相关介绍

**消息摘要 Message Digest**

消息摘要就是在消息数据上，执行一个单向的Hash函数，生成一个固定长度的Hash值，这个Hash值即是消息摘要也称为数字指纹.

消息摘要有以下特点：

1. 通过摘要无法推算得出消息本身

2. 如果修改了消息，那么摘要一定会变化（实际上，由于长明文生成短摘要的Hash必然会产生碰撞），所以这句话并不准确，我们可以改为：很难找到一种模式，修改了消息，而它的摘要不会变化。

消息摘要的这种特性，很适合来验证数据的完整性，比如在网络传输过程中下载一个大文件BigFile，我们会同时从网络下载BigFile和BigFile.md5，BigFile.md5保存BigFile的摘要，我们在本地生成BigFile的消息摘要，和BigFile.md5比较，如果内容相同，则表示下载过程正确。

注意，消息摘要只能保证消息的完整性，并不能保证消息的不可篡改性。

**MD5/SHA-0 SHA-1**

这些都是摘要生成算法，和签名没有关系。

**数字签名 - Signature**

数字签名就是信息的发送者用自己的私钥对消息摘要加密产生一个字符串，加密算法确保别人无法伪造生成这段字符串，这段数字串也是对信息的发送者发送信息真实性的一个有效证明。

数字签名是 非对称密钥加密技术 + 数字摘要技术 的结合。

数字签名技术是将信息摘要用发送者的私钥加密，与原文一起传送给接收者。接收者只有用发送者的公钥才能解密被加密的信息摘要，然后接收者用相同的Hash函数对收到的原文产生一个信息摘要，与解密的信息摘要做比对。如果相同，则说明收到的信息是完整的，在传输过程中没有被修改；不同则说明信息被修改过，因此数字签名能保证信息的完整性。并且由于只有发送者才有加密摘要的私钥，所以我们可以确定信息一定是发送者发送的。

**数字证书 - Certificate**

数字证书是一个经证书授权 中心数字签名的包含公开密钥拥有者信息以及公开密钥的文件。CERT.RSA包含了一个数字签名以及一个数字证书。

需要注意的是Android APK中的CERT.RSA证书是自签名的，并不需要这个证书是第三方权威机构发布或者认证的，用户可以在本地机器自行生成这个自签名证书。

签名文件列表:

├── META-INF

   ├── CERT.RSA/CERT#RSA CERT.SF签名和数字证书

   ├── CERT.SF/CERT#.SF MANIFEST摘要

    └── MANIFEST.MF 文件摘要

MANIFEST.MF:

对apk包中的每个文件(非文件夹/非签名), 生成SHA1的摘要信息, 再对这个信息进行Base64编码。

CERT.SF/CERT#.SF:

对整个MANIFEST进行SHA计算，并将每个字段再次计算SHA值。关键函数: writeSignatureFile

CERT.RSA/CERT#RSA:

使用私钥和加密算法方式生成signer信息，创建并将publicKey导入到证书，将signer和cert生成签名数据并编码为asn1格式。关键函数: writeSignatureBlock

# 重新签名命令如下:

java -jar signapk platform.x509.pem platform.pk8 Email.apk Email\_signed.apk

SignAPK签名过程:

相关代码: build/tools/signapk/SignApk.java

计算机生成了可选文字:
导 入 所 有 公 钥 文 件 
X509Certificate 
publicKey 
通 过 证 书 获 取 签 名 算 法 ． 
S 1w1 以 IFS 肖 > S 1 
SHA2ö6withRSA > S 256 
通 过 第 一 个 证 书 获 取 
时 间 at · m 匕 a 
（ 有 效 时 间 ） 
导 入 所 有 钥 
privateKey 
打 开 输 入 输 出 K 
〔 ～ 〕 参 数 
inputJar 
outputF ． I 
签 名 整 个 又 件 。 ile 
生 成 文 件 ． 
META INF.com/andro d/otacert 
writeSignatureBlock ： 生 成 CERT. RS 肖 又 件 
根 据 Manif 匕 列 表 拷 
贝 所 有 文 件 到 输 出 文 
件 ， 并 设 置 时 间 戳 
循 环 操 作 如 下 内 容 ， 次 数 为 公 钥 个 数 ： 
1. 使 用 M ff t 生 成 M ” if t 摘 要 又 件 
CERT. SF/CERT# SF 
2 ． 使 用 CERT. s ignedData, 公 钥 ， 
钥 生 成 证 书 文 件 (CERT. SF*Z 和 证 
书 ） ： CERT. RSA/CERT* RSA 
生 成 摘 要 信 芑 数 据 ： META 1 № / 、 uFES 上 MF 
1. 通 过 加 密 算 法 设 消 摘 要 引 擎 
2 ． 遍 历 每 个 文 件 计 算 其 s 值 （ 非 巨 录 非 签 名 文 件 ） 
3 ． 区 分 算 法 类 型 ： SHA 1 DIS 匕 或 者 s 256 Digest 
4 ． 将 s 值 进 行 Ba 64 编 码 
# MB: 又 件 摘 要 
Man.fest—Version: 1 0 
Created By: 1 ． 0 (Amdroi 引 
Ntme: res/drawable Xhdp i i c launcher png 
SHA 1 Dlgest: UCzYT6RE8m1 + z99 旧 In 1385Y 
Ntme: r es layout/mai n 11111 
SHA 1 Dlgest: fgQNY7tgjTjlAnfNFbPLS/XF/5E 
对 WIIFEST. MF 的 各 段 再 次 计 算 其 a 
值 （ 摘 要 ） ， 生 成 CERT. SF 文 件 。 
# CERT. SF/CERT# SF ： 
S 'gnature—Version: l. 0 
Create d By: 1. 0 (Android SignApk) 
2 ． 
3 ． 
4 ． 
4 ． 
创 建 数 字 证 书 ， 将 公 钥 添 加 其 中 。 
使 用 钥 创 建 si § n 匕 r 。 
使 用 公 钥 创 建 s f 亠 
将 CERT. SF 数 据 (slgData) 签 名 。 
将 签 名 数 据 编 码 为 肖 SNI 格 式 ， 
S 1 
Neme 
S 1 
Digest Manifest: NpGg9ChwLEOQ8CwHNOHjxA 、 rPCygz 
r es layout/folder header. 
Digest Manifest: mlls6W6zfGegD81rpfCK30wLnHw— 

签名验证过程:

相关代码: java/util/jar/JarVerifier.java 实现函数: JarVerifier. readCertificates

计算机生成了可选文字:
JarVerifier.veri行Certificate
读取拟入IFEST.llF数据
读取CERT.SF数据
读取CERT.RSA数据
JarUtils.verifysignature
.RSA数据解编
码ANSI
加密摘要
encryptedDigest
deaoid
daNajr吧
导入Man1feSt数据
ManifestReader
增加i55user证书链到
certificates
获取51,edData
数据
获取Signer信息
daoid
key:CERT.SF,value:
issuser证书链
从51,edData中
获取证书数据
encCerts[]
获取1SSuer
获取序列号
验证属性值
,,5ignature一Version刃
5nllm
获取签名引擎实例
SHAlwithRSA
是否由signtool创建
使用encCerts［口
创建X509证书
certs［二
定位证书
查找与I55uer匹配的
证书cert〔x二
初始化校验引擎
设置签名证书。erts〔x二
51,ature.init介rify
比对．SFSHAI一Digest一Manifest一Main-
Attribute：属性值与Manifest的消息摘
值是否相同
JarVerifier.verify.
L_____
拷贝生成有效
的．SF签名数据
sfBvtes
设置，SF签名数据
51,ature.update
摘要信息校验
校验
5ignature.verify
遍历比对．SFSHAI一Digest属性值与
Manifest各段的消息摘值是否相同
JarVerifier.verifs.
创建iSSuser证书链
createChain
Done