Android 系统从诞生到现在的1.0版本，一共经历了三代应用签名方案，分别是v1、v2和v3方案。 - v1 方案：基于 JAR 签名。 - v2 方案：APK 签名方案 v2，在 Android 7.0 引入。 - v3 方案：APK 签名方案v3，在 Android 9.0 引入。

其中，v1 到 v2 是颠覆性的，主要是为了解决 JAR 签名方案的安全性问题，而到了 v3 方案，其实结构上并没有太大的调整，可以理解为 v2 签名方案的升级版。

v1 到 v2 方案的升级，对开发者影响是最大的，就是渠道签署的问题。v2的签名也是为了让不同渠道、市场的安装包有所区别，携带渠道的唯一标识，也即是我们俗称的渠道包。好在各大厂都开源了自己的签渠道方案，例如：Walle（美团）、VasDolly（腾讯）都是非常优秀的方案。

1. V1签名

签名工具

Android 应用的签名工具有两种：jarsigner 和 apksigner。它们的签名算法没什么区别，主要是签名使用的文件不同。它们的区别如下：

jarsigner：jdk 自带的签名工具，可以对 jar 进行签名。使用 keystore 文件进行签名。生成的签名文件默认使用 keystore 的别名命名。

apksigner：Android sdk 提供的专门用于 Android 应用的签名工具。使用 pk8、x509.pem 文件进行签名。其中 pk8 是私钥文件，x509.pem 是含有公钥的文件。生成的签名文件统一使用“CERT”命名。

签名过程

在 META-INF 文件夹下有三个文件：MANIFEST.MF、CERT.SF、CERT.RSA。它们就是签名过程中生成的文件，它们的作用如下。

MANIFEST.MF

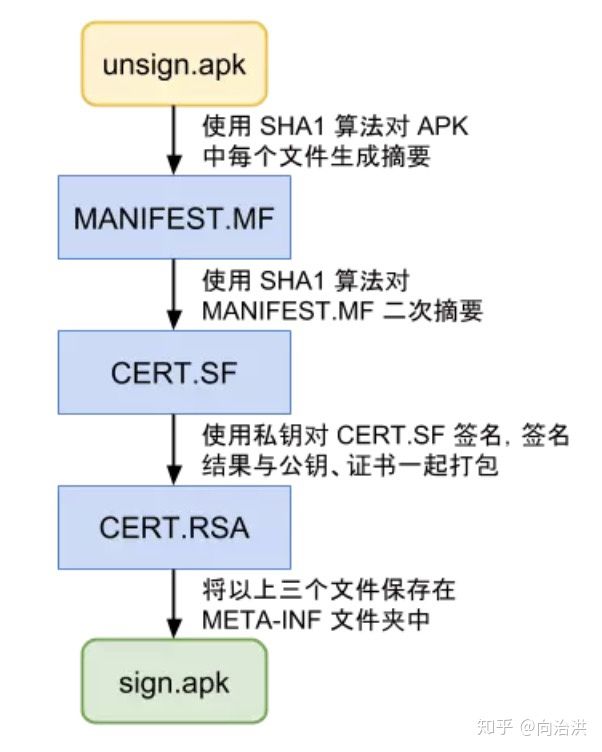
该文件中保存的其实就是逐一遍历 APK 中的所有条目，如果是目录就跳过，如果是一个文件，就用 SHA1（或者 SHA256）消息摘要算法提取出该文件的摘要然后进行 BASE64 编码后，作为「SHA1-Digest」属性的值写入到 MANIFEST.MF 文件中的一个块中。该块有一个「Name」属性， 其值就是该文件在 APK 包中的路径。

CERT.SF

- SHA1-Digest-Manifest-Main-Attributes：对 MANIFEST.MF 头部的块做 SHA1（或者SHA256）后再用 Base64 编码。 - SHA1-Digest-Manifest：对整个 MANIFEST.MF 文件做 SHA1（或者 SHA256）后再用 Base64 编码。 - SHA1-Digest：对 MANIFEST.MF 的各个条目做 SHA1（或者 SHA256）后再用 Base64 编码。

CERT.RSA

把之前生成的 CERT.SF 文件用私钥计算出签名, 然后将签名以及包含公钥信息的数字证书一同写入 CERT.RSA 中保存。需要注意的是，Android APK 中的 CERT.RSA 证书是自签名的，并不需要第三方权威机构发布或者认证的证书，用户可以在本地机器自行生成这个自签名证书。Android 目前不对应用证书进行 CA 认证。



签名校验

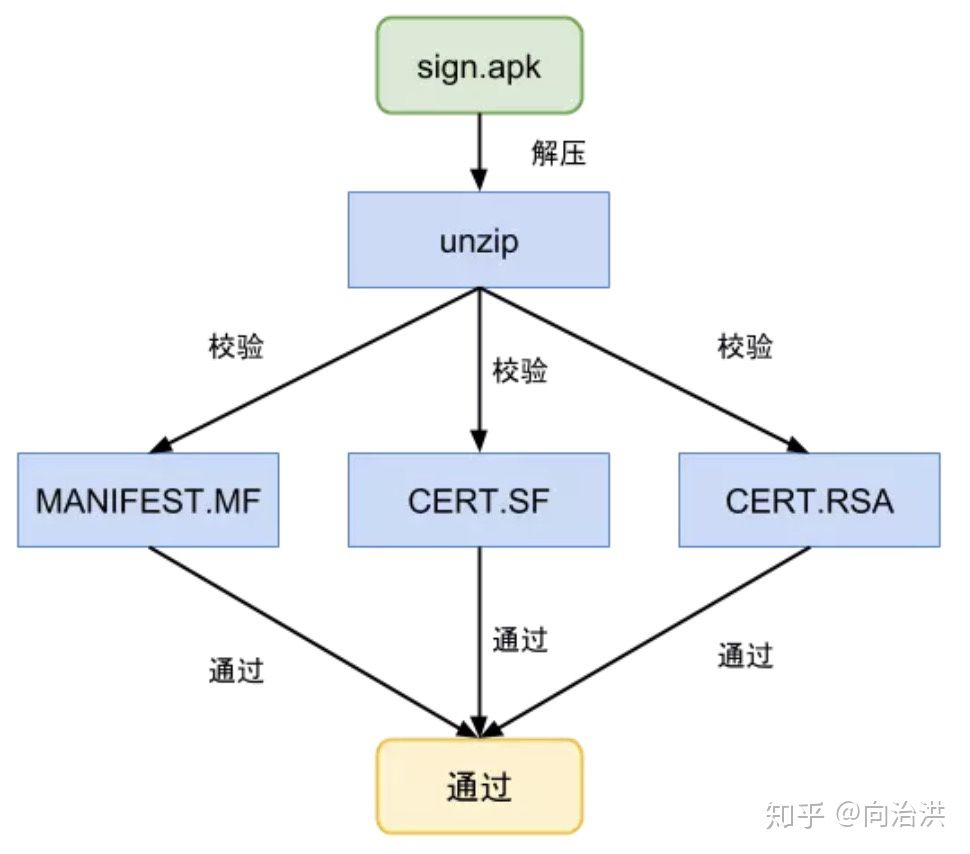
签名验证是发生在 APK 的安装过程中，一共分为三步：

1.检查 APK 中包含的所有文件，对应的摘要值与 MANIFEST.MF 文件中记录的值一致。

2.使用证书文件（RSA 文件）检验签名文件（SF 文件）没有被修改过。

3.使用签名文件（SF 文件）检验 MF 文件没有被修改过。

综上所述，一个完整的签名验证过程如下所示：



1. V2签名

APK 签名方案 v2 是一种全文件签名方案，该方案能够发现对 APK 的受保护部分进行的所有更改，从而有助于加快验证速度并增强完整性保证。通过前面的分析，可以发现 v1 签名有两个地方可以改进： - 签名校验速度慢 校验过程中需要对apk中所有文件进行摘要计算，在 APK 资源很多、性能较差的机器上签名校验会花费较长时间，导致安装速度慢。

完整性保障不够 META-INF 目录用来存放签名，自然此目录本身是不计入签名校验过程的，可以随意在这个目录中添加文件，比如一些快速批量打包方案就选择在这个目录中添加渠道文件。

为了解决这两个问题，在 Android 7.0 版本 中引入了全新的 APK Signature Scheme v2。

V2的改进

由于在 v1 仅针对单个 ZIP 条目进行验证，因此，在 APK 签署后可进行许多修改 — 可以移动甚至重新压缩文件。事实上，编译过程中要用到的 ZIPalign 工具就是这么做的，它用于根据正确的字节限制调整 ZIP 条目，以改进运行时性能。而且我们也可以利用这个东西，在打包之后修改 META-INF 目录下面的内容，或者修改 ZIP 的注释来实现多渠道的打包，在 v1 签名中都可以校验通过。

v2 签名将验证归档中的所有字节，而不是单个 ZIP 条目，因此，在签署后无法再运行 ZIPalign（必须在签名之前执行）。正因如此，现在，在编译过程中，Google 将压缩、调整和签署合并成一步完成。

v2 签名

v2 签名会在原先 APK 块中增加了一个新的块（签名块），新的块存储了签名、摘要、签名算法、证书链和额外属性等信息，这个块有特定的格式。最终的签名APK其实就有四块：头文件区、V2签名块、中央目录、尾部。下图是V1签名和V2签名的组成。



整个签名块的格式如下：

size of block，以字节数（不含此字段）计 (uint64)

带 uint64 长度前缀的“ID-值”对序列：

size of block，以字节数计 - 与第一个字段相同 (uint64)

magic“APK 签名分块 42”（16 个字节）

在多个“ID-值”对中，APK签名信息的 ID 为 0x7109871a，包含的内容如下： 带长度前缀的 signer：

带长度前缀的 signed data，包含digests序列，X.509 certificates 序列，additional attributes序列

带长度前缀的 signatures（带长度前缀）序列

带长度前缀的 public key（SubjectPublicKeyInfo，ASN.1 DER 形式）

value可能会包含多个 signer，因为Android允许多个签名。

总结一下：一个签名块，可以包含多个ID-VALUE，APK的签名信息会存放在 ID 为 0x7109871a的键值对里。他的内容可以包含多个签名者的签名信息，每个签名信息下包含signed data、signatures、public key，其中，signed data主要存放摘要序列、证书链、额外属性，signatures包含多个签名算法计算出来的签名值，public key表示签名者公钥，用于校验的时候验证签名的。

签名过程

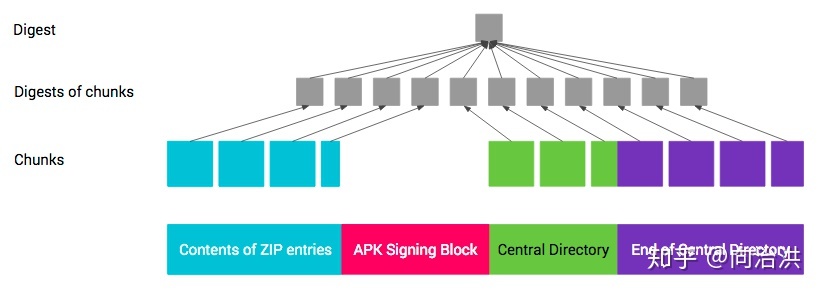
首先，说一下 APK 摘要计算规则，对于每个摘要算法，计算结果如下:

将 APK 中文件 ZIP 条目的内容、ZIP 中央目录、ZIP 中央目录结尾按照 1MB 大小分割成一些小块。

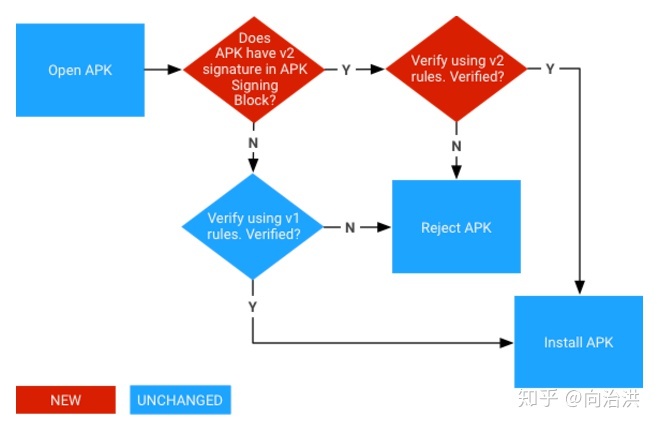
计算每个小块的数据摘要，数据内容是 0xa5 + 块字节长度 + 块内容。

计算整体的数据摘要，数据内容是 0x5a + 数据块的数量 + 每个数据块的摘要内容

总之，就是把 APK 按照 1M 大小分割，分别计算这些分段的摘要，最后把这些分段的摘要在进行计算得到最终的摘要也就是 APK 的摘要。然后将 APK 的摘要 + 数字证书 + 其他属性生成签名数据写入到 APK Signing Block 区块。



签名校验过程



其中， v2 签名机制是在 Android 7.0 以及以上版本才支持的。因此对于 Android 7.0 以及以上版本，在安装过程中，如果发现有 v2 签名块，则必须走 v2 签名机制，不能绕过。否则降级走 v1 签名机制。v1 和 v2 签名机制是可以同时存在的，其中对于 v1 和 v2 版本同时存在的时候，v1 版本的 META\_INF 的 .SF 文件属性当中有一个 X-Android-APK-Signed 属性。

X-Android-APK-Signed: 2

之前的渠道包生成方案是通过在 META-INF 目录下添加空文件，用空文件的名称来作为渠道的唯一标识。但在新的应用签名方案下 META-INF 已经被列入了保护区了，向 META-INF 添加空文件的方案会对区块 1、3、4 都会有影响。

1. V3签名

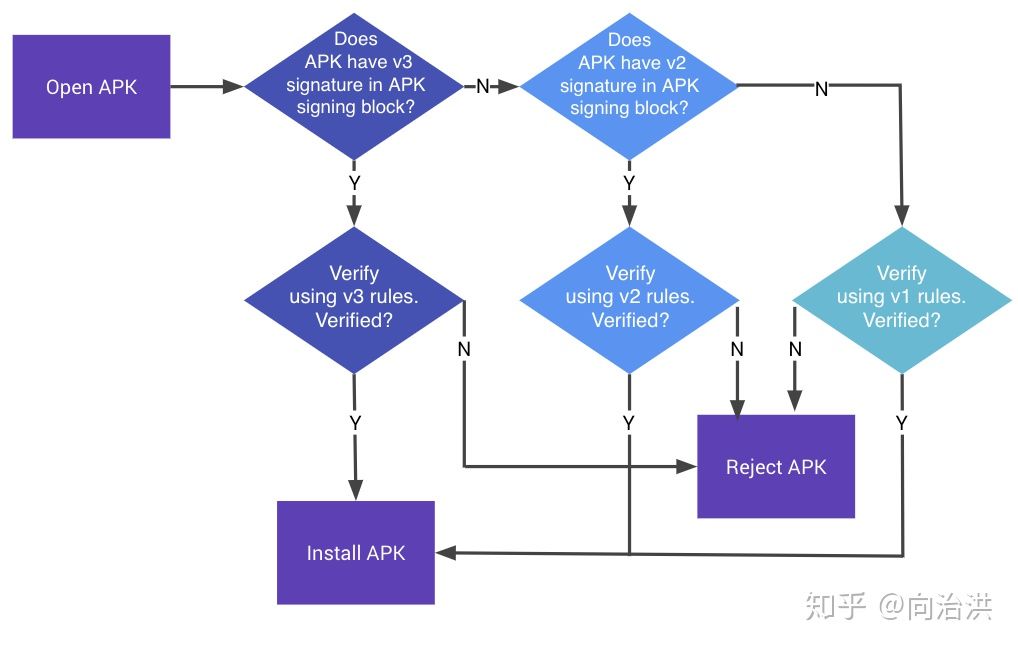
新版v3签名在v2的基础上，仍然采用检查整个压缩包的校验方式。不同的是在签名部分增可以添加新的证书（Attr块）。在这个新块中，会记录我们之前的签名信息以及新的签名信息，以密钥转轮的方案，来做签名的替换和升级。这意味着，只要旧签名证书在手，我们就可以通过它在新的 APK 文件中，更改签名。

v3 签名新增的新块（attr）存储了所有的签名信息，由更小的 Level 块，以链表的形式存储。

其中每个节点都包含用于为之前版本的应用签名的签名证书，最旧的签名证书对应根节点，系统会让每个节点中的证书为列表中下一个证书签名，从而为每个新密钥提供证据来证明它应该像旧密钥一样可信。

签名校验过程

Android 的签名方案，无论怎么升级，都是要确保向下兼容。因此，在引入 v3 方案后，Android 9.0 及更高版本中，可以根据 APK 签名方案，v3 -> v2 -> v1 依次尝试验证 APK。而较旧的平台会忽略 v3 签名并尝试 v2 签名，最后才去验证 v1 签名。 整个验证的过程，如下图：



需要注意的是，对于覆盖安装的情况，签名校验只支持升级，而不支持降级。也就是说设备上安装了一个使用 v1 签名的 APK，可以使用 v2 签名的 APK 进行覆盖安装，反之则不允许。