**引言**

        关于开发证书配置（Certificates & Identifiers & Provisioning Profiles），相信做iOS开发的同学没少被折腾。对于一个iOS开发小白、半吊子（比如像我自己）抑或老兵，或多或少会有或曾有过以下不详、疑问、疑惑甚至困惑：

1. 什么是App ID？Explicit/Wildcard App ID有何区别？什么是App Group ID？
2. 什么是证书（Certificate）？如何申请？有啥用？
3. 什么是Key Pair（公钥/私钥）？有啥用？与证书有何关联？
4. 什么是签名（Signature）？如何签名（CodeSign）？怎样校验（Verify）？
5. 什么是（Team）Provisioning Profiles？有啥用？
6. Xcode如何配置才能使用iOS真机进行开发调试？
7. 多台机器如何共享开发者账号或证书？
8. 遇到证书配置问题怎么办？

本文将围绕相关概念做个系统的梳理串烧。

**写在前面**

1.假设你使用过Apple设备（iMac/iPad/iPhone）且**注册**过Apple ID（Apple Account）。

2.假设你或你所在的开发组已加入苹果开发者计划（Enroll in iOS Developer Program to become a[member](https://developer.apple.com/membercenter/index.action" \t "_blank)），即已**注册**开发者账号（Apple Developer Account）。

* 只有拥有开发者账号，才可以申请开发/发布证书及相关配置授权文件，进而在iOS真机上开发调试Apps或发布到App Store。
* 开发者账号分为Individual和Company/Organization两种类型。如无特别交代，下文基于$99/Year的普通个人开发者（Individual）账号展开。

3.若要真机调试实践，你必须至少拥有一台装有Mac OS X/Xcode的Mac开发机（iMac or MacBook），其上自带原生的Keychain Access。

**一.App ID（bundle identifier）**

App ID即Product ID，用于标识一个或者一组App。

App ID应该和Xcode中的Bundle Identifier是一致（Explicit）的或匹配（Wildcard）的。

App ID字符串通常以**反域名**（reverse-domain-name）格式的Company Identifier（Company ID）作为前缀（Prefix/Seed），一般不超过255个ASCII字符。

App ID全名会被追加Application Identifier Prefix（一般为TeamID.），分为两类：

* Explicit App ID：唯一的App ID，用于唯一标识一个应用程序。例如“com.apple.garageband”这个App ID，用于标识Bundle Identifier为“com.apple.garageband”的App。
* Wildcard App ID：含有通配符的App ID，用于标识一组应用程序。例如“\*”（实际上是Application Identifier Prefix）表示所有应用程序；而“com.apple.\*”可以表示Bundle Identifier以“com.apple.”开头（苹果公司）的所有应用程序。

用户可在Developer MemberCenter网站上注册（Register）或删除（Delete）已注册的App IDs。

App ID**被配置到**【XcodeTarget|Info|Bundle Identifier】下；对于Wildcard App ID，只要bundle identifier包含其作为Prefix/Seed即可。

**二.设备（Device）**

Device就是运行iOS系统用于开发调试App的设备。每台Apple设备使用**UDID**来唯一标识。

iOS设备连接Mac后，可通过iTunes->Summary或者Xcode->Window->Devices获取iPhone的UDID（identifier）。

Apple Member Center网站个人账号下的**Devices**中包含了注册过的所有可用于开发和测试的设备，普通个人开发账号每年累计最多只能注册**100**个设备。

* Apps signed by you or your team run only on designated development devices.
* Apps run only on the test devices you specify.

用户可在网站上注册或启用/禁用（Enable/Disable）已注册的Device。

本文的Devices是指**连接到**Xcode被授权用于开发测试的iOS设备（iPhone/iPad）。

**三.开发证书（Certificates）**

**1.证书的概念**

**证书**是由公证处或认证机关开具的证明资格或权力的*证件*，它是表明（或帮助断定）事理的一个*凭证*。证件或凭证的尾部通常会烙印*公章*。

每个中国人一生可能需要70多个证件，含15种身份证明。证件中“必需的”有30到40个。将这些证件按时间顺序铺开，那就是一个天朝子民的一生——持**准生证**许可落地，以**户籍证明**入籍，以**身份证**认证身份，持**结婚证**以合法同居，最终以**死亡证**明注销。

**2.数字证书的概念**

***数字证书***就是互联网通讯中**标志**通讯各方***身份信息***的一串数字，提供了一种在Internet上验证通信**实体身份**的方式，其作用类似于司机的驾驶执照或日常生活中的身份证。它是由一个由权威机构——**CA机构**，又称为证书授权中心（Certificate Authority）发行的，人们可以在网上用它来识别对方的身份。

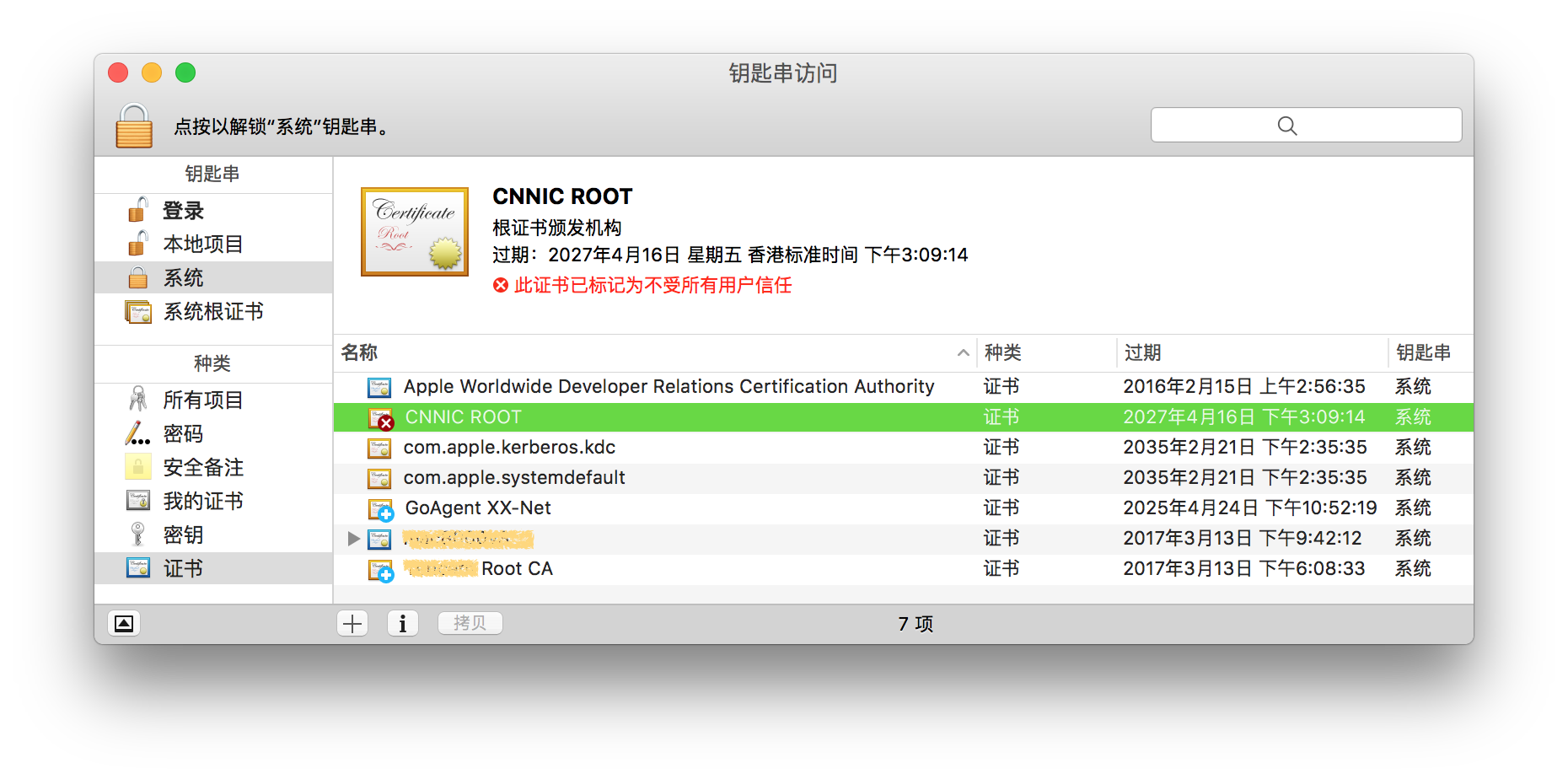
* 数字证书是一个经证书授权中心数字签名的包含公开密钥拥有者信息以及公开密钥的文件。最简单的证书包含一个公开密钥、名称以及证书授权中心的数字签名。
* 数字证书还有一个重要的特征就是时效性：只在特定的时间段内有效。

数字证书中的公开密钥（公钥）相当于公章。

某一认证领域内的**根证书**是CA认证中心给自己颁发的证书，是信任链的**起始点**。安装根证书意味着对这个CA认证中心的信任。

为了防止[GFW](http://bbs.3dmgame.com/thread-2356754-1-1.html" \t "_blank)进行中间人攻击(MitM)，例如篡改[github](http://program-think.blogspot.com/2015/03/weekly-share-82.html" \t "_blank)证书，导致无法访问github网站等问题，可选择不信任[CNNIC](http://bbs.kechuang.org/read/69655/1" \t "_blank)：

* 在［钥匙串-系统］中双击[CNNIC ROOT](http://www.williamlong.info/archives/4192.html" \t "_blank)，在【信任】|【使用此证书时】下拉选择【永不信任】。



在天朝子民的一生中，户籍证明可理解为等效的根证书：有了户籍证明，才能办理身份证；有了上流的身份证，才能办理下游居住证、结婚证、计划生育证、驾驶执照等认证。

**3.iOS（开发）证书**

iOS证书是用来证明iOS App内容（executable code）的合法性和完整性的**数字证书**。对于想安装到真机或发布到AppStore的应用程序（App），只有经过**签名验证**（Signature Validated）才能确保来源可信，并且保证App内容是完整、未经篡改的。

iOS证书分为两类：Development和Production（Distribution）。

* Development证书用来开发和调试应用程序：A ***development certificate*** identifies you, as a team member, in a development provisioning profile that allows apps signed by you to ***launch***on devices.
* Production主要用来分发应用程序（根据证书种类有不同作用）：A ***distribution certificate*** identifies your team or organization in a distribution provisioning profile and allows you to ***submit***your app to the store. Only a team agent or an admin can create a distribution certificate.

普通个人开发账号最多可注册iOS Development/Distribution证书各2个，用户可在网站上删除（Revoke）已注册的Certificate。

下文主要针对iOS App开发调试过程中的开发证书（Certificate for Development）。

**4.iOS（开发）证书的根证书**

那么，iOS开发证书是谁颁发的呢？或者说我们是从哪个CA申请到用于Xcode开发调试App的证书呢？

iOS以及Mac OS X系统（在安装Xcode时）将自动安装***[AppleWWDRCA.cer](https://developer.apple.com/certificationauthority/AppleWWDRCA.cer" \t "_blank)***这个中间证书（**Intermediate Certificates），**它实际上就是iOS（开发）证书的证书，即**根证书**（Apple Root Certificate）。

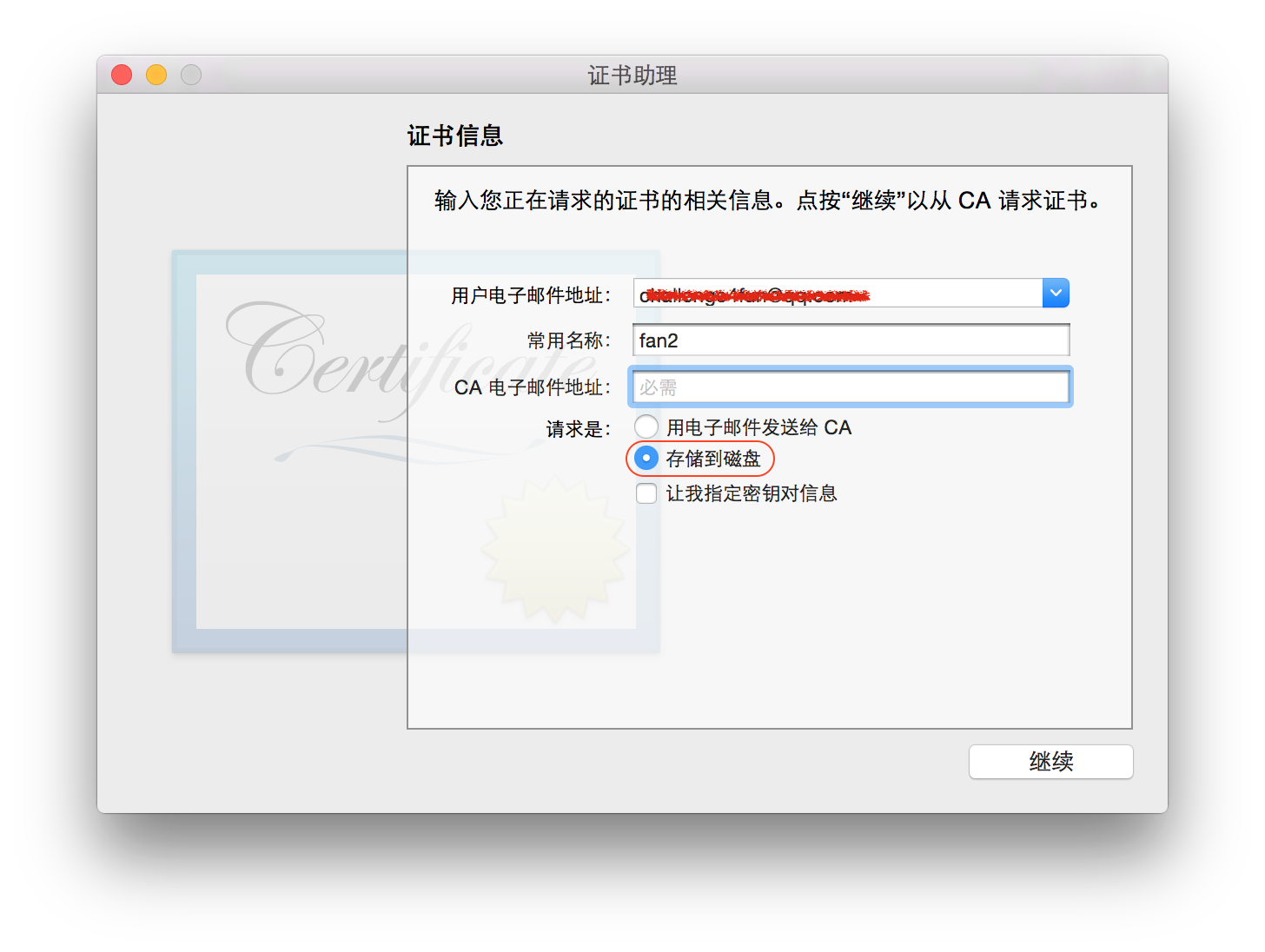
AppleWWDRCA（Apple Root CA）类似注册管理户籍的公安机关户政管理机构，AppleWWDRCA.cer之于iOS（开发）证书则好比户籍证之于身份证。

如果Mac Keychain Access证书助理在申请证书时尚未安装过该证书，请先下载安装（Signing requires that you have both the signing identity and the intermediate certificate installed in your keychain）。

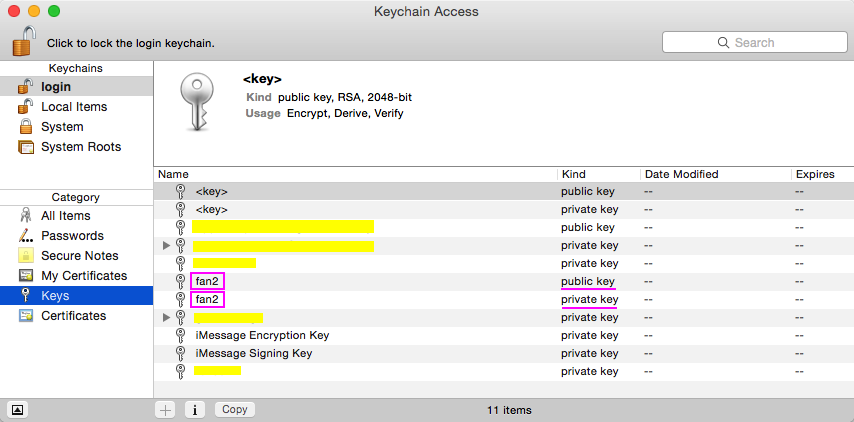


**5.申请证书（CSR：Certificate Signing Request）**

可以在缺少证书时通过Xcode Fix Issue自动请求证书，这里通过Keychain**证书助理**从证书颁发机构请求证书：填写开发账号邮件和常用名称，勾选【存储到磁盘】。



keychain将生成一个包含开发者身份信息的**CSR**（Certificate Signing Request）文件；同时，Keychain Access|Keys中将新增一对Public/Private **Key Pair**（This signing identity consists of a public-private key pair that Apple issues）。



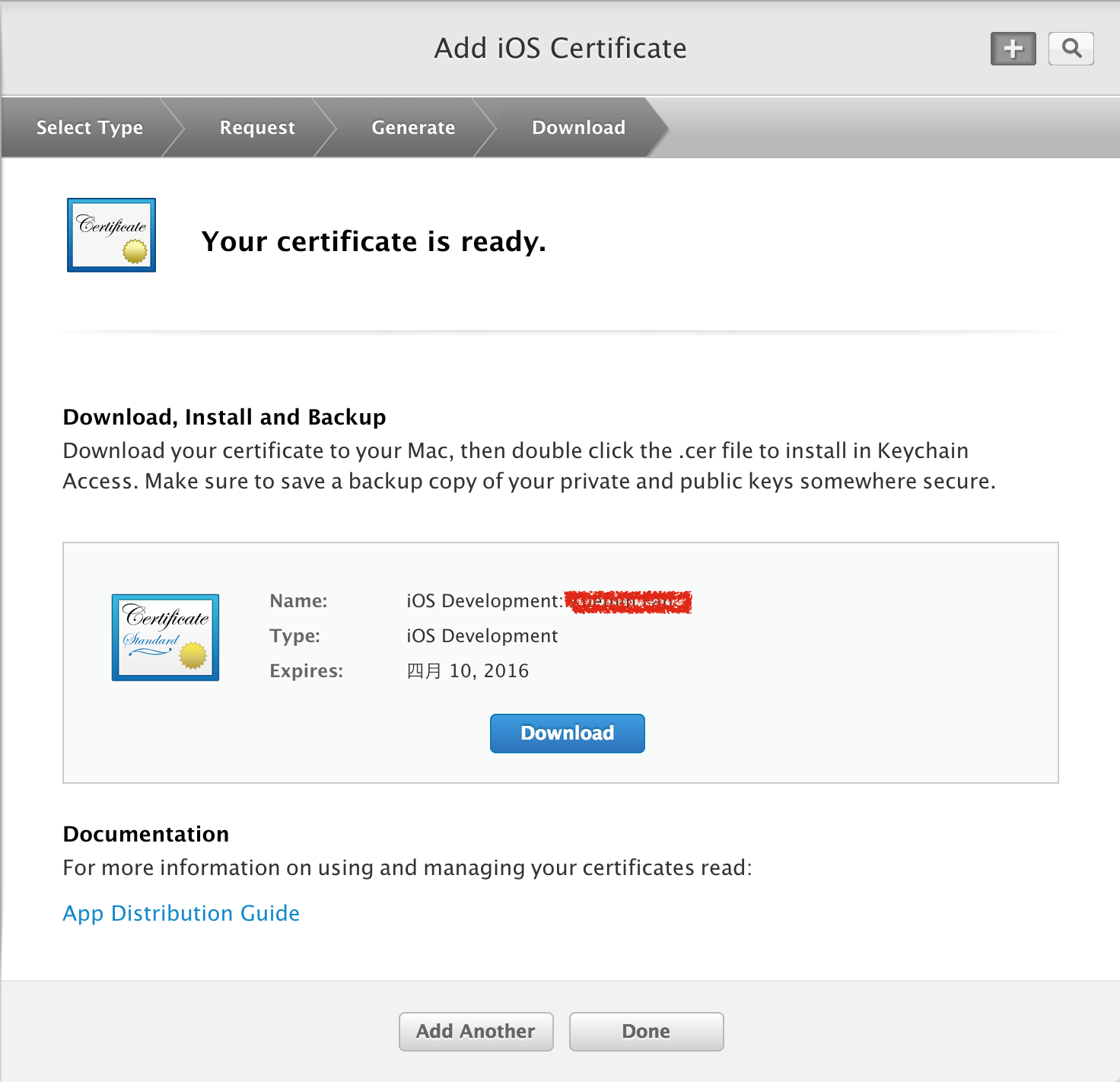
***private key***始终保存在Mac OS的Keychain Access中，用于签名（CodeSign）对外发布的App；***public key***一般随证书（随Provisioning Profile，随App）散布出去，对App签名进行校验认证。用户必须保护好本地Keychain中的private key，以防伪冒。

* Keep a secure backup of your public-private key pair. If the private key is lost, you’ll have to create an ***entirely new*** identity to sign code.
* Worse, if someone else has your private key, that person may be able to ***impersonate***you.

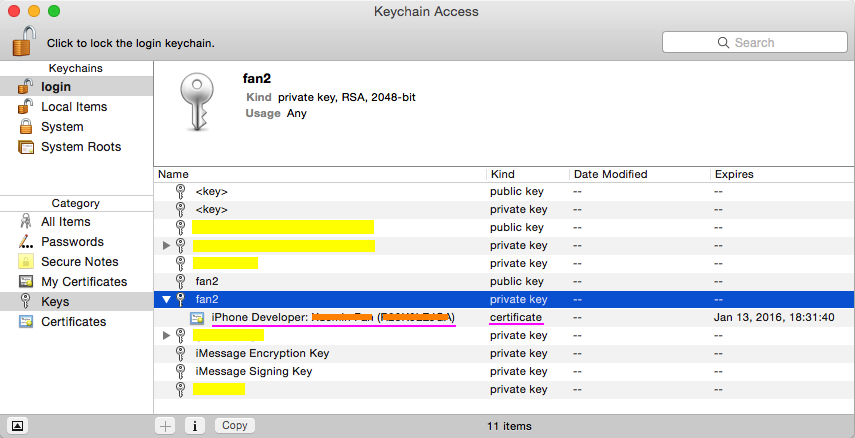
在Apple开发网站上传该CSR文件来添加证书（Upload CSR file to generate your certificate）：

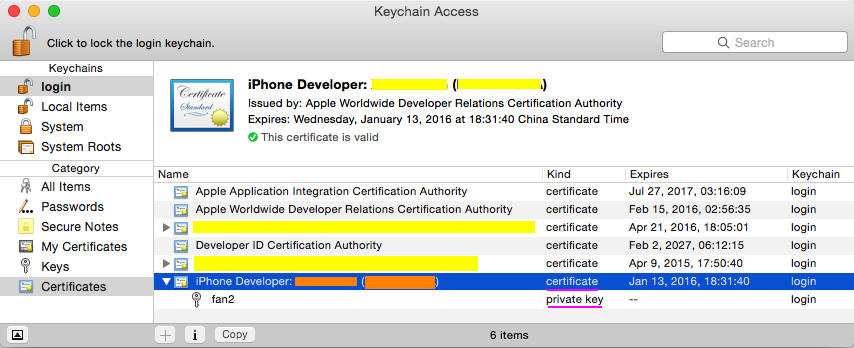


Apple证书颁发机构WWDRCA*[(Apple Worldwide Developer Relations Certification Authority)](https://developer.apple.com/certificationauthority/AppleWWDRCA.cer" \t "_blank)*将使用private key对CSR中的public key和一些身份信息进行加密签名生成**数字证书**（ios\_development.cer）并记录在案（Apple Member Center）。

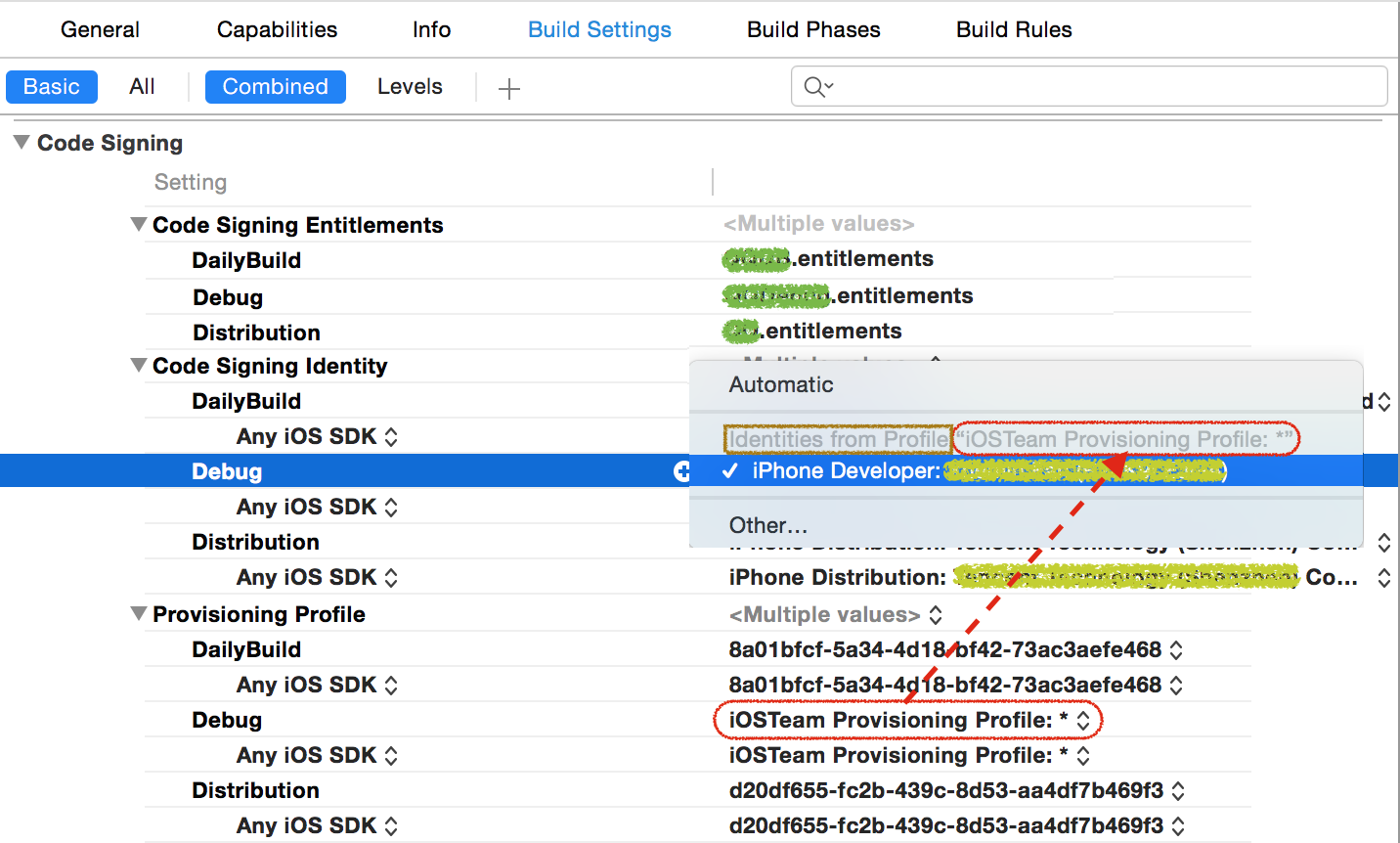


从Apple Member Center网站下载证书到Mac上双击即可安装（当然也可在Xcode中添加开发账号自动同步证书和[生成]配置文件）。证书安装成功后，在 KeychainAccess|Keys中展开创建CSR时生成的Key Pair中的私钥前面的箭头，可以查看到包含其对应公钥的证书（Your requested certificate will be the public half of the key pair.）；在Keychain Access|Certificates中展开安装的证书（ios\_development.cer）前面的箭头，可以看到其对应的私钥。





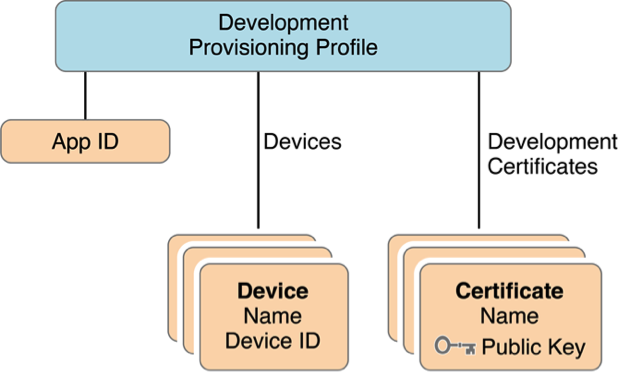
Certificate**被配置到**【Xcode Target|Build Settings|Code Signing|Code Signing Identity】下，下拉选择Identities from Profile "..."（一般先配置Provisioning Profile）。以下是Xcode配置示例：



**四.供应配置文件（[Provisioning Profiles](https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/MaintainingCertificates/MaintainingCertificates.html" \t "_blank)）**

**1.Provisioning Profile的概念**

Provisioning Profile文件包含了上述的所有内容：**证书、App ID和设备**。



一个Provisioning Profile对应一个Explicit App ID或Wildcard App ID（一组相同Prefix/Seed的App IDs）。在网站上手动创建一个Provisioning Profile时，需要依次指定App ID（单选）、证书（Certificates，可多选）和设备（Devices，可多选）。用户可在网站上删除（Delete）已注册的Provisioning Profiles。

Provisioning Profile决 定Xcode用哪个证书（公钥）/私钥组合（Key Pair/Signing Identity）来签署应用程序（Signing Product）,将在应用程序打包时嵌入到.ipa包里。安装应用程序时，Provisioning Profile文件被拷贝到iOS设备中，运行该iOS App的设备也通过它来认证安装的程序。

如果要打包或者在真机上运行一个APP，一般要经历以下三步：

* 首先，需要指明它的App ID，并且验证Bundle ID是否与其一致；
* 其次，需要证书对应的私钥来进行签名，用于标识这个APP是合法、安全、完整的；
* 然后，如果是真机调试，需要确认这台设备是否授权运行该APP。

Provisioning Profile把这些信息全部打包在一起，方便我们在调试和发布程序打包时使用。这样，只要在不同的情况下选择不同的Provisioning Profile文件就可以了。

Provisioning Profile也分为Development和Distribution两类，有效期同Certificate一样。Distribution版本的ProvisioningProfile主要用于提交App Store审核，其中不指定开发测试的Devices（0，unlimited）。App ID为Wildcard App ID（\*）。App Store审核通过上架后，允许所有iOS设备（Deployment Target）上安装运行该App。

Xcode将全部供应配置文件（包括用户手动下载安装的和Xcode自动创建的Team Provisioning Profile）放在目录[~/Library/MobileDevice/Provisioning Profiles](http://blog.csdn.net/iamfreedom2011/article/details/22160853" \t "_blank)下。

**2.Provisioning Profile的构成**

以下为典型供应配置文件\*.mobileprovision的**构成简析**：

（1）***Name***：该mobileprovision的文件名。

（2）***UUID***：该mobileprovision文件的真实文件名。

（3）***TeamName***：Apple ID账号名。

（4）***TeamIdentifier***：Team Identity。

（5）***AppIDName***：explicit/wildcard App ID name（ApplicationIdentifierPrefix）。

（6）***ApplicationIdentifierPrefix***：完整App ID的前缀（TeamIdentifier.\*）。

（7）***DeveloperCertificates***：包含了可以为使用该配置文件应用签名的所有证书。

证书是基于Base64编码，符合PEM(PrivacyEnhanced Mail, RFC 1848)格式的，可使用OpenSSL来处理（opensslx509 -text -in file.pem）。

从DeveloperCertificates提取之间的内容到文件cert.cer（cert.perm）：

-----BEGIN CERTIFICATE-----

将之间的内容拷贝至此

-----END CERTIFICATE-----`

Mac下右键QuickLook 查看cert.cer（cert.perm），在Keychain Access中右键Get Info查看对应证书ios\_development.cer，正常情况（公私钥KeyPair配对）应吻合；Windows下没有足够信息 （WWDRCA.cer），无法验证该证书。

如果你用了一个不在这个列表中的证书进行签名，无论这个证书是否有效，这个应用都将CodeSign Fail。

（8）***Entitlements***键对应的：

**keychain-access-groups**：$(AppIdentifierPrefix)，参见***Code Signing Entitlements***(\*.entitlements)。

每个应用程序都有一个可以用于安全保存一些如密码、认证等信息的**[keychain](http://blog.k-res.net/archives/1081.html" \o "View all posts in keychain" \t "_blank)**，一般而言自己的程序只能访问自己的keychain。通过对应用签名时的一些设置，还可以利用keychain的方式实现同一开发者签证（就是相同bundle seed）下的不同应用之间共享信息的操作。比如你有一个开发者帐户，并开发了两个不同的应用A和B，然后通过对A和B的keychain access group这个东西指定共用的访问分组，就可以实现共享此keychain中的内容。

**application-identifier**：带前缀的全名，例如$(AppIdentifierPrefix)com.apple.garageband。

**com.apple.security.application-groups**：App Group ID（group. com.apple），参见***Code Signing Entitlements***(\*.entitlements)。

**com.apple.developer.team-identifier**：同Team Identifier。

（9）***ProvisionedDevices***：该mobileprovision授权的开发设备的UDID 。

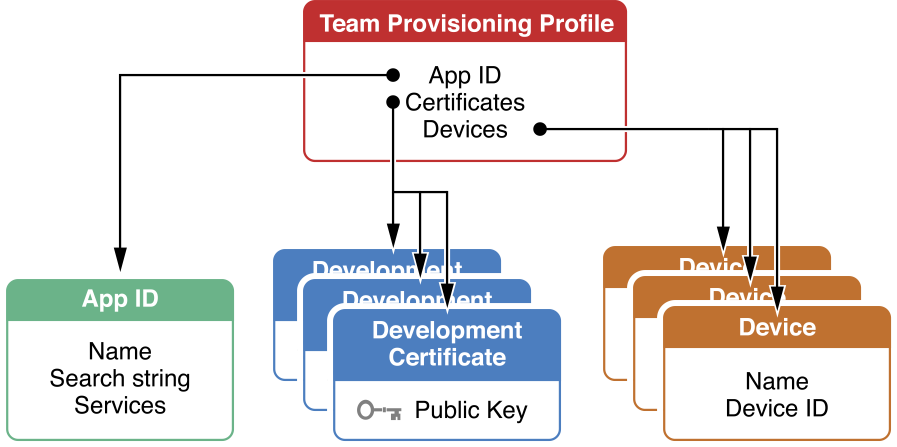
Provisioning Profile**被配置到**【XcodeTarget|Build Settings|Code Signing|Provisioning Profile】下，然后在Code Signing Identity下拉可选择Identities from Profile "..."（即Provisioning Profile中包含的Certificates）。

**五.开发组供应配置文件（[Team Provisioning Profiles](https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppStoreDistributionTutorial/CreatingYourTeamProvisioningProfile/CreatingYourTeamProvisioningProfile.html" \t "_blank)）**

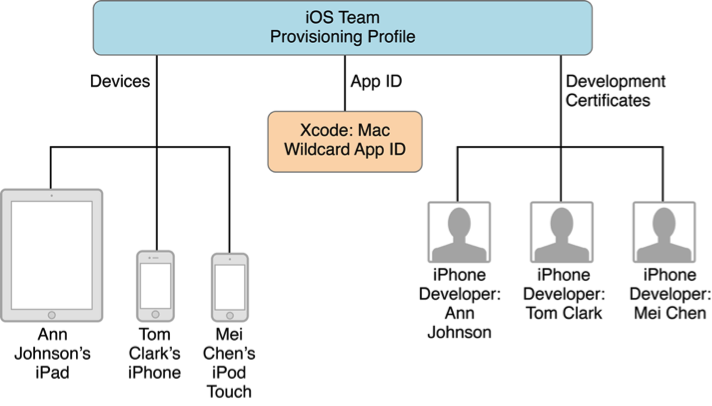
**1.Team Provisioning Profile的概念**

每个Apple开发者账号都对应一个唯一的**Team ID，**Xcode3.2.3预发布版本中加入了Team Provisioning Profile这项新功能。

在Xcode中添加Apple Developer Account时，它将与Apple Member Center后台勾兑**自动生成**iOS Team Provisioning Profile（Managed by Xcode）。



Team Provisioning Profile包含一个为Xcode iOS Wildcard App ID(\*)生成的iOS Team Provisioning Profile:\*（匹 配所有应用程序），账户里所有的Development Certificates和Devices都可以使用它在这个team注册的所有设备上调试所有的应用程序（不管bundle identifier是什么）。同时，它还会为开发者自己创建的Wildcard/Explicit App IDs创建对应的iOS Team Provisioning Profile。



**2.Team Provisioning Profile生成/更新时机**

* Add an Apple ID account to Xcode
* Fix issue "No Provisioning Profiles with a valid signing identity" in Xcode
* Assign Your App to a Team in Xcode project settings of General|Identity
* Register new device on the apple development website or Xcode detected new device connected

利用Xcode生成和管理的iOS Team Provisioning Profile来进行开发非常方便，可以不需要上网站手动生成下载Provisioning Profile。

Team Provisioning Profile同Provisioning Profile，只不过是由Xcode自动生成的，也**被配置到**【XcodeTarget|Build Settings|Code Signing|Provisioning Profile】下。

**六.App Group （ID）**

**1.App Group的概念**

WWDC14除了发布了OS X v10.10和switf外，iOS 8.0也开始变得更加开放了。说到开放，当然要数应用扩展（[App Extension](http://blog.csdn.net/phunxm/article/details/42715145" \t "_blank)）了。顾名思义，应用扩展允许开发者扩展应用的自定义功能和内容，能够让用户在使用其他应用程序时使用该项功能，从而实现各个应用程序间的功能和资源共享。可以将扩展理解为一个轻量级（nimble and lightweight）的分身。

扩展和其Containing App各自拥有自己的沙盒，虽然扩展以插件形式内嵌在Containing App中，但是它们是独立的二进制包，不可以互访彼此的沙盒。为了实现Containing App与扩展的数据共享，苹果在iOS 8中引入了一个新的概念——App Group，它主要用于同一Group下的APP实现数据共享，具体来说是通过以App Group ID标识的共享资源区——App Group Container。

App Group ID同App ID一样，一般不超过255个ASCII字符。用户可在网站上编辑Explicit App IDs的App Group Assignment；可以删除（Delete）已注册的AppGroup （ID）。

**2.App Group的配置**

Containing App与Extension的Explicit App ID必须Assign到同一App Group下才能实现数据共享，并且Containing App与Extension的App ID命名必须符合规范：

1. 置于同一App Group下的App IDs必须是唯一的（Explicit，not Wildcard）
2. Extension App ID以Containing App ID为Prefix/Seed

假如Garageband这个App ID为“com.apple.garageband”，则支持从语音备忘录导入到Garageband应用的插件的App ID可能形如“com.apple.garageband.***extImportRecording***”。

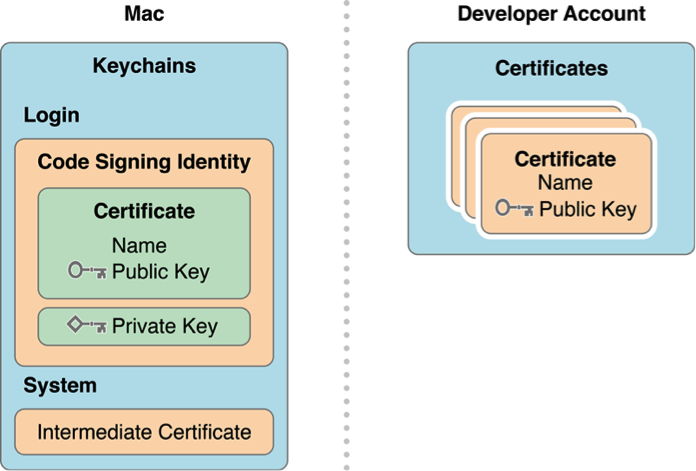
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **App(ex)** | **App Group ID** | **Provisioning Profile** | | |
| **Code Signing Identity**  （Certificate Key Pair） | **App ID**  （bundle identifier） | **Devices**  （test） |
| ***GarageBand*** | 置于同一分组：  group.com.apple | （1）共用同一证书：ios\_development.cer  （2）共用证书Key Pair中的Private Key进行CodeSign | com.apple.garageband | **授权开发测试设备的UDIDs** |
| ***GarageBand扩展插件*** | com.apple.garageband.***extImportRecording*** |

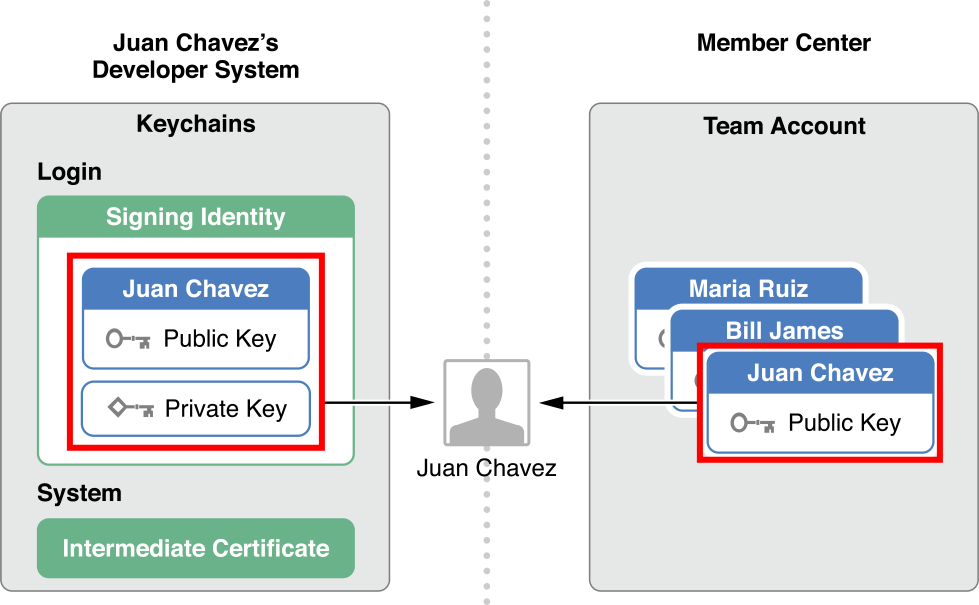
关于Provisioning Profile，可以使用自己手动生成的，也可以使用Xcode自动生成的Team Provisioning Profile。

App Group会**被配置到**【Xcode Target|Build Settings|Code Signing|Code Signing Entitlements】文件（\*.entitlements）的键com.apple.security.application-groups下，不影响Provisioning Profile生成流程。

**七.[证书与签名](https://developer.apple.com/library/mac/documentation/Security/Conceptual/CodeSigningGuide/CodeSigningGuide.pdf" \t "_blank)（Certificate& Signature）**

**1.Code Signing Identity**





Xcode中配置的Code Signing Identity（entitlements、certificate）必须与Provisioning Profile匹配，并且配置的Certificate必须在本机Keychain Access中存在对应Public／Private Key Pair，否则编译会报错。

Xcode所在的Mac设备（系统）使用CA证书（WWDRCA.cer）来判断Code Signing Identity中Certificate的合法性：

* 若用WWDRCA公钥能成功解密出证书并得到公钥（Public Key）和内容摘要（Signature），证明此证书确乃AppleWWDRCA发布，即证书来源可信；
* 再对证书本身使用哈希算法计算摘要，若与上一步得到的摘要一致，则证明此证书未被篡改过，即证书完整。

**2.Code Signing**

每个证书（其实是公钥）对应Key Pair中的**私钥**会被用来对内容（executable code，resources such as images and nib files aren’t signed）进行数字**签名**（CodeSign）——使用哈希算法生成内容**摘要**（digest）。

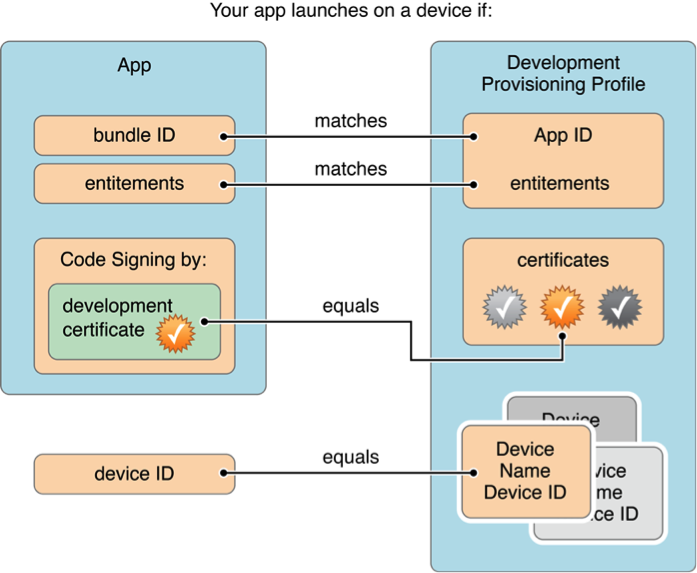
Xcode使用指定证书配套的私钥进行签名时需要授权，选择【始终允许】后，以后使用该私钥进行签名便不会再弹出授权确认窗口。



**3.Verify Code Signature with Certificate**

上面已经提到，公钥被包含在数字证书里，数字证书又被包含在描述文件(Provisioning File)中，描述文件在应用被安装的时候会被拷贝到iOS设备中。

第一步，App在Mac／iOS真机上启动时，需要对配置的bundle ID、entitlements和certificate与Provisioning Profile进行匹配校验：



第二步，iOS/Mac真机上的ios\_development.cer被AppleWWDRCA.cer中的 public key解密校验合法后，获取每个开发证书中可信任的公钥对App的可靠性和完整性进行校验。

**iOS/Mac设备（系统）使用App Provisioning Profile（Code Signing Identity）中的开发证书来判断App的合法性：**

* 若用证书公钥能成功解密出App（executable code）的内容摘要（Signature），证明此App确乃认证开发者发布，即来源可信；
* 再对App（executable code）本身使用哈希算法计算摘要，若与上一步得到的摘要一致，则证明此App（executable code）未被篡改过，即内容完整。

**小结：**

* 基于Provisioning Profile校验了CodeSign的一致性；
* 基于Certificate校验App的可靠性和完整性；
* 启动时，真机的device ID（UUID）必须在Provisioning Profile的***ProvisionedDevices***授权之列。

**八.在多台机器上共享开发账户/证书**

**1.Xcode导出开发者账号(\*.developerprofile)或[PKCS12文件(\*.p12)](http://certhelp.ksoftware.net/support/solutions/articles/17251-what-is-a-p12-file-or-a-pkcs12-file-" \t "_blank)**

进入Xcode Preferences|Accounts：

* 选中Apple IDs列表中对应Account的的Email，点击+-之后的☸|Export Accounts，可导出包含account/code signing identity/provisioning profiles信息的\*.**developerprofile**（Exporting a Developer Profile）文件供其他机器上的Xcode开发使用（Import该Account）。

选中右下列表中某行Account Name条目|ViewDetails，可以查看Signing Identities和Provisioning Profiles。

* 选中欲导出的Signing Identity条目，点击栏底+之后的☸|Export，必须输入密码，并需授权export key "privateKey" from keychain，将导出**Certificates.[p12](http://appfurnace.com/2015/01/how-do-i-make-a-p12-file/" \t "_blank)。**

点击左下角的刷新按钮可从Member Center同步该账号下所有的Provisioning Profile到本地。  
选中右击列表中某个Provisioning Profile可以【Show in Finder】到[~/Library/MobileDevice/Provisioning\ Profiles]目录，其中Provisioning Profile的真实名称为$(UUID).mobileprovision，名如"2488109f-ff65-442e-9774- fd50bd6bc827.mobileprovision"，其中Name中为Xcode中看到的描述 性别名。

**2.Keychain Access导出[PKCS12](http://blog.csdn.net/kmyhy/article/details/6431609" \t "_blank)文件(\*.[p12](http://help.adobe.com/zh_CN/as3/iphone/WS144092a96ffef7cc-371badff126abc17b1f-7fff.html" \t "_blank))**

**在Keychain Access|Certificates中选中欲导出的certificate或其下private key，右键Export或者通过菜单File|Export Items导出Certificates.[p12](https://watch/?v=1X10zCzhukI" \t "_blank)——PKCS12 file holds the** **private key** **and certificate。**

其 他Mac机器上双击Certificates.p12（如有密码需输入密码）即可安装该共享证书。有了共享证书之后，在开发者网站上将欲调试的iOS设备 注册到该开发者账号名下，并下载对应证书授权了iOS调试设备的Provisioning Profile文件，方可在iOS真机设备上开发调试。

**九.[证书配置常见错误](https://developer.apple.com/library/ios/technotes/tn2318/_index.html" \t "_blank)**

**1.no such provisioning profile was found**

Xcode Target|Genera|Identity Team下提示"Your build settings specify a provisioning profile with the UUID "xxx",howerver, no such provisioning profile was found."

Xcode Target|BuildSettings|Code Signing|当前配置的指定UDID的provisioning profile在本地不存在，此时需要更改Provisioning Profile。必要时手动去网站下载或重新生成Provisioning Profile或直接在Xcode中Fix issue予以解决（可能自动生成iOS Team ProvisioningProfile）！

**2.No identities from profile**

Build Settings|CodeSigning的Provisioning Profile中选择了本地安装的provisioning profile之后，Code Signing Identity中下拉提示[No identities from profile “…”](http://stackoverflow.com/questions/21675371/no-identities-from-profile-happened-after-i-upgraded-to-xcode-5" \t "_blank)or No identities from keychain.

Xcode配置指定UDID的provisioning profile中的DeveloperCertificates在本地KeyChain中不存在（[No identities are available](http://stackoverflow.com/questions/18746703/no-identities-are-available-for-signing-xcode-5" \t "_blank)）或不一致（KeyPair中的Private Key丢失），此时需去网站检查ProvisioningProfile中的App ID-Certificate-Device配置是否正确。如果是别人提供的共享账号（\*.developerprofile）或共享证书 (\*.p12)，请确保导出了对应Key Pair中的Private Key。必要时也直接在Xcode中Fix issue予以解决（可能自动生成iOS Team ProvisioningProfile）。

**3.Code Signing Entitlements file do not match profile**

"[Invalid application-identifier Entitlement](https://developer.apple.com/library/ios/qa/qa1710/_index.html" \t "_blank)" or "Code Signing Entitlements file do not match those specified in your provisioning profile.(0xE8008016)."

**（1）**检查对应版本（Debug）指定的\*.entitlements文件中的“Keychain Access Groups”键值是否与ProvisioningProfile中的Entitlements项相吻合（后者一般为前者的Prefix/Seed）。

**（2）也可以将**Build Settings|Code Signing的Provisioning Profile中对应版本（Debug）的Entitlements置空。

4.Xcode配置反应有时候不那么及时，可刷新、重置相关配置项开关（若有）或重启Xcode试试。