iOS 应用签名机制

对称加密和非对称加密

常用的加密算法有 DES、3DES、AES 和 RSA。

对称加密：加密和解密使用同一个密钥。

非对称加密：分为公钥和私钥，公钥一般可以公开，公钥加密需要私钥解密，私钥加密需要公钥解密。

DES：Data Encryption Standard，数据加密标准。是一种将 64bit 数据（明文）加密成 64bit 密文的对称加密算法，密钥长度为 64bit，但是每隔 7bit 会设置一个错误检查的 bit，因此实际密钥长度为 56bit。每次只能加密 64bit 的数据，大数据需要进行反复迭代。目前已经可以在短时间内被破解，并不建议使用。

3DES：加密方式与 DES 相同，可以设置三个密钥，重复对数据进行加密，安全性稍大于 DES。

AES：Advanced Encryption Standard，高级加密标准。密钥长度有 128、192、256bit 三种，安全性优于 DES，目前已经逐步取代 DES 成功首选对称加密算法。

RSA：由罗纳德·李维斯特（Ron Rivest）、阿迪·萨莫尔（Adi Shamir）和伦纳德·阿德曼（Leonard Adleman）在 1977 年提出，由三者的姓氏首字母命名，为目前最常用的非对称加密算法。

单项散列函数

根据任意长度数据计算出固定长度的散列值，消息不同散列值也不同，散列具有单向性。又称为消息摘要函数、哈希函数。

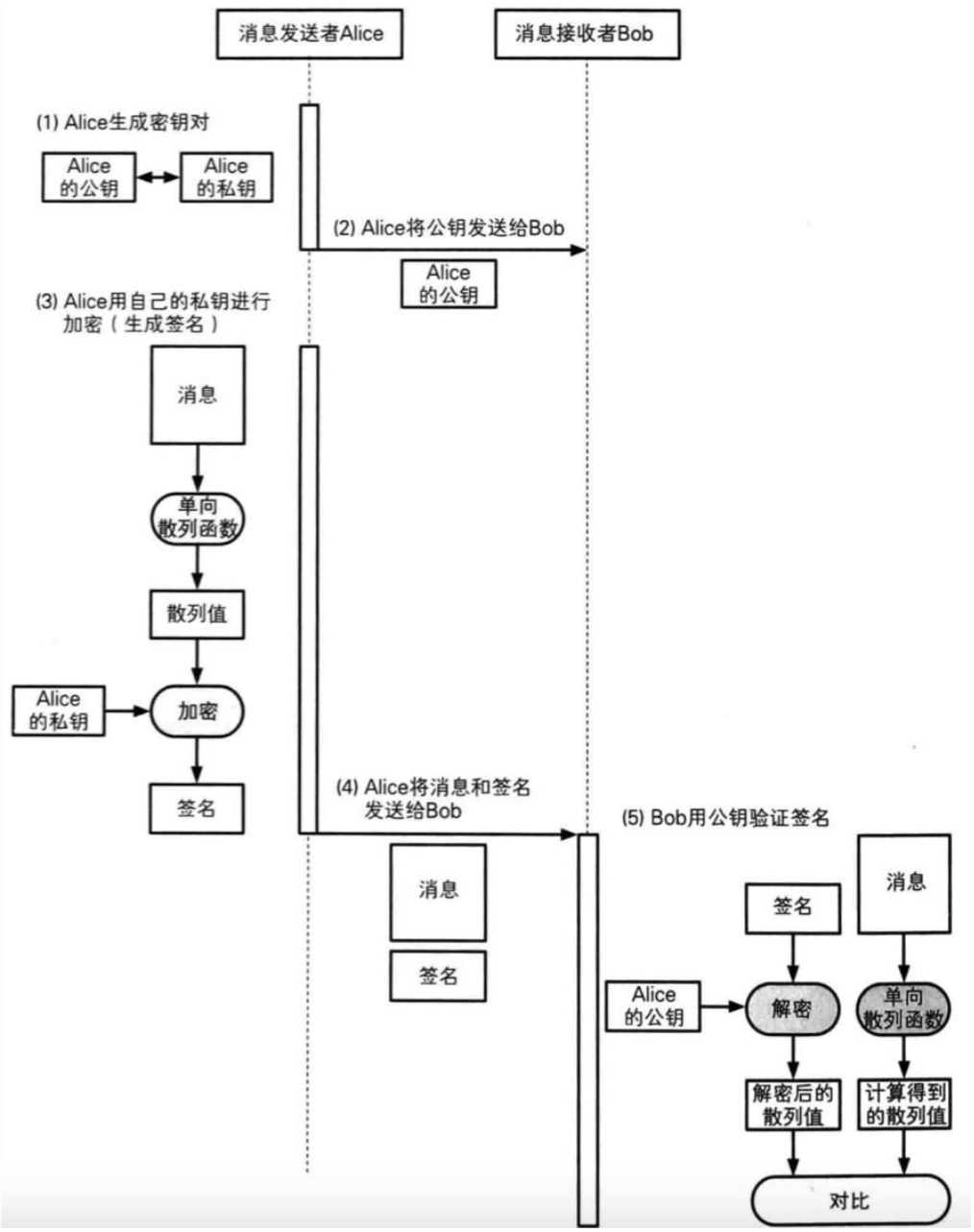
常见的散列函数有 MD4/MD5、SHA-1、SHA-2、SHA-3。

MD4/MD5：Message Digest，信息摘要，可以产生 128bit 的散列值。

SHA：Secure Hash Algorithm，密码散列算法，SHA-1、SHA-2、SHA-3分别为第一代到第三代标准。

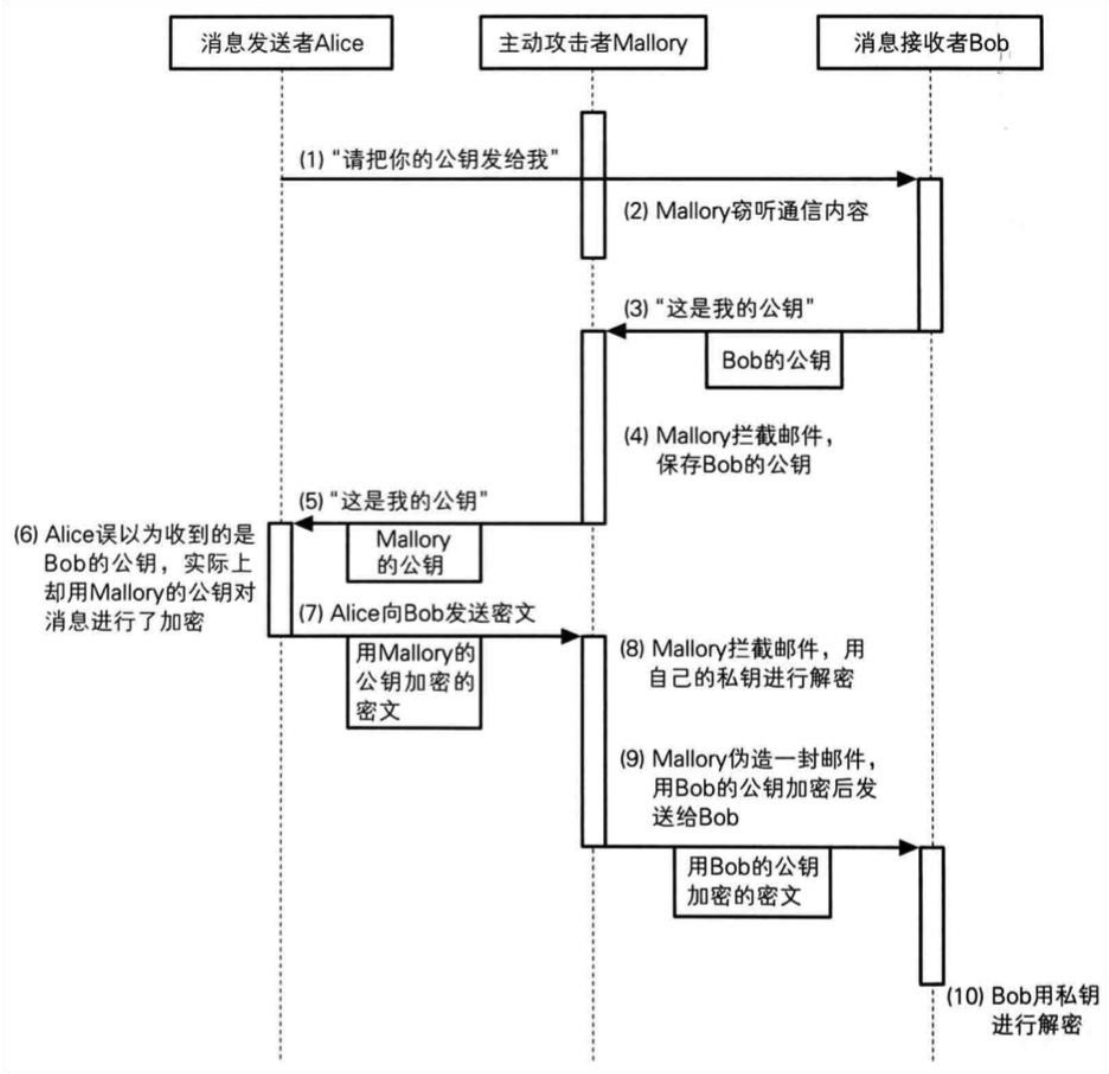
数字签名

Digital Signature，根据传输数据生成一段数据标识，用于防止**数据被篡改**，一般使用单项散列函数获取传输数据的摘要，然后使用 RSA 私钥对摘要进行加密，生成“数字签名”。消息接收者在收到数据时手动计算数据的摘要，并与使用公钥解密的“数字签名”进行对比，如果一致则证书数据未被篡改。



因为 RSA 公钥私钥成对出现，私钥加密只有公钥能解密，且 RSA 私钥不能公开，如 A 的公钥能解密 A 私钥加密的信息则证明该信息为 A 发出，故 RSA 加密的信息可以被当作“数字签名”。

如受到中间人攻击，公钥被伪造时数字签名将不安全，故验证签名之前需要验证公钥的合法性。

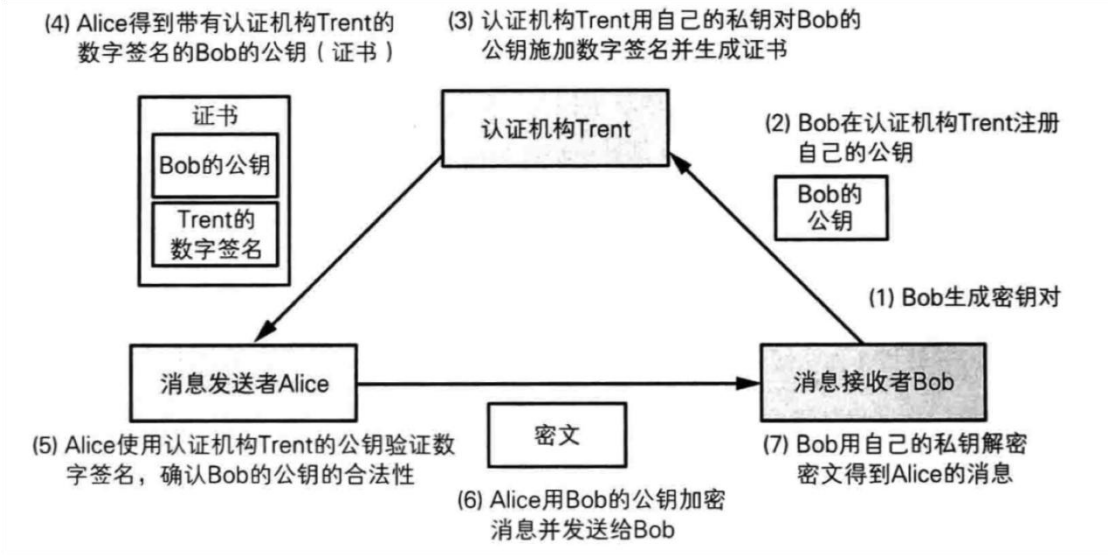


证书

Certificate，全称为公钥证书（Public-key Certificate，PKC），包含申请人的个人信息（如姓名、邮箱）、证书信息（过期时间、签发者）及申请人公钥，并由认证机构（Certificate Authority，CA）施加数字签名的文件。

CA 就是能够认定“公钥确实属于此人”并能够生成数字签名的个人或组织，

如国际性、政府设立组织、提供认证服务的企业或个人，如苹果提供应用相关证书、https 证书。



根证书

根证书是 CA 认证中心给自己颁发的证书，是证书链的起点，安装根证书意味着对这个 CA 认证中心信任。

验证证书的真伪需要 CA 中心的公钥验证，而 CA 中的公钥存在于对这份证书进行签名的证书内，故需要下载该证书并验证真伪，验证证书又需要签发该证书的证书来验证，形成一条证书链。证书链的终结就是根证书，跟证书的签发者为本身，下载根证书表明对根证书一下所签发的证书都信任，证书的验证追溯至根证书结束。用户在使用数字证书之前必须先下载根证书。

证书标准和编码格式

证书标准主要定义了证书中应该包含哪些内容，CA 中心普遍使用标准为 X.509 系列和 PKCS 系列。

X.509：由国际电信联盟（ITU-T）制定的数字证书标准。包括版本号、证书序列号、CA 标识符、签名算法标识、签发者名称、证书有效期等信息。

PKCS： Public-Key Cryptography Standards，由美国 RSA 数据安全公司及其合作伙伴制定的一组公钥密码学标准。包括 PKCS#1、PKCS#3、PKCS#5-15 十三个标准。

常见的证书编码格式有 DER、PEM、PKCS10、PKCS12 等，其中 DER 和 PEM 同属 X.509 标准。

DER：使用二进制编码，可使用 openssl 将 DER 编码文件转换为 pem 文件，扩展名有.der、.cer、.cert、.crt、.rsa。

PEM：Printable Encoded Message，一般基于 base64 编码，可能只包含公钥，也可能包含完整的证书链（包括公钥、私钥和跟证书），也可能用来编码 CSR 文件，扩展名有.pem、.csr。

PKCS10：Certificate Signing Request，证书签名请求文件，只包含公钥，扩展名有.p10、.csr、.certSigningRequest。

PKCS12：个人信息交换语法标准，包含公钥、私钥及其证书，扩展名有.p12、.pfx。

Tips：.cer、.cert、.crt 一般为 DER 编码，但也有可能是 PEM 编码，可使用 openssl 工具查看编码方式。

Apple 根证书

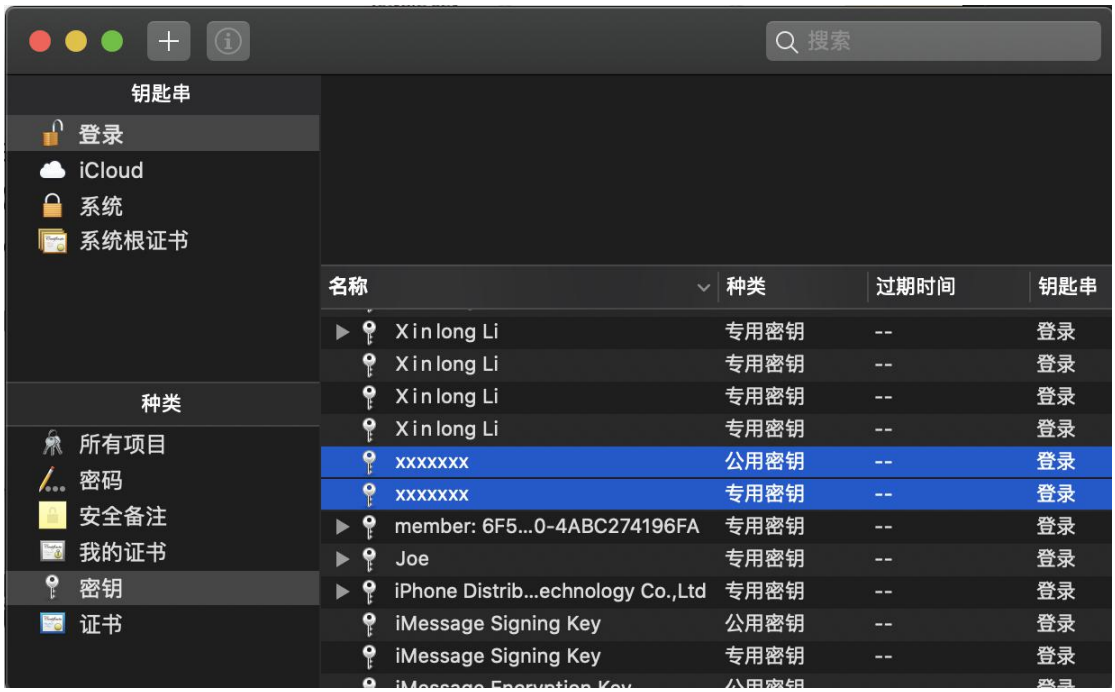
在安装 Xcode 时将自动安装 Apple Worldwide Developer Relations Certification Authority 的根证书。

Apple Worldwide Developer Relations Certification Authority
中级证书颁发机构
过期时间：2023年2月8日 星期三 中国标准时间 上午5:48:47
此证书有效

从 Member Center 下载的证书均由该证书验证。

iOS 开发证书申请

mac 上申请证书需要打开钥匙串选择“从证书颁发机构请求证书...”，输入用户电子邮件地址及常用名称后点击继续，将在钥匙串访问中生成一对非对称加密密钥对 Public/Pirvate Key Pair（公钥和私钥）和一个包含开发者身份信息和公钥的 CSR（Certificate Signing Request）文件.certSigningRequest。



随后在 Member Center 中上传 CSR 文件，Apple 证书颁发机构使用其私钥对 CSR 中的公钥及其中的身份信息进行加密签名形成数字证书并记录起来。证书文件后缀为.cer，其中只包含公钥信息。下载后双击打开，系统会根据 CER 中的公钥信息自动匹配私钥信息，如果匹配到即可加入钥匙串访问中。因为 XCode 需要使用私钥进行数据签名，如钥匙串访问中不存在对应的私钥，则会添加失败（如直接将 CER 文件提供给团队其它人使用）。

如需向团队其它人提供证书可将私钥导出个人交换信息 p12 文件和 cer 文件一起交给开发人员，此时 p12 文件中只包含公钥私钥信息。

或者将 cer 加到钥匙串后导出 p12 交给其它开发者，此时 p12 中包含公钥私钥及证书信息。

iOS 证书类型

常见的 iOS 开发证书类型有如下这些：

iOS App Development：开发、真机调试用

App Store and Ad Hoc：上架和 AdHoc 方式发布时用

In-House：企业发布证书，该证书打包的应用可随意安装在设备上。

Apple Push Notification service SSL (Sandbox)：开发阶段使用苹果的推送服务

Apple Push Notification service SSL (Production)：上架后使用苹果推送服务

AuthKey_****.p8：授权 p8 证书，一般用于账号下的应用推送，只包含私钥。

iOS 授权描述文件

证书能够证明 app 所属及 app 的完整性，保证 app 本身是安全的。但是证书中的信息是有限的, 不能细化到每个应用中的某些服务是 Apple 认可的, 比如：APNS 推送服务、设备是否为测试设备等，所以需要额外的文件

——mobileprovision。

常用的描述文件有如下四种：

1、App Development：开发描述文件

2、Ad Hoc：内部测试描述文件

3、App Store：发布 App Store 描述文件

4、In House Distribution：企业级开发者账号的发布描述文件

使用命令可以查看 mobileprovision 中的信息：

```
security cms –D –i embedded.mobileprovision
```

也可以在 Mac 设备上点击文件直接展示其简要信息。

其中包含如下信息：

1、应用信息：包括 App ID、App Name、Platform

2、组织（公司）信息：组织名称、组织 ID

3、文件信息：UUID 唯一标识、创建和过期时间

4、功能授权列表：比如开启 APNS 推送服务、Group 等

5、关联证书：不同证书代表不同的发布方式。

6、授权设备列表：包含授权设备的 UDID

7、Apple 签名：使用 Apple 私钥的签名信息，保证文件只能从 Apple 获取，获取后不能进行篡改，保证所有规则都必须由苹果制定和约束。

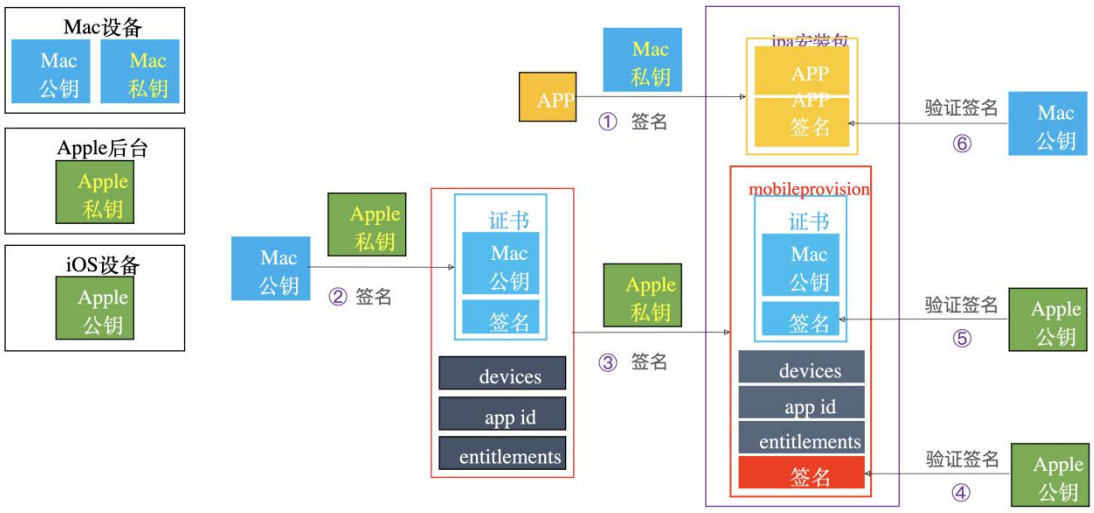
四种描述文件大体相同，只有细微差别：

授权设备区别：其中 App Development 和 Ad Hoc 需要关联授权设备，即允许安装的设备, Apple 规定每个账号每种设备只允许加载 100 台, 如 iPhone100 台、iPad100 台等。App Store 中无授权设备信息，因为需要提交至 Apple 审核不能直接安装至设备中。In House Distribution 中也没有授权设备信息，但是其中包含 Provisions All Devices 的字段且为 Yes，故应用可以直接安装在任意设备上。

关联证书区别：App Development 可以关联多个开发证书，其余三种只允许关联一个发布证书。

允许调试：App Development 描述文件中 get-task-allow，字段为 true，证明该文件为真机调试描述文件，允许开发调试，其余三种字段为 false，只允许安装。

iOS 签名机制



当 Mac 设备申请下载证书或从它处获取 p12 文件并安装在钥匙串访问中后，Mac 公钥私钥及证书即安装在设备中。Apple 后台保存 Apple 生成的私钥，公钥内置在每台 iOS 设备中。

1、使用 XCode 编译或打包时，XCode 将使用 codesign 工具及 Mac 中的私钥对代码的 Mach-O 文件进行签名并保存在 Mach-O 文件中。

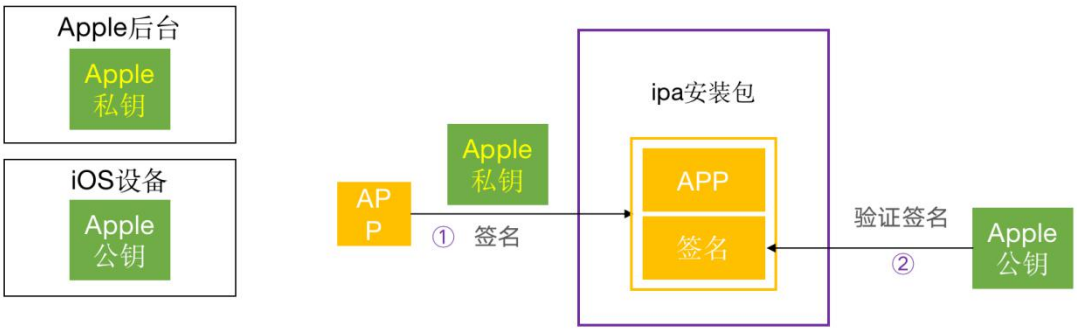
2、在 Member Center 生成证书时会使用 Apple 私钥对 Mac 公钥及申请人信息进行签名形成证书

3、在生成描述文件时将证书、授权设备、应用及组织等信息保存起来并对其使用 Apple 私钥签名。应用编译或打包时将描述文件内置在应用 Zip 包中。

4、安装应用时首先使用内置的 Apple 公钥对描述文件的签名进行验证，防止描述文件被篡改。

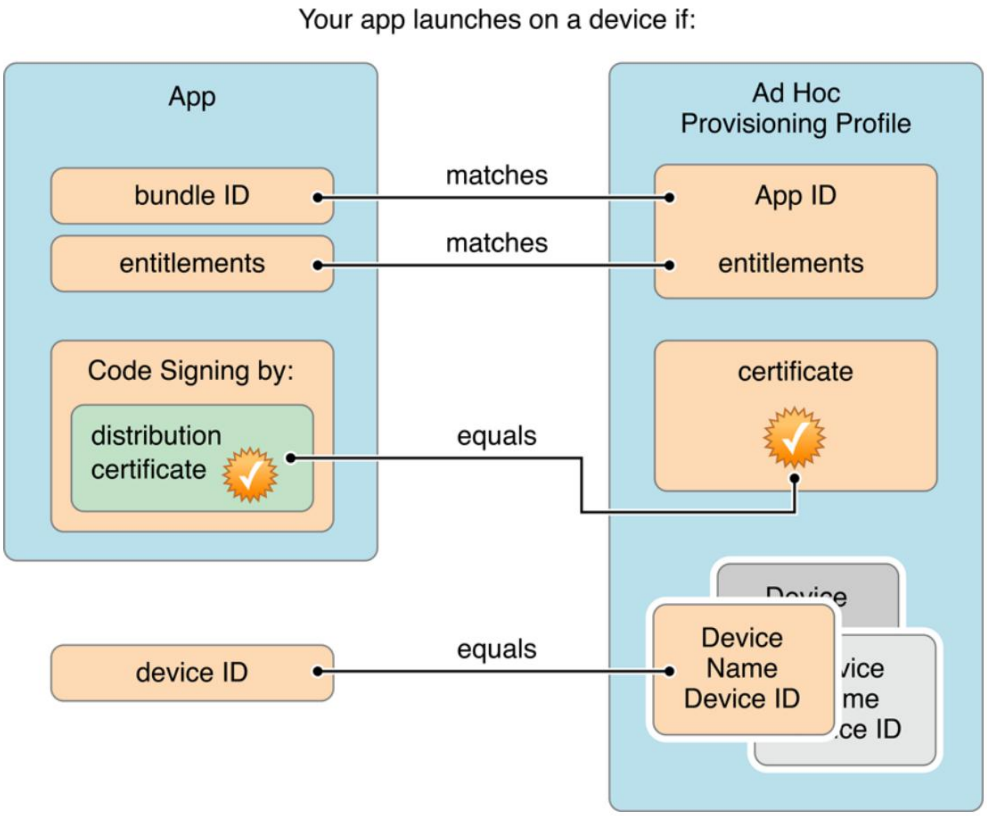
5、随后使用 Apple 公钥对其中的证书进行校验，防止证书公钥被篡改。

6、以上两个签名验证都通过证明证书中的公钥是 Apple 审核通过的，随后使用证书中的公钥对 Mach-O 文件进行签名验证，如果验证通过则证明所有流程合规则允许安装。



Apple 后台审核通过后，将直接使用 Apple 私钥对应用包进行重签名，应用安装后只需校验 Mach-O 文件未被篡改即可，故 Apple Store 下载的应用包中不包含 mobileprovisions 文件。

iOS 应用启动校验



非 AppStore 下载应用启动时还会对描述文件中的信息进行校验，将 Mach-O 文件中的信息于描述文件中的信息进行对比，对 App ID、entitlements、certificate 及 device 进行对比，均通过后才会进行启动。

iOS 应用重签名

从 App Store 下载的应用经过了 Apple 的加壳操作，如果对 App Store 下载的应用进行重签名需要对其进行脱壳处理。

加壳：利用特殊的算法，对可执行文件的编码进行改变（比如压缩、加密），以达到保护程序代码的目的。

脱壳（砸壳）：摘掉壳程序，将未加密的可执行文件还原出来。

重签名步骤:

1、embedded.mobileprovision 文件，并放入.app 包中。

2、从 embedded.mobileprovision 文件中提取出 entitlements.plist 权限文件

```
security cms –D –i embedded.mobileprovisionsecurity cms –D –i embedded.mobileprovision > temp.plist
/usr/libexec/PlistBuddy –x –c 'Print :Entitlements' temp.plist > entitlements.plist
```

3、查看可用证书

```
security find–identity –v –p codesiging
```

4、对.app 内部的动态库、AppExtension 等进行签名

```
codesign –fs“证书 ID”“xxx.dylib”
```

5、对.app 包进行签名

```
Codesign –fs“证书 ID”––entitlements enttlements.plist“xxx.app”
```

签名完成重启压缩生成.ipa 即可安装在 iOS 设备上。也可以使用重签名工具，如 iOS App Signer、iReSign 等。