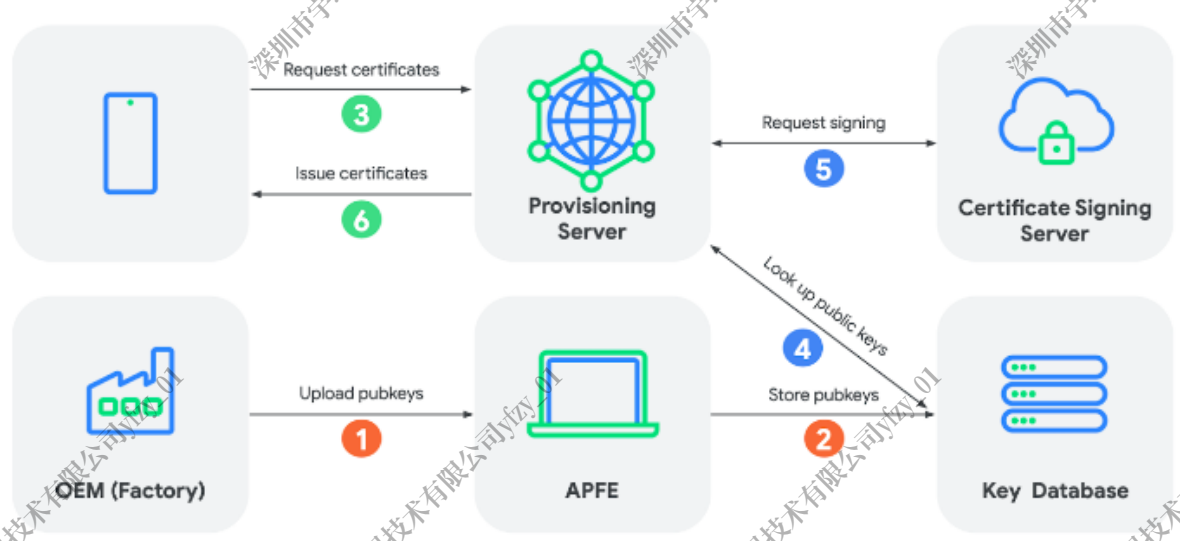

图 2-1: provisioning 4.0 交互模型

1. 上传设备信息和 BCC。

1.1. 设备信息包括设备制造商、型号、芯片体系结构、设备名称、产品名称和构建信息。

1.2. BCC，即启动证书链，是设备的唯一标识和信任根。

1.3. Android 团队将为 Android 设备提供工具，Widevine 将为 living room 设备提供工具，以从每个设备中提取信息。

1.4. 最后，需要通过 Android 或 Widevine API 将信息上传到 Google 数据库。

2. Android/Widevine 服务器将上传的数据保存到数据库中;
3. 当设备在线并且需要进行配置时，应用程序需要获取并发送配置请求到 Widevine 服务器;

3.1. 在此步骤中，BCC 和设备信息将嵌入到请求中。

3.2. BCC 将使用服务证书的隐私密钥进行加密。

3.3. 请求必须发送到 Widevine 服务器。

4. Widevine 服务器检查 BCC 并查询数据库以查看其公钥是否存在。如果找到匹配项，则创建一个 OEM 叶证书 (OEM leaf certificate)。
5. Widevine provision 服务器向证书签名服务器获取 DRM 证书;
6. 返回第二阶段的配置以获取 DRM 证书

说明

注意：如上图步骤 1 中 OEM 执行 Upload pubkey 的动作，其实就是上传 CSR 文件的操作，因为 CSR 文件中包含了 BCC 的 pubkey 信息，所以才有了如下的提取并上传 CSR 文件的步骤。

2.2.6 谷歌审批网站注册 widevine L1 步骤

只要用到 widevine L1，keybox 或者 provision 4.0 两种配置模式都需要配置（2 选 1）。其中 provisoning 4.0 机制下，开发 widevine L1 前，需要前往谷歌审批网站注册 widevine L1，这样从终端设备提取并上传的 CSR 文件才会被谷歌服务器接收到并保存到其数据库里面。下面介绍 L1 下 provisoning 4.0 的具体配置。

1. 使用谷歌伙伴账号，登录审批网站：https://partner.android.com/approvals/
2. 按照从 1 到 6 的步骤，逐步点击打开页面，选择”edit“选项：

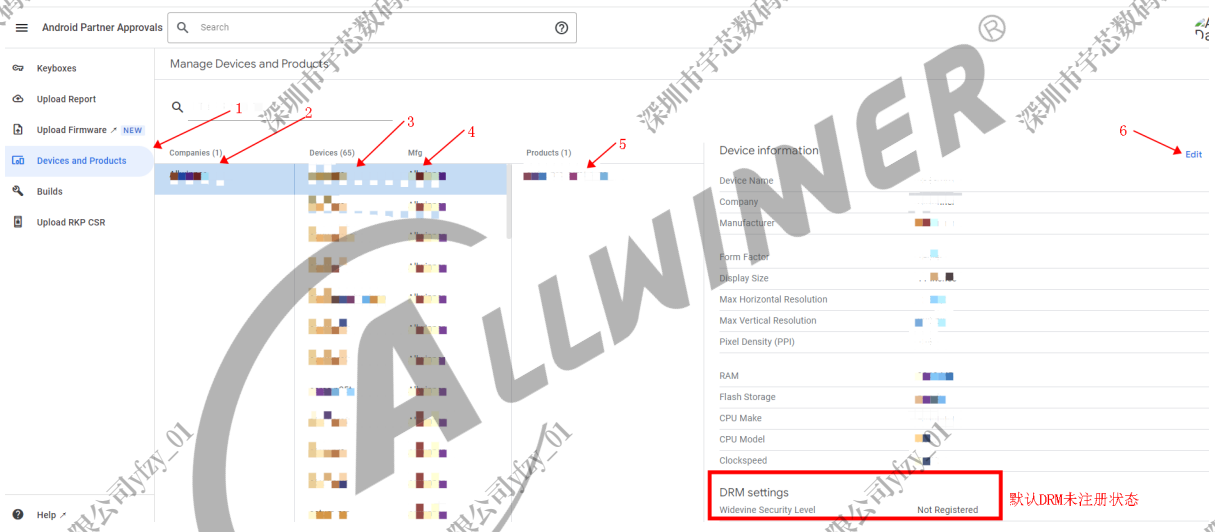

图 2-2: 审批设置 1

3. 进入到 “edit” 后，选择”Register“选项：

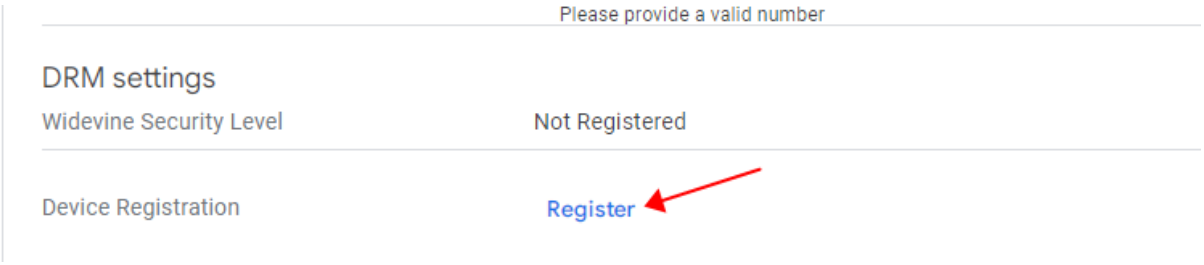

图 2-3: 审批设置 2

4. 进入后填写具体注册信息，注意，对于使用 provisioning 4.0 的设备，如 A733，注册必须选择”provisioning 4.0“的选项：

图 2-4: 审批设置 3

5. 填写完信息后并确认无误后，点击”submit“提交，之后谷歌审批完成，即可看到 DRM 的注册状态，包括使用的是 keybox 还是 provisioning 4.0：

图 2-5: 审批设置完成界面

⚠ 注意

注意：一旦注册完成，想要修改信息非常麻烦，且耗时很长，请务必确保注册 DRM 时的提交信息正确无误！新的 IC，如 A733，注册信息必须选择为 provisioning 4.0！

2.2.7 使用谷歌原生工具提取 widevine CSR json 文件操作步骤

单独编译谷歌 rkp bin：

```
cd a733-android15/system/security/provisioner

mm -j16;
```

将会在 a733-android15 out 目录的 vendor/bin 目录生成 rkp_factory_extraction_tool(lunch 32 位时) 或 rkp_factory_extraction_tool64(lunch 64 位时)

将二进制 bin 文件推送到连接的设备，然后使用以下命令运行它：

```
adb shell "chmod +x /data/rkp_factory_extraction_tool64"
adb shell "/data/rkp_factory_extraction_tool64 --output_format build+csr > /data/csr.json"
adb pull /data/csr.json 到本地目录，将csr json文件拖出来，然后再手动通过上传服务器上传给谷歌。
```

说明

注意: 使用谷歌原生工具的操作，仅用于本地调试。

2.2.8 使用全志 RKP 工具提取 widevine CSR json 文件操作步骤

1. 终端设备机器刷写固件升级后，需要确保启动过一次 Android 系统，才能确保密钥正常提取。
2. 升级 DragonSN 工具版本至 V3.0.3 及以上，然后打开 DragonSN 工具，打开左下角” 配置 key“选项：

图 2-6: DragonSN 设置 1

3. 一共需要配置如下四项：

图 2-7: DragonSN 设置 2

4. 点击” 添加 “选项，配置 key-extraction，自定义文件保存路径，这里选择为 D 盘 test 目录，并勾选” 读取 widevine“选项，配置后保存:

图 2-8: DragonSN 设置 3

5. 点击” 添加 “选项，配置 key-widevine，自定义文件保存路径，这里选择为 D 盘 test 目录，并勾选” 读取 widevine“选项，配置后保存:

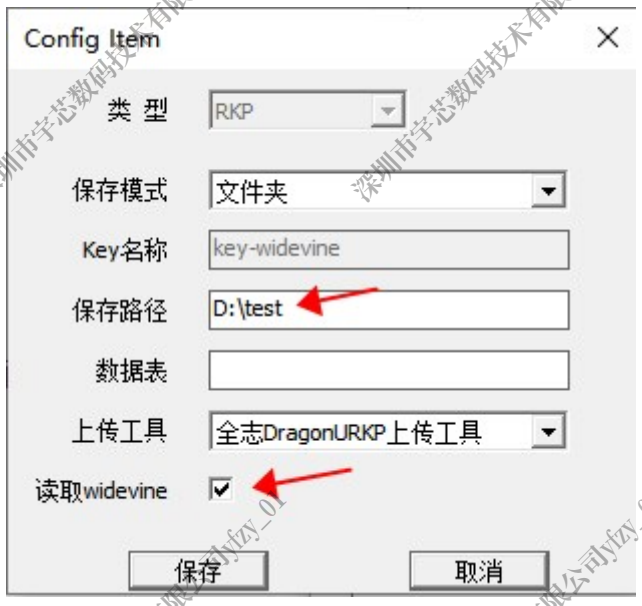

图 2-9: DragonSN 设置 4

6. 点击” 添加 “选项，配置 chipid，配置 key type 为 efuse，配置后保存:

图 2-10: DragonSN 设置 5

7. 点击” 添加 “选项，配置 reported-serial，配置 key type 为 flash，配置后保存:

图 2-11: DragonSN 设置 6

8. 配置完成后打开 DragonSN，连接上有线 usb，关机或重启，Dragon 工具将会自动提取，如下是提取成功的打印：

图 2-12: DragonSN 提取成功界面

9. 进入保存目录确认提取的文件是否包含 widevine 信息。上面配置的是 D 盘 test 目录，因此这里进到对应目录。由于上面配置了 key-extraction 和 key-widevine，因此会在 D:/test 目录生成 2 个 json 文件，分别是 key-extraction 和 key-widevine 的文件：

图 2-13: 提取出来的 csr json 文件

而 key-extraction 还勾选” 读取 widevine “选项，因此 key-extraction 会同时包含 keymint 和 widevine 的 CSR 信息，如下：

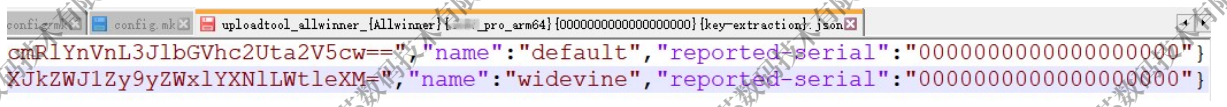

图 2-14: 包含 keymint 和 widevine 的 CSR 信息

而 key-widevine 则只包含 widevine 的 CSR 信息，如下：

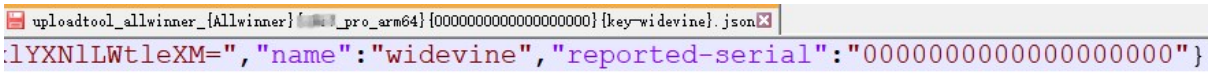

图 2-15: 只包含 widevine 的 CSR 信息

10. 若同时需要支持 keymint 和 widevine，可以上传 key-extraction 的 json 文件；若只需要支持 widevine，只上传 key-widevine 的 json 文件即可。

2.2.9 使用全志 RKP 工具上传 widevine CSR json 文件操作步骤

工具端的同事已开发了一款专门用于 RKP 上传的工具——DragonURKP，使得上传操作便捷高效，需要特别注意的是 DragonURKP 工具必须安装在一台可以正常访问 Google 云服务（Google Cloud Service）网站的电脑上。RKP 上传详细操作步骤参看《DragonURKP_ 使用指南》，该使用指南可在 APST 中 DragonURKP 工具安装目录下获取。

提取出来的 CSR json 文件，由于上传需要可访问外网，因此内部上传可以请王帅帮忙上传 CSR 文件到 Google 云服务。

⚠ 注意

2.2.10 注意事项

1. **Google Cloud Service 仅能由 MADA 资质厂商申请。工厂需要将提取的 RKP 密钥发送给 MADA OEM，然后由 MADA OEM 执行上传操作。**
2. **工厂进行密钥提取时需要使用出货的 Android GMS 固件，且此固件烧写后需要确保启动过一次 Android 系统，才能确保密钥正常提取。**
3. **QA 需要保证每一台设备的密钥已上传成功。没有上传或上传失败将导致 GMS 服务异常和安全加密视频播放失败。**

3 测试

3.1 provisioning 流程测试

3.1.1 provisioning 4.0 原理

终端设备在上传了其 csr 文件到谷歌服务器之后，接下来终端设备需要和谷歌服务器进行 provision 操作，即进行远程密钥的配置校验操作，流程如下：

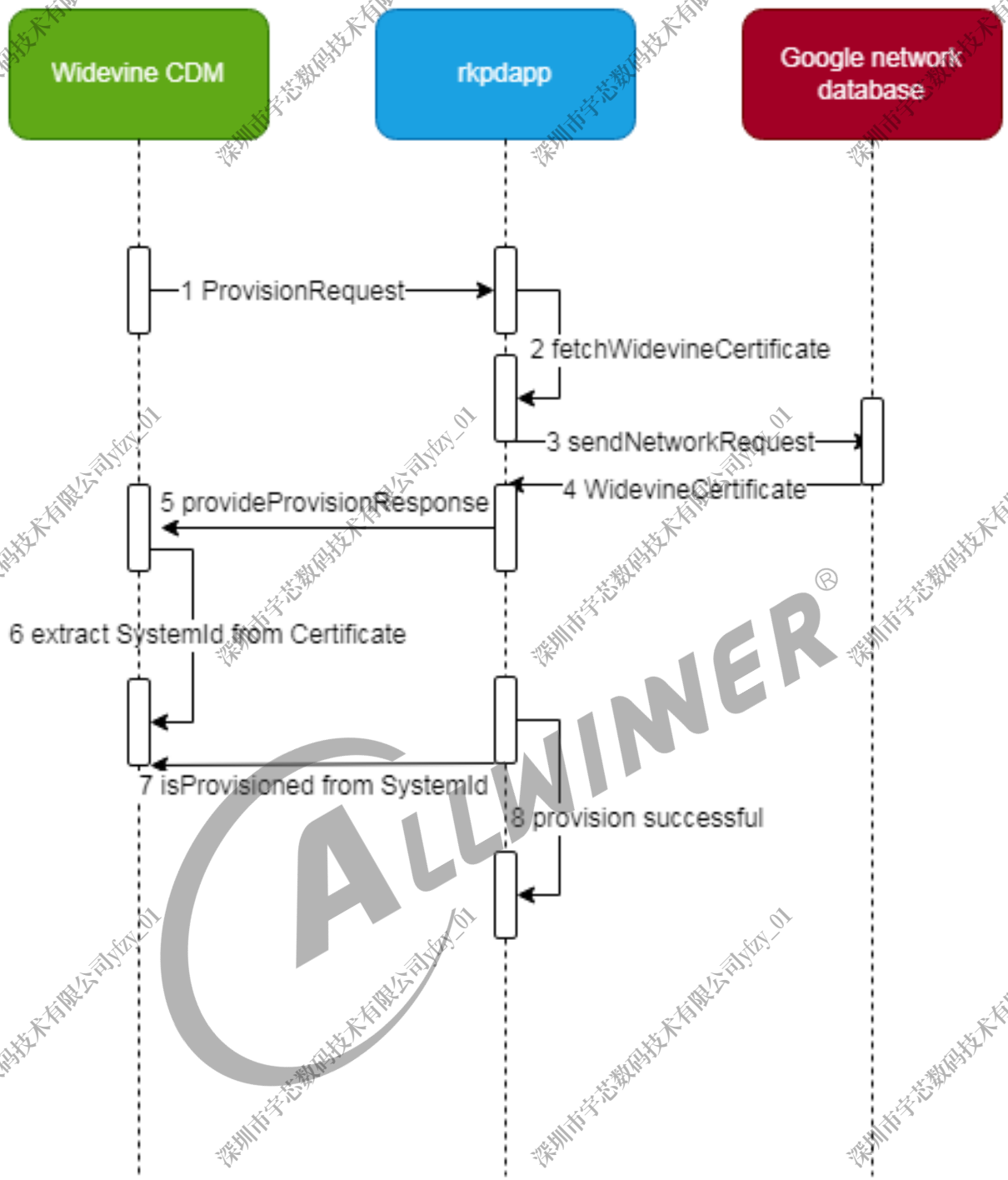

图 3-1: 密钥校验流程

终端设备的 widevine 模块，谷歌系统自带的 rkpdapp 模块，谷歌远程数据库服务器，三方会进行设备的 provision 配置交互，确认 provisioning 交互是否正常。

正常的交互流程是一个 2 步配置的流程：

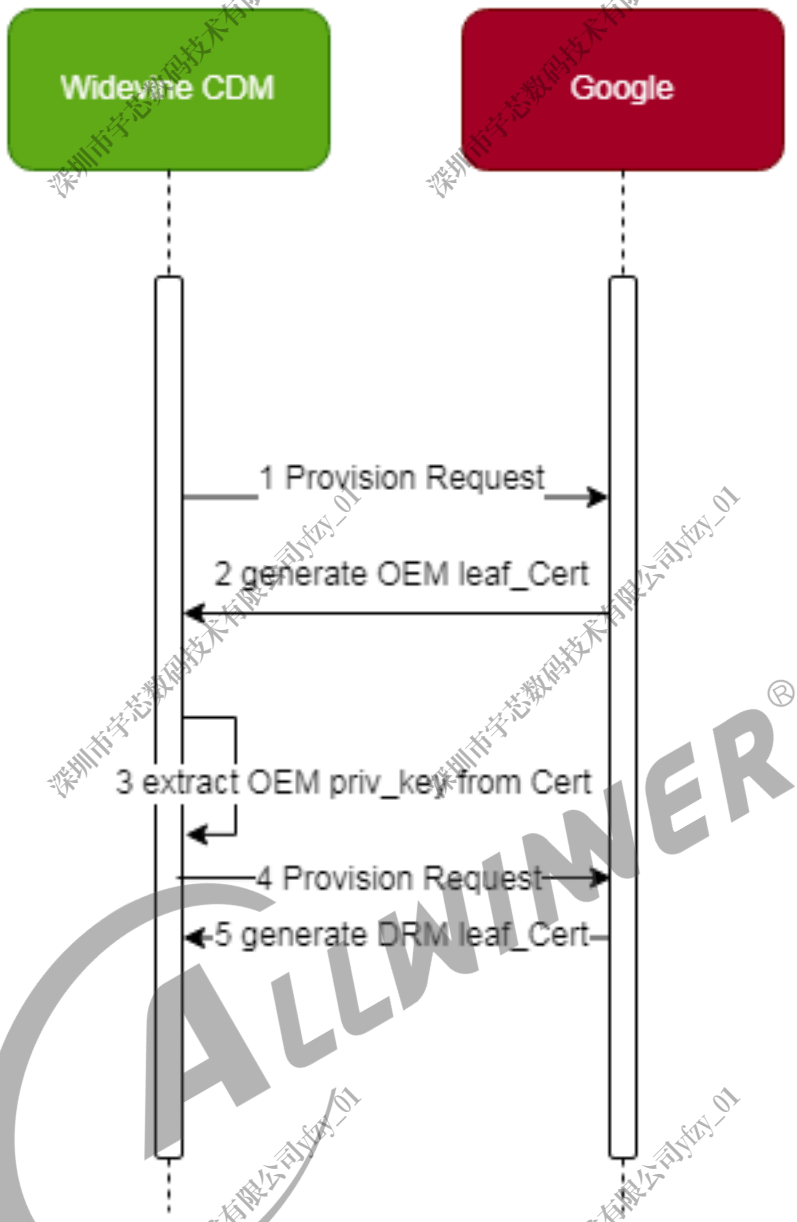

图 3-2: 2 步配置流程

3.1.2 provisioning 4.0 测试

上传完 widevine CSR json 文件后，联上外网，确认 provisioning 4.0 是否正常工作

查看 logcat 打印，如果成功，将会看到如下打印日志：

Provisioning successful.

或

This device has already been provisioned with its WV cert.

图 3-3: provision 成功的日志打印

如果失败，将会看的如下大量 RkpdWIdevine 的错误打印：

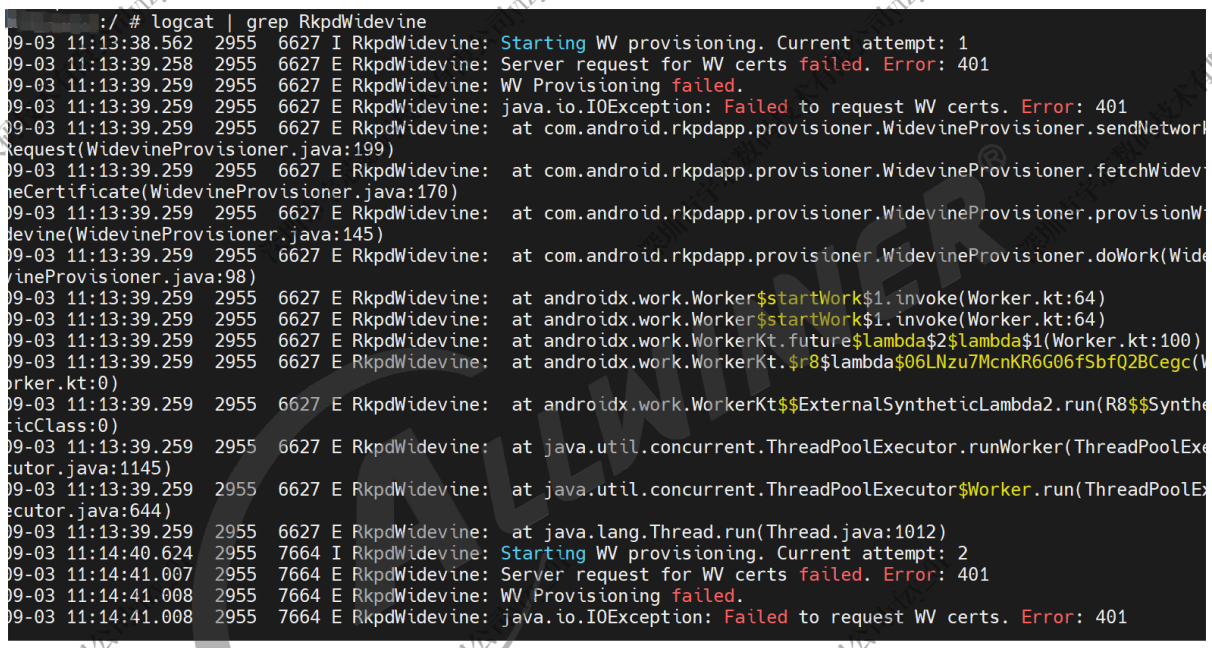

图 3-4: provision 失败的日志打印

说明

注意：每个设备的 csr 都需要单独提取并上传给谷歌，且只需提取上传一次，后续即使对设备进行全盘擦除刷机，也不会有影响，因为 csr 会保存在谷歌远程数据库服务器上面。

3.2 GMS 测试

widevine 涉及到的 GMS 测试项，是 GTS 和 VTS，其中：

3.2.1 GTS

run gts -m WvtsDeviceTestCases -s serial_name –logcat-on-failure

run gts -m GtsYouTubeTestCases -s serial_name –logcat-on-failure

run gts -m GtsMediaTestCases -s serial_name –logcat-on-failure

run gts -m GtsExoPlayerTestCases -s serial_name –logcat-on-failure

3.2.2 VTS

run vts -m VtsAidlHalDrmTargetTest -s serial_name –logcat-on-failure

⚠ 注意

注意：对于支持 widevine L1 的设备，在测试 GMS 之前，必须要确保上述 provision 测试成功，再开始 GMS 测试。

3.3 Exoplayer 测试

目前支持 h264、h265 和 vp9 的安全方案，APK 路径为 android/hardware/aw/widevine/demo，应用中带 clear 字样的片源不需要解密；带 secure video path 的字样的片源一定需要 L1 支持，如果固件烧写的为 L3 固件，在测试 L1 片源时只能播放前 10 秒。下列步骤可测试 L1 是否支持成功。

（1）安装 ExoPlayerDemo.apk。

（2）Widevine DASH Policy Tests（GTS）—–> WV:Secure video path required(MP4,H264)。

（3）播放超过 10 秒表示 L1 功能支持成功。

不需要测试带有 hdcp 字样的片源。

3.4 disney plus 测试

谷歌商店下载的迪士尼应用在 provisioning 4.0 下，需要 widevine L1 才能播放任何视频。

3.5 prime video 测试

谷歌商店下载的亚马逊应用在 provisioning 4.0 下，如果能显示 1080P 高清画质，说明 widevine L1 功能正常。

著作权声明

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明





（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。