密级状态：绝密( ) 秘密( ) 内部( ) 公开(√)

Security Class: Top-Secret( ) Secret( ) Internal( ) Public(√)

Rockchip Android Remote key Provisioning Guide

#

文件状态 Status: [√] 草稿 Draft [ ] 正式发布 Released [ ] 正在修改 Modifying	文件标识 File No.	RK-KF-YF-747
	当前版本 Current Version	V1.0
	作者 Author	蔡建清 Callen.cai
	完成日期 Finish Date	2023-02-13
	审核 Auditor	
	审核日期 Finish Date	2023-02-15

版本号 Version no.	作者 Author	修改日期 Revision Date	修改说明 Revision Description	备注 Remark
V1.0	蔡建清 callen.cai	2023-02-11	发布初稿 Initial version release	

文档问题反馈：callen.cai@rock-chips.com
If there is any question about the document, please send email to: callen.cai@rock-chips.com

概述 Overview

远程密钥配置(Remote Key Provisioning, RKP)包括一种新型的密钥管理，它将密钥配置过程从工厂内转移到空中配置。这是通过将工厂内的公钥提取和无线证书配置与短期证书相结合来实现的。作为Android认证基础设施的一部分，RKP加强了Android信任链的安全性，增加了设备信任在出现漏洞时的可恢复性。

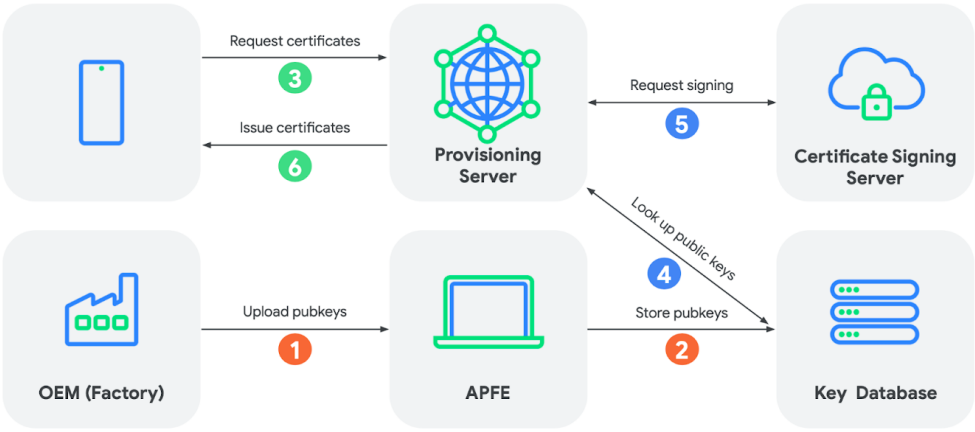

图 1. RKP工作流程图

- 在OEM工厂，从每个设备中提取唯一的公钥，并由设备制造商上传到谷歌后端服务器。
- 设备通过向provisioning 服务器证明拥有自己的设备私钥来请求认证密钥证书。服务器验证设备公钥，然后为设备签名证书。

在Android 13和更高版本上启动的设备，带有供应商软件要求(VSR) 12和更高的芯片组，必须支持RKP。

要获取和上传远程密钥配置中使用的密钥，主要需要以下三个步骤：

- 设置 Android 合作伙伴 API
- 证书签名请求提取
- 上传证书签名请求

快速实践：

如果已经设置 Android 合作伙伴 API，则进行如下操作即完成RKP的流程：

- 证书签名请求提取：

```
adb shell  rkp_factory_extraction_tool --output_format build+csr > csrs.json
```

- 上传证书签名请求：

```
./device_info_uploader.py --credentials /secure/storage/cred.json --json-csr
csrs.json --cache-token --company-id COMPANY_ID
```

- 进行GTS测试：

```
run gts -m GtsGmscoreHostTestCases
```

注意：

RKP当前不是替代attestation Key，而是一个补充。以下描述设备的RKP验证，确保设备已经正常烧录attestation key(当前请使用1.5.3版本key烧录工具)。

可以使用以下方式验证是否已经烧录attestation key:

```
adb push VtsAidlKeyMintTargetTest /data/local/tmp/
adb shell chmod +x /data/local/tmp/VtsAidlKeyMintTargetTest
adb shell
su
./data/local/tmp/VtsAidlKeyMintTargetTest --
gtest_filter=PerInstance/AttestKeyTest.RsaAttestedAttestKeys/0_android_hardware_
security_keymint_IKeyMintDevice_default
```

`VtsAidlKeyMintTargetTest` 测试项可以在 `hardware/interfaces/security/keymint/aidl/vts/functional` 执行:

`mm`编译生成。

1. 设置 Android 合作伙伴 API

Android 合作伙伴 API 是一个支持用户帐户的Google Cloud Project (GCP) API 。如果您之前已将 Android Partner API 用于其他服务并且已将凭据存储在`cred.json`文件中，则可以重复使用这些凭据。

如果您以前从未使用过该 API 并且需要 API 访问权限，请参阅设置 Android 合作伙伴 API 。为您的 Google Cloud 项目创建 OAuth 2.0 客户端 ID 后，您可以下载并将其保存为`cred.json`文件。

所需权限：本节中所附链接只能由具有 GMS 分发许可证的 MADA 或 MADA 变体（例如 EMADA、TMADA 或 TADA）签署方访问。请仔细按照其具体步骤配置。

注意：

如果是新配置的Android 合作伙伴API,需要发送启用 Android Partner API 的请求，然后在GCP中enable。

2. 证书签名请求提取

在出厂时，需要从每个设备中提取包含设备公钥的证书签名请求。 Google 提供了一个参考提取工具rkp_factory_extraction_tool，已集成在SDK代码编译的固件中。提取工具的输出被写入单独的文件。

提取设备证书签名请求:

```
adb shell  rkp_factory_extraction_tool --output_format build+csr > csrs.json
```

注:

工具源代码位于`system/security/provisioner/rkp_factory_extraction_tool.cpp`，如果需要，可自行订制开发。

工具命令行标志

工厂提取工具采用以下命令行标志：

参数	描述
`--test_mode`	如果通过，该工具将在测试模式下运行，并且底层 `IRemotelyProvisionedComponent` 标志会生成一个临时密钥。使用此标志生成的有效负载**不**适合上传。
`--self_test`	如果通过，该工具将在测试模式下获取 CSR，并使用 VTS 用于验证芯片组的相同逻辑对其进行验证。
`--output_format`	接受一个格式标志来改变输出。有效标志如下：`csr`（默认）：CBOR 证书请求结构。`build+csr`：包含构建指纹和 CBOR 证书请求的 JSON 字符串。添加此标志是为了简化从哪些设备收集哪些 CSR 的跟踪。

工具退出代码

工厂提取工具具有以下退出代码：

代码	结果
`0`	成功
`1`	执行工具期间发生意外错误。描述写入 `stderr`。
`-1`	向工具传递了无效的命令行参数。描述写入 `stderr`。

3.上传证书签名请求

本节描述如何将证书签名请求息上传到 Google 的后端服务器。

1. 准备环境：
   - 使用以下命令安装所需的依赖项：

     ```
     pip3 install google-auth==2.13.0 requests==2.28
     ```

   - 下载 `device_info_uploader.py`。
2. 使用证书签名请求提取: `csrs.json`。
3. 找到您需要在上传命令中使用的 `COMPANY_ID`。要找到此 ID，请登录到 Android Partner Approvals 门户并查看公司元数据（**Builds** > **Metadata**）。一个例子如下所示：

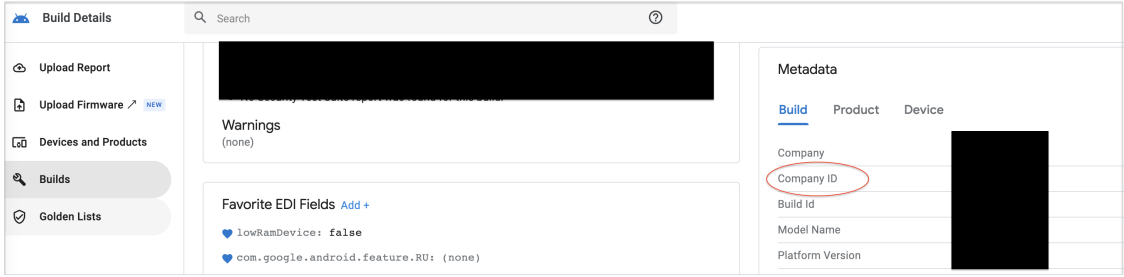

图 2. Partner Approvals 门户中的公司 ID

4. 上传设备信息。
    - 要上传新的设备信息，请使用 `batchUpload` 服务。
    - 要更新现有设备注册，请使用 `batchUpdate` 服务。

要通过 API 调用，请使用以下选项之一：

- 选项 1：通过 GCP 帐户使用 `device_info_uploader.py`
- 选项 2：将 `device_info_uploader.py` 与 GCP 服务帐户一起使用
- 选项 3：直接使用 `curl` 命令调用Android Partner API

选项 1：通过 GCP 帐户使用 `device_info_uploader.py`

此选项使用 `device_info_uploader.py` 脚本。

1. 运行脚本。示例命令如下所示：

    ```
    ./device_info_uploader.py --credentials /secure/storage/cred.json --json-csr
    csrs.json --cache-token --company-id COMPANY_ID
    ```

    注意： -- `--credentials` 参数需要本地安装的支持 JavaScript 的浏览器进行身份验证。使用终端（例如 `ssh` ）远程登录时，您必须使用服务帐户。为此，请在 Google Cloud 中创建一个服务帐户，将其添加到白名单，下载服务帐户的 JSON 密钥文件，然后使用 `--credentials-keyfile keyfile.json` 参数进行身份验证。

    - 将 `/secure/storage/cred.json` 替换为您的凭证文件路径。
    - `--cache-token` 标志将授权令牌存储在本地磁盘上。如果存在本地存储不安全的风险，请不要使用此选项。例如，不要在共享计算机上使用此选项。
    - 如下所示格式化 `csrs.json` 文件（这是 `rkp_factory_extraction_tool --output_format build+csr` 写入 `stdout` 的数据格式）。

        ```
        {"build_fingerprint":"YOUR_FINGERPRINT","csr":"YOUR_CERTIFICATE_REQUES
        T"}
        ```

2. 等待脚本输出，其中包含要登录的 URL。
3. 复制脚本输出中显示的登录 URL 并将其粘贴到新的浏览器选项卡中（这将打开 Google 登录屏幕）。选择您的公司帐户（格式为 `android-partner-api@company.com` ）。一个例子如下所示：

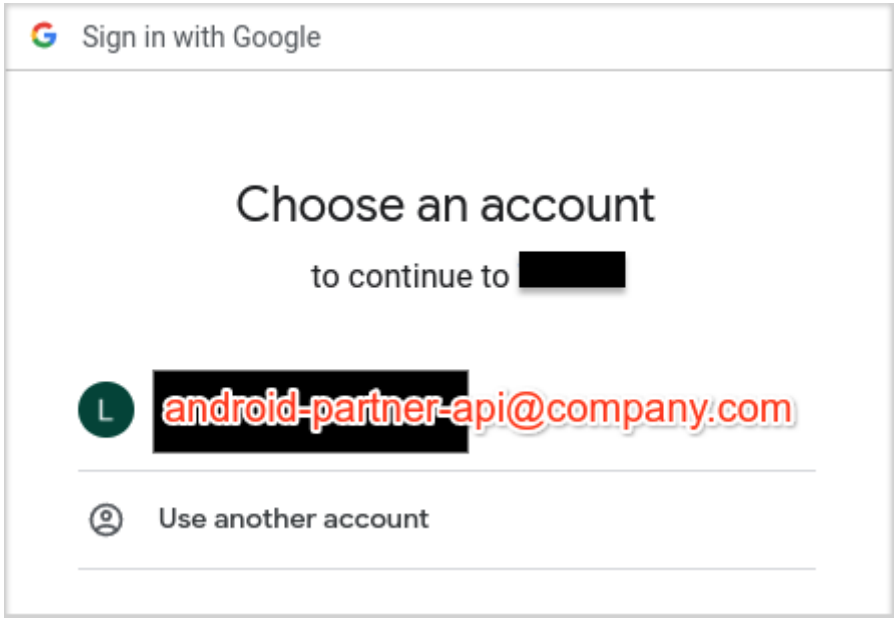

图 3.选择一个帐户

4. 选择“**允许**”以授予对您帐户的测试访问权限，使其能够查看和管理合作伙伴设备信息。一个例子如下所示：

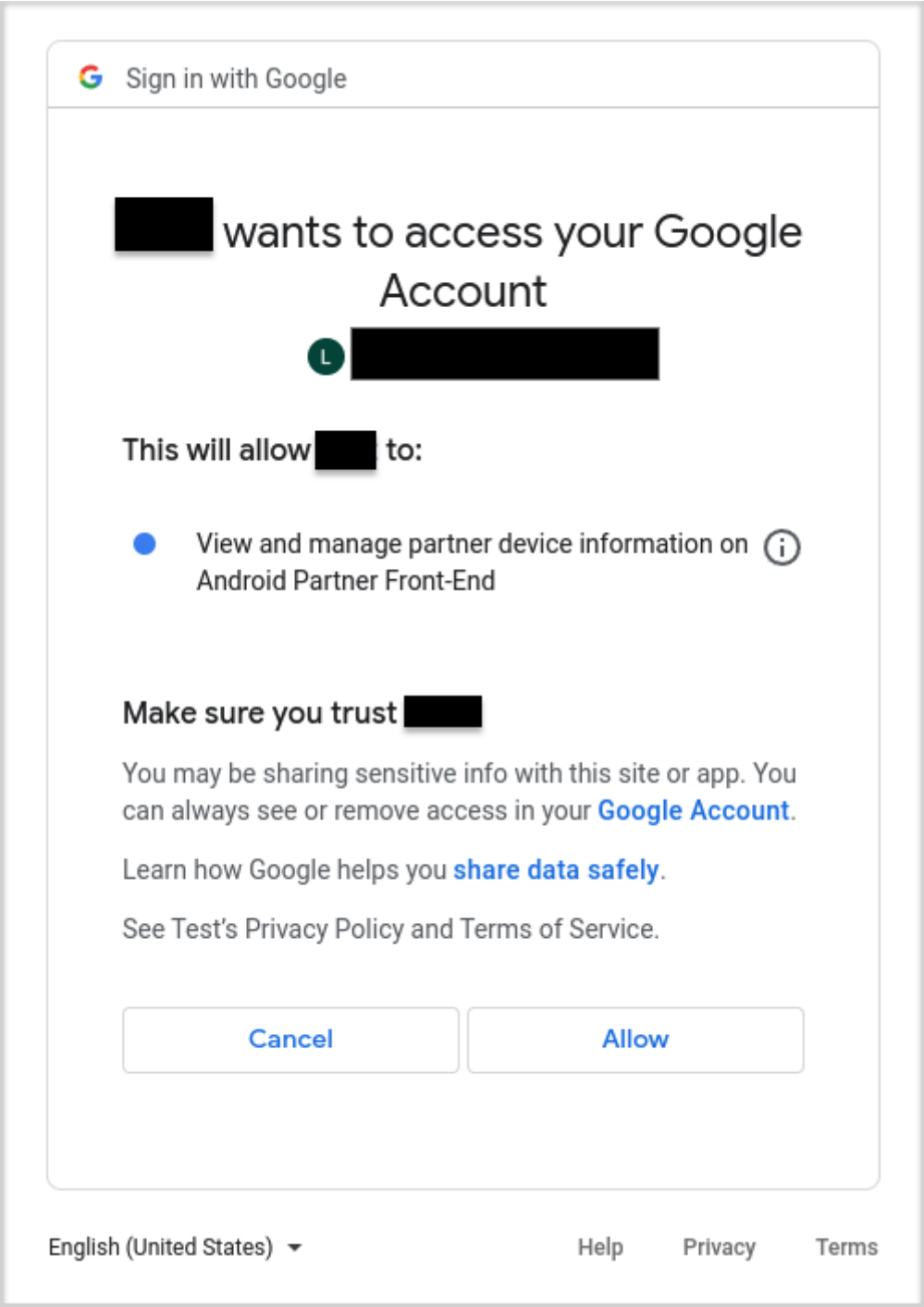

图 4.允许访问 Google 帐户

选项 2：将`device_info_uploader.py`与 GCP 服务帐户一起使用

使用`csrs.json`文件和您在Service Account Credentials中下载的`my-project-0000000000-ffffffffffff.json`密钥文件运行`device_info_uploader.py`脚本。示例命令如下所示：

```
./device_info_uploader.py --credentials-keyfile /secure/storage/my-project-0000000000-ffffffffffff.json --json-csr csrs.json --company-id COMPANY_ID
```

选项 3：使用`curl`命令直接调用 Android Partner API。

使用`curl`调用 Android Partner API 并上传文件。示例命令如下所示：

```
curl -s -X POST -H "Content-Type: application/json" -H "$(oauth2l header --json
/secure/storage/cred.json androidPartner)" -H "X-GFE-SSL: yes" -d '{company_id:
COMPANY_ID, device_payloads: [{certificate_payloads:
[{certificate_signing_request: "CERTIFICATE_SIGNING_REQUEST"}]}]}'
https://androidpartner.googleapis.com/v1/devices:batchUpload
```

替换以下值：

- 将`/secure/storage/cred.json`替换为您的凭证文件路径。
- 将`COMPANY_ID`替换为您的 Android 合作伙伴管理公司 ID。
- 将`CERTIFICATE_SIGNING_REQUEST`替换为提取工具生成的 CBOR 编码证书请求。

4.测试验证

设备验证建议

对于每个 X 设备测试，我们**强烈建议**将设备连接到外部网络，然后从 GTS 运行`AttestationRootHostTest`。成功通过此测试的设备已正确注册并已收到来自谷歌的有效证明证书（您必须在设备到达测试站之前上传这些设备的公钥）。

确保提取和上传的可靠性

重要提示：如果设备的公钥未上传到 Google 的后端服务器，则设备无法检索证明密钥。

确保提取和上传是**100% 可靠的**。从提取工具中检索到的数据块应存储在数据冗余存储中，以确保上传前的数据完整性。

确保数据安全

从设备中提取的数据使用私钥加密，该私钥只能由 Google 的后端服务器访问（实际加密的数据是公钥，不需要高安全性存储）。为确保此数据安全：

- 限制对存储的写访问。恶意行为者可以将他们自己的伪造设备公钥添加到工厂数据库中，Google 会认真对待这种攻击向量。
- 限制和保护用于验证上传到 Google 后端服务器的 Android 合作伙伴 API 凭据。

5.RKP API 错误

此节列出了为 RKP 上传证书签名请求 (CSR) 时可能发生的 API 错误，供在测试过程中进行排查。

与特定问题相关的错误

- **公司 ID 的用户权限被拒绝**

当前用户没有为指定公司 ID 上传 CSR 的必要权限时，会发生此错误。

> 消息：未上传此 CSR。不允许用户 {USER} 上传公司 ID {COMPANY_ID} 的设备信息。请提交错误或联系 TAM 以获得权限。

> 代码：7（权限被拒绝）403 禁止访问

- **密钥已存在**

当用户上传 CSR 并且其公钥已存在于先前上传的 CSR 中时，会发生此错误。

> 消息：未上传此 CSR。设备公钥已存在。请调用 BatchUpdate 来修改现有密钥的数据。
>
> 代码：6（已存在）400 错误请求

要修改与公钥相关的设备信息，请使用更新API。

- **密钥不存在**

当用户尝试更新不存在的密钥时会发生此错误。

> 消息：未上传此 CSR。找不到设备公钥。请先调用BatchUpload API。
>
> 代码：5（未找到）400 错误请求

- **缺少 CSR 版本**

当用户上传缺少 CSR 版本的 CSR 时，会发生此错误。

> 消息：未上传此 CSR。 CSR 缺少版本，请确保您的 CSR 具有正确的字段。
>
> 代码：3（无效参数）400 错误请求

CSR 版本是必需的，因为它用于确定系统是否验证（或不验证）CSR 的制造商。

- **未找到制造商**

当无法从上传的 CSR 中提取制造商时，就会出现这些错误。错误消息和系统行为因 CSR 版本而异。

- CSR 版本 1：已上传 CSR，但会显示一条警告消息。

  > 消息：此 CSR 已成功上传。设备信息中的制造商信息在 VSR12 芯片组上是可选的，但从 VSR13 开始是必需的。请确保您的工厂工具已准备好在未来的芯片组上配置此信息。 （这是 HTTP 200 成功）

- CSR 版本 2 或更高版本：

  > 消息：未上传此 CSR。从 VSR13 芯片组开始，设备信息中的制造商信息是必需的。
  >
  > 代码：3（无效参数）400 错误请求

- **制造商的许可被拒绝**

当用户没有必要的权限来上传指定制造商的证书时，就会出现这些错误。 CSR 版本之间的错误消息和系统行为不同：

A.CSR 版本 1：已上传 CSR，但会显示一条警告消息。

> 消息：此 CSR 已成功上传。但是，您无权上传制造商 {MANUFACTURER} 的证书。我们仍然接受此带有 VSR 12 芯片组的 CSR 上传。请确保您拥有正确的权限以上传 VSR 13+ 芯片组（这是 HTTP 200 成功）

B.CSR 版本 2 或更高版本：未上传 CSR 并显示以下错误消息。

> 消息：未上传此 CSR。为制造商 {MANUFACTURER} 上传证书的权限被拒绝。请在 Android Partner Approvals 门户上注册并上传该制造商的报告。
>
> 代码：7（权限被拒绝）403 禁止访问

注意：如果您是第一次代表新的或未注册的`manufacturer`上传 CSR，请将包含此`manufacturer`信息的报告提交到 Android Partner Approvals 门户。

- **同一请求中的重复键**

当用户多次上传同一个密钥时会发生此错误。

> 消息：在请求中找到重复的设备公钥。密钥已经上传。
>
> 代码：3（无效参数）400 错误请求

- **CSR 的最大数量**

当用户每次请求上传超过 5000 个 CSR 时，会发生此错误。

> 消息：已达到 CSR 的最大数量。请不要为每个 DevicePayload 上传超过 5000 个 CSR。
>
> 代码：3（无效参数）400 错误请求

- **无法反序列化 CSR**

当用户上传无效的 CSR 时会发生此错误。

> 消息：未上传此 CSR。无法在注册期间反序列化证书签名请求。请确保您的 CSR 未损坏。
>
> 代码：3（无效参数）400 错误请求

一般错误

- **截止日期已过**

当用户上传太多密钥并且截止日期到期时，会发生此错误。

> 消息：已超过最后期限。请尝试在每个请求中使用较少的 CSR 重新上传。如果问题仍然存在，请联系您的 Google TAM。代码：4（超过截止日期）504 网关超时

- **内部错误**

此错误未缓存，需要 Google 工程人员对其进行调试。

> 消息：内部错误。
>
> 代码：13（内部错误）

- **未经认证**

当用户未通过身份验证时会发生此错误。有关详细信息，请参阅设置 Android 合作伙伴 API 。

> 消息：未经身份验证。
>
> 代码：16（未认证）401

- **资源耗尽**

当资源耗尽时会发生此错误，例如，已达到每个用户的配额或文件系统空间不足。请稍后重试该请求。

> 代码：8 (RESOURCE_EXHAUSTED) 429 请求过多

- **暂停服务**

当服务当前不可用时会发生此错误。这很可能是一种瞬态情况。退避重试该服务。

> 消息：服务不可用。
>
> 代码：14 (UNAVAILABLE) 503 服务不可用

- **中止**

当操作中止时会发生此错误，通常是由于并发问题，例如定序器检查失败或事务中止。

> 消息：中止
>
> 代码：10 (ABORTED) 409 冲突

- **Google Cloud Platform 服务已被停用**

如果用户尝试使用合作伙伴域帐户创建 Google Cloud Platform (GCP) 项目，则会显示此消息： *Please contact your administrator to turn the service on the G Suite Admin console* 。

您不能使用合作伙伴域帐户来创建 GCP 项目。使用您的公司帐户创建 GCP 项目。如果您使用的是 Google Workspace，请联系您的 IT 管理员为您的帐号启用 GCP。

6.RKP 常见问题

设备信息上传/更新

1. **为什么我会收到 500“遇到内部错误。”?**

某些 TEE 实现在其请求的格式中存在错误。这会导致服务器无法解析负载，从而导致 500 内部错误。请重试 反馈给我们。

2. **为什么每个 CSR 都不同，即使是从同一设备中提取的？设备公钥是否更改?**

在 Android 13 及更低版本中，需要 HAL 将部分 CSR 加密为设备公钥。这种加密方案涉及一些随机性，这意味着每次 V1 或 V2 HAL 生成证书签名请求 (CSR) 时密文都是不同的。请注意，每次生成 CSR 时明文密钥都必须相同，但如果不联系可以解密请求的 Google 后端团队，则无法验证 CSR。

3. **如何查看设备密钥明文?**

在 Android 13 及更低版本中，设备密钥在 HAL 内加密。只有谷歌后端服务器可以执行解密——甚至连谷歌员工都无法访问。因此，无法在 Android 13 上查看设备密钥。

从 Android 14 开始，不再要求在 HAL 中加密密钥。这既简化了架构又提高了设计的可测试性。

4. **我可以在设备信息中使用非生产型号名称吗?**

设备信息中的型号和制造商条目必须与贵公司在 Android 合作伙伴审批门户中注册的信息相匹配。

上传 CSR

1. **如果我们上传损坏的 CSR，是否会出现错误?**

是的，如果 CSR 损坏，API 会返回记录在案的错误。后端服务器验证在 CSR 中执行的加密操作。

2. **如果我上传了错误的 CSR，我可以取消或更改它吗?**

是的，`device_info_uploader`脚本会使用最新信息自动更新设备，因此您可以重新运行上传工具来更新设备。

3. **如果我上传重复的 CSR 会怎样?**

在正常的上传 API 调用中，服务返回密钥已上传的错误。如果您使用更新 API 调用，则密钥会更新为新的元数据（设备信息，例如`manufacturer`和`model`）。

4. **一次可以上传多少个 CSR 有限制吗?**

您一次最多可以上传 5,000 个 CSR。无论您将多少 CSR 传递给上传工具，上传工具都会自动将这些 CSR 以 2,000 个为一组进行批处理。例如，发送到上传器工具的 100,000 个 CSR 将分 50 批转发到 Google API，每批 2,000 个 CSR。

5. **上传 CSR 和设备可用之间的延迟是多长时间?**

没有延迟。 CSR 上传到 Google 后端后，RKP 会立即在设备上运行。

6. **是否可以检查在 Google 后端服务器中注册的 CSR?**

不行，但Google正在研究将此添加为一项功能。

现在，您可以执行更新操作并查看是否收到指示设备不存在的错误。请注意，更新的信息是持久化的。

比如尝试上传同一个 证书签名请求文件或者同一设备不同时间生成的证书签名请求文件，device_info_uploader.py 加个参数：--no-update

```
./device_info_uploader.py --credentials /secure/storage/cred.json --json-csr
csrs.json --cache-token --company-id COMPANY_ID --no-update
```

会收到类似以下返回信息：

```
"status": {
        "code": 6,
        "message": "This CSR was not uploaded. Device public key already exists.
Please call BatchUpdate to modify an existing key's data."
      }
```

表明此设备的证书签名请求已经上传。

测试

1. **如何使用预生产设备测试 RKP?**

由于生产设备会生成 Google 用于签署证书请求的真实证明证书，因此早期的预生产设备不应在 RKP 后端注册为生产设备。相反，将请求负载中的`is_test`字段设置为`true`。这会将设备密钥**永久**注册为后端的测试设备。无法更新测试设备的信息以将其从测试设备转换为生产设备。

当测试设备远程提供密钥时，设备会收到链接到测试根的证书。如果测试需要生产证书（例如 GTS 测试），则设备可以注册为生产（而不是测试）设备。

`device_info_uploader.py`工具可以通过传递`--is-test`命令行标志将设备标记为测试。

遵循以下规则：

- 将**启用**安全启动的设备标记为**测试**设备。
- 将**禁用**安全启动的设备标记为**生产**设备。

2. **在 userdebug 设备上进行测试与在生产设备上进行测试是否不同?**

是的，如果您在测试版本（userdebug 或 eng）上运行测试，那么provisioning 服务会为未注册的设备提供测试证书。无需上传。这些证书链接到测试根，而不是生产根。这个根不应该被信任用于生产证明。但是，它适合测试。

如果设备已注册，则无论构建状态如何，它都将**始终**收到生产证书。

3. **我可以为不受发货限制的设备上传 CSR 吗?**

是的，您可以上传用于内部或第三方测试的 CSR。

4. **运行 AttestationRootHostTest 是否会在设备上留下痕迹?**

确实如此，但是擦除用户数据会清除此测试留下的所有内容。

5.**执行 AttestationRootHostTest 的最低要求是什么？是否必须插入 GTS 测试程序中描述的 SIM 卡?**

最低要求是：

- 能够访问 Google 后端服务器的网络连接
- 上传到谷歌后端的设备密钥

6.**如果设备使用 Android 12 启动并升级到 Android 13，它是否会无法通过 RKP GTS 测试?**

不可以。我们要求搭载 Android 13 的设备支持 RKP。确定是否需要 RKP 的精确逻辑，用伪代码表示是：

```
bool is_rkp_supported() {  # not supported on CN-GMS devices  if
has_feature("cn.google.services") AND has_package("com.google.android.gms"):
return false  # supported on T+ with VSR S+  if ro.product.first_api_level >= 33
AND ro.vendor.api_level >= 31:    return true  # otherwise unsupported  return
false}
```

7.**为什么 `testFallback` 测试或 `AttestationRootHostTest.testFallbackKey` 失败?**

`testFallback` 和 `AttestationRootHostTest.testFallbackKey` 检查您的 Keymint 实现是否可以回退到工厂配置的密钥，以防它找不到远程配置的密钥。

由于设备没有本地证明密钥，keystore 无法生成带证明的签名密钥（因为没有备份密钥来证明它），测试失败是预期的。请确保存在本地证明密钥，然后重试。

8. **为什么我的所有 RKP GTS 测试都失败了?**

失败的常见原因是 RemoteProvisioner APK 未包含在系统映像中。您必须明确包含此 APK，因为通用 AOSP 图像不包含它。 RKP 是 GMS 要求，而不是 CDD 要求，因此必须明确包含在 GMS 设备中。

`AttestationRootTest.testRemoteProvisionerInstalled`指示供应商是否已安装。如果不是，请更新您的系统映像以包含它。

9. **`com.android.remoteprovisioner.RemoteProvisioningException: HTTP error status encountered: 444` 是什么意思？`？**

如RemoteProvisioningException中所定义，444 表示设备未在 Google 后端服务器中注册。