智能 POS V2 签名验签方案详细设计说明 V1.0.6

文档修订说明

版本号	*修订日期*	*修订摘要*	*修订人*
V1.0.0	*2018/1/22*	*创建文档*	*刘海华*
V1.0.1	*2018/1/23*	1、添加服务器签名方法简要说明 2、V2 签名 ID：0x00584744 3、添加验签标志位说明 4、修改签名块示例图片	*蔡桂汕、郭晓、刘海华*
V1.0.2	*2018/1/23*	1、排版 2、添加客户签名方案 3、添加验签标志、权限扩展选项描述说明	*蔡桂汕、刘海华*
V1.0.3	*2018/2/1*	1、添加证书更新说明	*刘海华*
V1.0.4	*2018/2/9*	1、让验签标志位使用 ***INTEGER*** 类型 2、明确签名数据的原始范围 3、修改了新国都 Id-Value 的 Id 值 4、删除了表格里的预留字段 5、更新验签流程描述及各流程图、结构图	*王海、刘海华*
V1.0.5	*2018/4/23*	1、更新验签流程图 2、细化验签流程说明	*刘海华*
V1.0.6	*2018/5/07*	1、调整了文档标题格式 2、新增了机构签名文件保存方式章节	*王海、刘海华*

目录

1.名称解释

术语	定义
非对称加密算法	非对称加密算法需要两个密钥：公开密钥和私有密钥。公开密钥 与私有密钥是一对，如果用公开密钥对数据进行加密，只

	有用对应的 私有密钥才能解密；如果用私有密钥对数据进行加密，那么只有用对 应的公开密钥才能解密。因为加密和解密使用的是两个不同的密钥， 所以这种算法叫作非对称加密算法
数字证书	数字证书，又称为电子凭证、数字证书、数字凭证、电子凭证或 数字证书，是一种用于身份识别机制
ASN.1	ANS.1(Abstract Syntax Notation dot one),抽象语法标记 1，是定义抽象数据类型形式的标准，用于描述数据表示、传输、编码的记法。它提供了一整套正规的格式用于描述对象的结构，而不管语言上如何执行及这些数据的具体指代，也不用去管到底是什么样的应用程序
DER 编码	DER（Distinguished Encoding Rules，可辨别编码规则）是 ASN.1 的一种编码方式，任何数据对应唯一的一个编码
X.509	X.509 是一种非常通用的证书格式。所有的证书都符合 ITU-T X.509 国际标准；因此(理论上)为一种应用创建的证书可以用于任何其 他符合 X.509 标准的应用。X.509 标准定义了证书中应该包含哪些信息， 并描述了这些信息是如何编码的(即数据格式)，所有的 X.509 证书包含 以下数据：X.509 版本号、证书持有人的公钥、证书的序列号、主题 信息、证书的有效期、认证机构、发布者的数字签名、签名算法标识 符
数字签名	以电子形式存在于数据信息之中的，或作为其附件的或逻辑上与 之有联系的数据，可用于辨别数据签署人的身份，并表明签署人对数 据信息中包含的信息的认可

表 1.1 名词解释

2.引言

2.1. 编写目的

解决旧方案中存在的问题，适应 Android 签名 V2 方案，实现各厂商普适性验签机制

2.2. 文档针对人员

方案设计人员，软件开发人员

2.3. 方案兼容性

本签名方案在 Android 原生 V2 签名方案基础上设计，应用签名信息以新 ID-Value 的形式追加到 APK 签名块中。 签名不破坏原有的签名信息，可以兼容原生 V1、V2 签名方案签名应用，方便存量机器同新签名方案机器同时维护使用。

3.方案详细设计

3.1. 客户签名方案

客户使用加密机生成根公、私钥对和工作公、私钥对。根公钥用于生成根证书颁发给厂商。根私钥客户保管，用于根证书和工作证书的签名加密。工作私钥也由客户保管，用于对应用的签名加密。

客户签名过程中需要上传签名应用和对应的权限文件，并设置验签选项（后面会有专门章节说明权限文件和验签选项）。

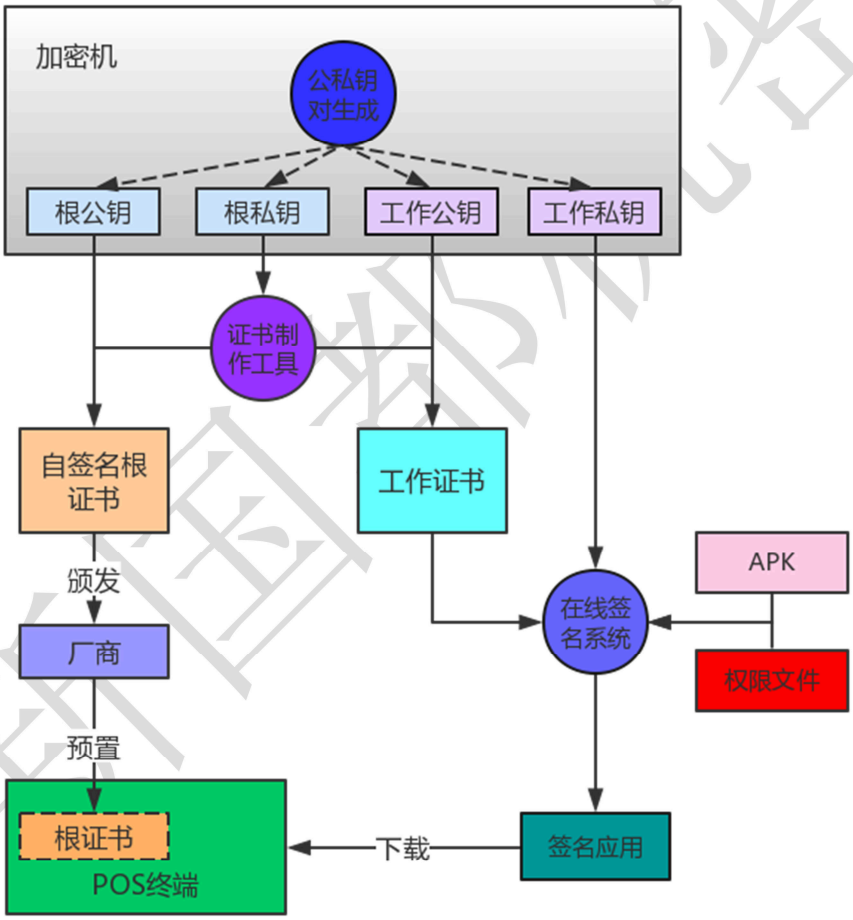


图 2.1.1 客户签名方案流程

3.1.1. 证书更新

终端允许对根证书升级，且证书升级只允许低版本向高版本升级，升级替换之后原版本签名应用将无

法安装到终端。

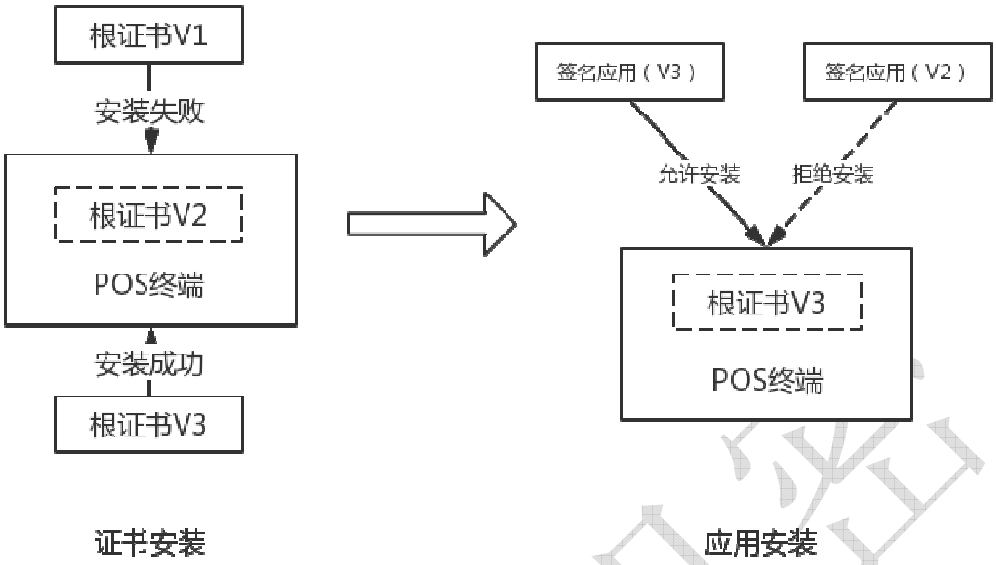


图 2.1.1.1 证书替换说明

3.2. 应用签名详细流程

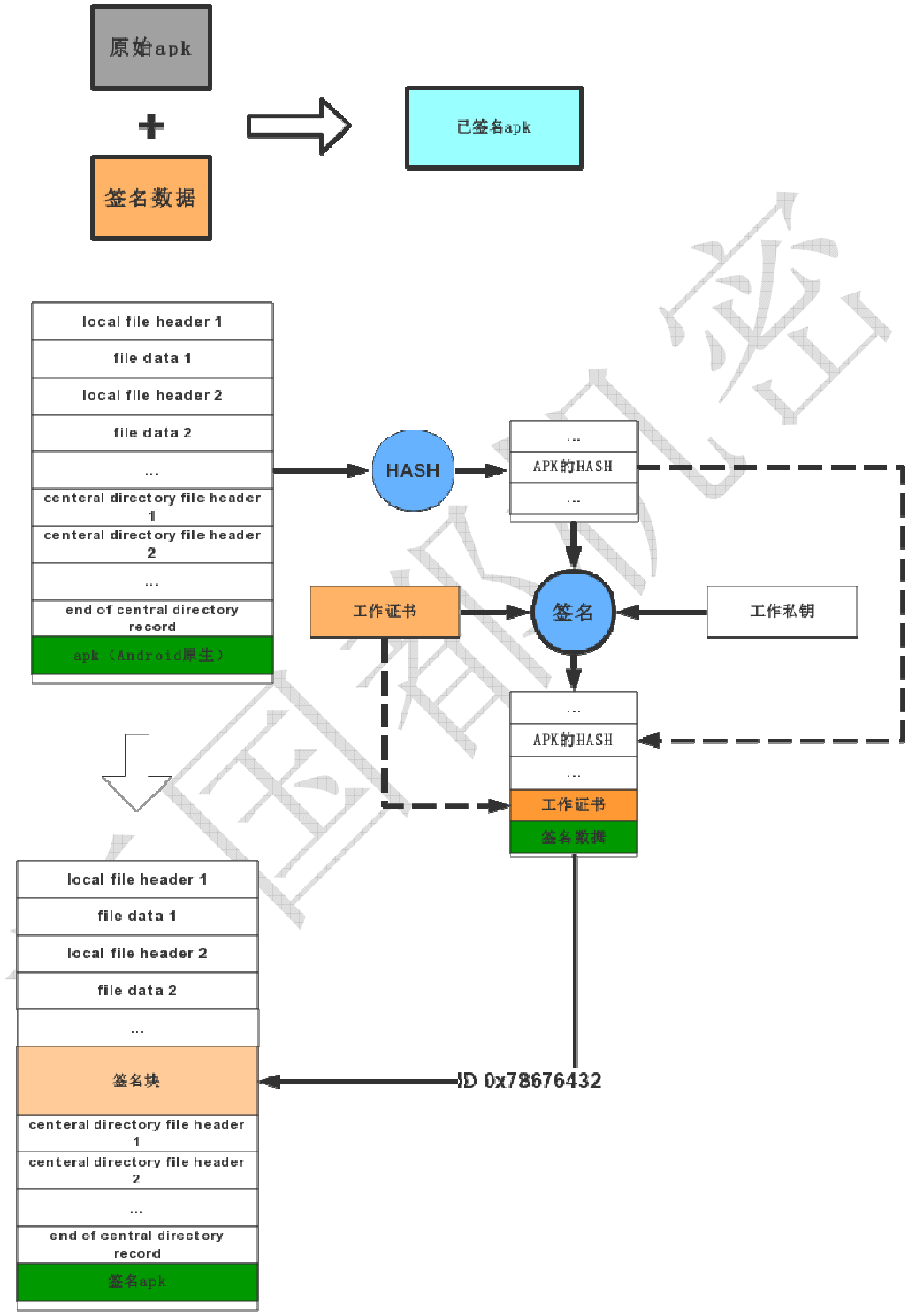


图 2.2.1 APK 签名示意图

计算原始 apk hash 值，将 hash 值加入签名信息主体，再利用工作私钥签名，将签名数据、签名信息主体、工作证书以 ID 0x78676432 插入到签名块中。

如应用未签过原生 V2 签名，则以“XGD Sig Block 42”为魔数生成签名块。如应用签过原生 V2 签名，则将机构签名信息数据插入到原生签名数据之后，生成签名后的应用。

3.3. 终端设备验签流程

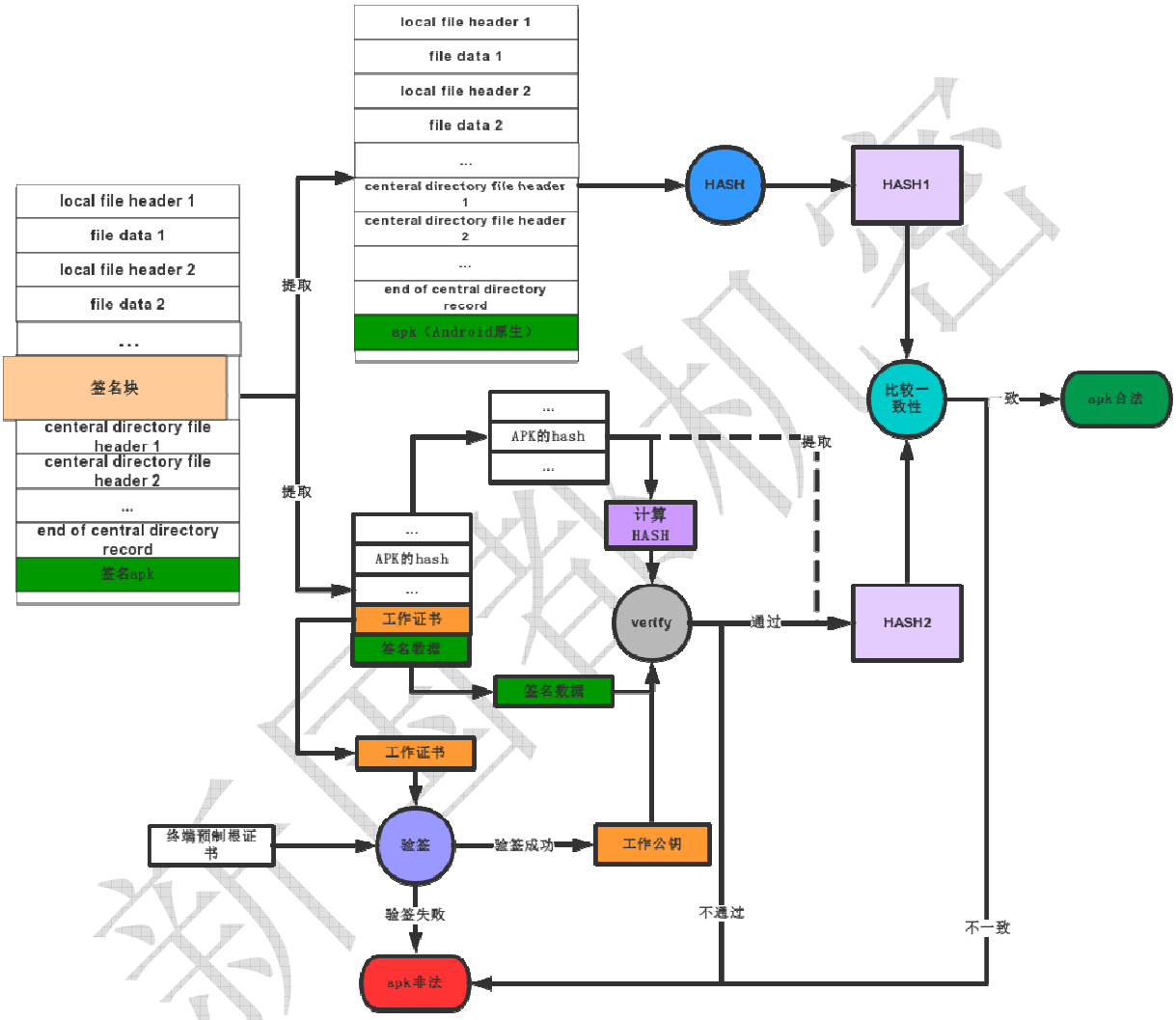


图 2.3.1 POS 终端验签流程图

终端设备在出厂时，厂商将客户颁发的根证书预装在终端设备中。终端设备安装应用时，首先进行 Android 原生验签，然后进行机构验签。机构验签流程如下：

1. 根据魔数”XGD Sig Block 42”或”APK Sig Block 42”查找签名块;
2. 提取签名块中的机构签名数据，并分离得到原始 apk;
3. 分离机构签名信息得到签名数据和工作证书数据;
4. 使用终端保存的根证书验证工作证书的合法性，如果失败则证明 APK 非法，验签终止；如果验签成功，则从工作证书中提取出工作公钥；
5. 使用工作公钥验证机构签名的合法性，计算签名主体信息的 hash 值、使用签名数据对该 hash 进

行签名校验。如果验证不通过证明 APK 非法，验证通过则从签名信息主体提取原始 APK 的 hash 得到 HASH2;

6. 计算原始 APK 的 hash 值得到 HASH1，比较 HASH1、HASH2 是否一致，一致则 APK 合法，不一致则 APK 非法；

7．上述流程全部 OK 则认为机构验签通过，否则认为机构验签失败。

3.4. 签名数据格式

新签名方案采取谷歌签名方案 V2 形式存储签名数据，签名数据按 DER 格式编码。采用非对称加密算法，证书采用 x.509 格式。

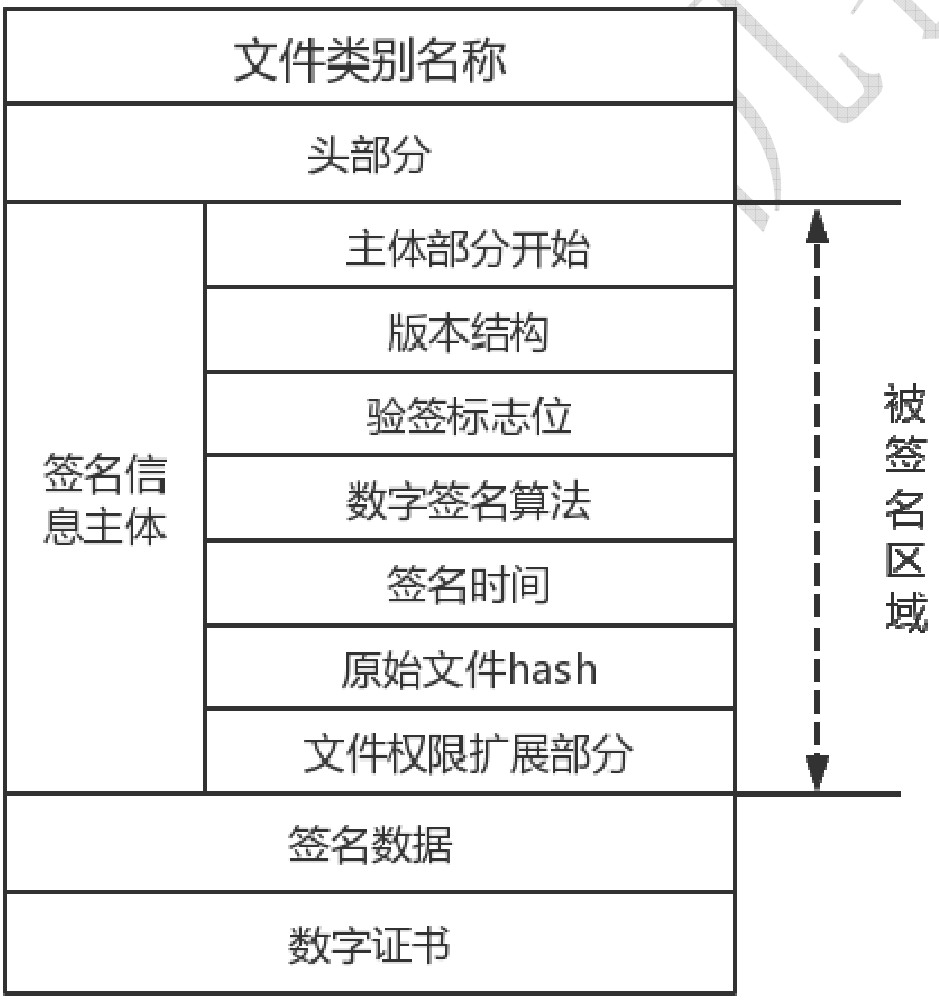


图 2.4.1 机构签名信息格式

3.5. 机构签名信息

区域			内容（hex）	描述
文件类别名			13 11 41 43 51 55 49 52 45 52 2d 53 47 4e 2d 49 4e 46 4f	PrintableString 类型(13)，长度 17(11)，” ACQUIRER-SGN-INFO”
头部分			30 82 04 a4	SEQUENCE 类型(30)，长度为 1188（04 a4
签名信息主体	主体部分开始		30 59	SEQUNSE 类型(30)，长度 89(59)
	结构版本		02 01 01	整数类型(02),长度 1(01)，目前固定为 0x01
	验签标志位		01 01 00	INTEGER 类型(02),长度 1(01),默认 false(00)
	数字签名算法		06 09 2a 86 48 86 f7 0d 01 01 0b	OBJECT IDENTIFIER 类型(06)，长度为 9(09)，签名算法为 SHA-256 和 RSA OID 1.2.840.113549.1.1.11: rsa-with-sha256
	签名时间		13 10 32 30 31 37 2d 31 32 2d 32 30 20 31 32 3a 30 30	PrintableString 类型(13)，长度 16(10)，” YYYY-MM-DD hh:mm
	原始文件 hash		02 20 xx xx xx xx … xx	整数类型(02),长度 32(20)，若生成的哈希长度超过 32 字节，则只保存 前面 32 字节
	文件权限扩展		a3 xx 30 xx	扩展部分(a3),长度为 xx()，SEQUENCE 类型 (30),长度 xx()
		权限描述文件	30 xx	SEQUENCE 类型(30),长度 xx()
			13 0e 45 50 41 59 2d 49 4c 45 2d 44 45 53 43	PrintableString 类型(13)，长度 14(0e)，”EPAY-FILE-DESC”，权限描述文件标识
			02 XX xx xx xx xx	整数类型(02),长度 XX()，权限描述文件数据
签名数据			02 82 01 00 xx xx xx xx … xx	整数类型(02)，长度 256(01 00)
数字证书			03 82 03 46 00 xx xx xx xx … xx	BIT STRING 类型(03)，长度为 838（03 46）， 填充 0 比特位数为 0(00)，机构工作公钥证 书

表 2.5.1 签名结构

3.6. 验签标志与权限扩展项

客户在使用签名系统签名应用时，可以选择关闭应用升级验签选项。这样签名系统会设置应用的验签标志位，当系统发现应用是升级安装时（系统已经安装对应包名的应用，且应用的签名相同），则跳过对应用的验签动作直接进行应用安装。如此可以节省应用升级安装的时间。验签选项默认为升级过程也需要验

签。

为了加强系统对应用的安全权限管控，防止任意应用对密码键盘、打印机等接口的随意调用。同时通过签名加密的方式对应用申请的权限项进行保护，防止恶意篡改。在使用签名系统给应用签名前，要求客户根据应用需求填写权限文件。签名系统会将权限描述项添加到签名信息主体上。应用在调用到对应的接口时，终端会判断应用是否有权限调用相应接口。

权限文件格式为 txt 文本格式，权限文件内容仅为应用需要申请的权限项，每行一条。可申请权限列表如下：

权限	说明
android.permission.SAFE_MODULE	访问证书管理与加密运算模块权限
android.permission.MSR	访问磁条卡读卡器设备权限
android.permission.SMARTCARD	访问接触式 IC 卡阅读器权限
android.permission.CONTACTLESS_CARD	访问非接触 IC 卡读卡设备权限
android.permission.PRINTER	访问打印机设备权限
android.permission.PINPAD	访问 PIN 输入设备权限
android.permission.PIN_GET_PIN_BLOCK	PIN 输入设备计算 Pinblock 权限
android.permission.PIN_MAC	PIN 输入设备计算 Mac 权限
android.permission.PIN_ENCRYPT_DATA	PIN 输入设备加密数据权限
android.permission.PIN_UPDATE_MASTER_KEY	PIN 输入设备更新终端主密钥权限
android.permission.PIN_UPDATE_USER_KEY	PIN 输入设备更新用户密钥权限
android.permission.SERIAL	访问串口设备权限
android.permission.LED	控制 Led 灯设备权限
android.permission.EMV	访问 EMV 卡权限

表 2.6.1 权限说明

4.机构签名文件保存方式

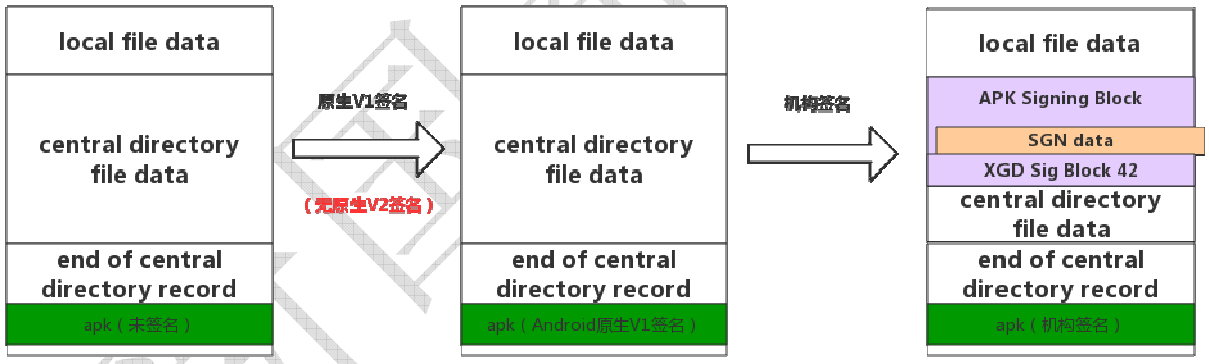
通过对 APK 原生签名格式的分析后，发现把**机构签名数据**添加到 APK Signing Block 中，既可以保存机构签名信息，又可以**保证所添加的签名信息** 不会影响 **原 APK 的安装流程**的目的。

本签名方案是直接对整个经过压缩后的 APK 文件（APK 文件本身已经经过了 Android 原生签名）进行签名，将生成的签名数据 SGN 插入到压缩的文件内容源数据与目录源数据之间的 APK 签名块中。对 Android 原生验签并不影响。

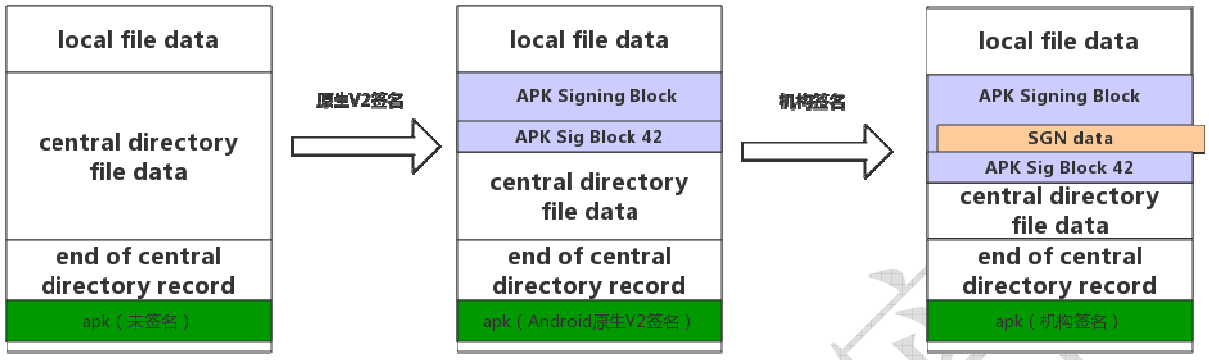
4.1. Apk 签名方式

1. 应用未签过原生 V2 签名，则以“XGD Sig Block 42”为魔数生成签名块。
2. 应用签过原生 V2 签名，则将机构签名信息数据插入到原生签名数据之后

4.1.1. 只通过原生 V1 签名的 apk



4.1.2. 经过原生 V2 签名的 apk



4.2. APK 签名块格式

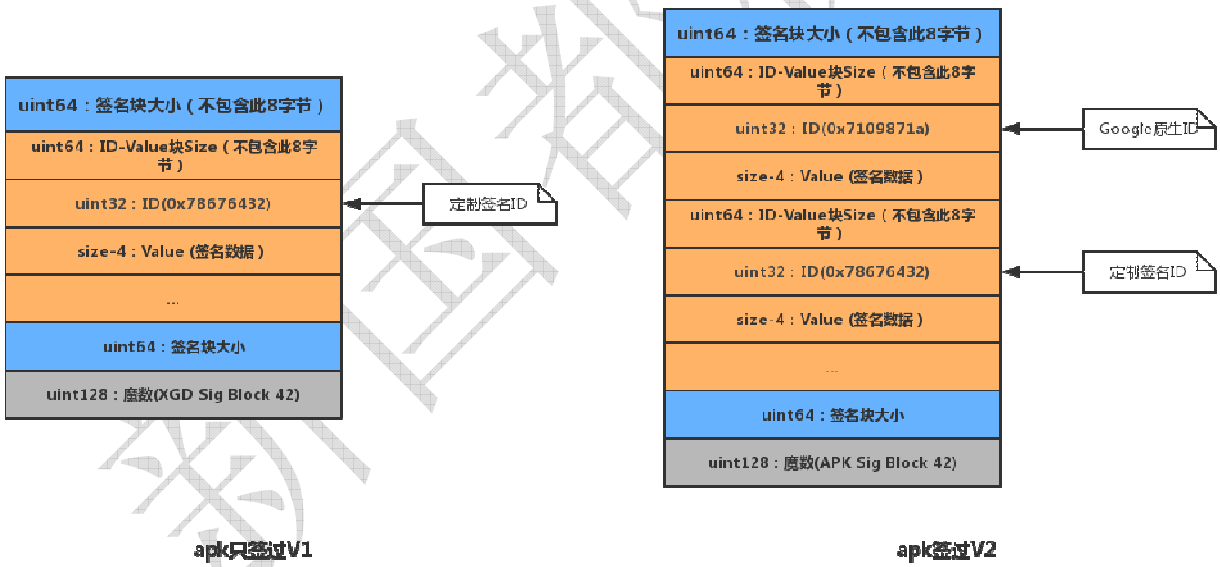


图 4.1 APK 签名块格式

整个签名块的结构为：

首先是 8 字节的签名块大小，此大小不包含该字段本身的 8 字节；

接下来是 1 个以上的如下数据字段：

{

8 字节的 ID-Value 块大小，此大小不包含改字段本身的 8 字节；

V2 模式块，也就是 ID-Value 序列，包括一个 4 字节的 ID 和对应的数据；

}

然后又是一个 8 字节的签名块大小，与开始的 8 字节是相等的；

最后是固定的 16 字节的签名块魔数如果应用签过原生 V2 签名则魔数值为”APK Sig Block 42”，否则魔

数值为”XGD Sig Block 42”；

4.2.1. 只通过原生 V1 签名的 apk

域			字节数	内容	描述
Block Len			8	xx … xx	小端存储，签名块长度，(最大 0x7fffffff-8)
	自定义 ID-value	长度	8	xx … xx	小端存储，自定义 ID-value 的长度
		标识	4	自定义	小端存储，用于标识自定义数据
		value data	n1	xx … xx	自定义数据
	…				
Block Len			8	xx … xx	小端存储，签名块长度，与第一个 Block Len 的值相同，(最大 0x7fffffff-8)
Magic			16	58 47 44 20 53 69 67 20 42 6C 6F 63 6B 20 34 32	魔数，字符串“XGD Sig Block 42”

4.2.2. 经过原生 V2 签名的 apk

域			字节数	内容	描述
Block Len			8	xx … xx	小端存储，签名块长度，(最大 0x7fffffff-8)
Sign Data	原生的 ID-value	长度	8	xx … xx	小端存储，ID-value 的长度，包括标识+签名信息
		标识	4	1A 87 09 71	小端存储，0x7109871a
		value data	n1		原生签名数据
	自定义 ID-value	长度	8	xx … xx	小端存储，自定义 ID-value 的长度
		标识	4	自定义	小端存储，用于标识自定义数据
		value data	n2	xx … xx	自定义数据
	…				
Block Len			8	xx … xx	小端存储，签名块长度，与第一个 Block Len 的值相同，(最大 0x7fffffff-8)
Magic			16	41 50 4B 20 53 69 67 20 42 6C 6F 63 6B 20 34 32	魔数，字符串“APK Sig Block 42”

4.3. 结构图

图 3.1.1 V2 签名结构

签名块内容：

包括对 apk 第一部分，第二部分，第三部分的二进制内容做加密保护，摘要算法以及签名算法。签名块本身不做加密，这里需要特殊注意的是由于第三部分包含了对第二部分的引用偏移，因此如果签名块做了改变，比如在签名过程中增加一种签名算法，或者增加签名者等信息就会导致这个引用偏移发生改变，因此在算摘要的时候需要剔除这个因素要以第三部分对签名块的偏移来做计算。

5.方案对比

区别点	原方案	新方案
签名原理	利用 Android 原生签名体系，修改 算法为 RSA2048+SHA256 算法，管控证书来限制安装	在谷歌 V2 验签方案基础上实现，系统专用验签机制
签名文件	覆盖 Android 验签 V1 版签名文件	无签名文件，签名数据存储在签名块中兼容 V1、V2 方案
扩展性	签名文件覆盖形式，可扩展性差	预留字段为后续扩展留有余地
方便性	较方便，完全兼容原生安卓体系， 可直接使用 Android 签名工具签 名，校验。不需要另外开发签名工	兼容旧的签名。存量机升级方便

表 4.1 方案对比