



# MIDlet 程序自签名方法

针对 Nokia Serial 系列手机

文档撰写人：今去冠首你你魔

文档完成日：2008 年 1 月 9 日

書劍堂 · 今去冠首你你魔

文档来源 · 书籍来源 · 网络来源 · 网络来源 · 网络来源 · 网络来源

修改记录			
修改类型	修改日期	修改者	描述
创建	2008-1-9	JqgsNinimo	描述了 MIDlet 程序自签名的方法

# 目录

1.	适用情况.....	4
2.	使用限制.....	5
3.	实用步骤.....	6
3.1.	预先准备 .....	6
3.1.1.	IDE 工具 .....	6
3.1.2.	KeyTool 工具.....	6
3.1.3.	JadTool 工具 .....	6
3.2.	创建安装文件 .....	7
3.2.1.	权限声明标签.....	7
3.2.2.	通过 Carbide.j 进行权限声明 .....	8
3.2.3.	通过 WTK 进行权限声明 .....	9
3.3.	创建并导出证书 .....	11
3.3.1.	创建密钥.....	11
3.3.2.	导出证书.....	14
3.4.	对 MIDlet 套件签名 .....	15
3.4.1.	添加证书信息到 JAD 文件 .....	15
3.4.2.	添加签名信息到 JAD 文件 .....	17
3.5.	安装到目标手机 .....	19
3.5.1.	安装自签名证书.....	19
3.5.2.	核实证书安装情况.....	21
3.5.3.	安装已签名 MIDlet 套件.....	22
3.5.4.	程序设置.....	24
3.5.5.	已签名与未签名套件之比较.....	25
4.	其它签名方法.....	27
4.1.	预先准备 .....	27
4.2.	使用 Carbide.j 签名 .....	27
4.3.	使用 WTK 签名 .....	31

## 1. 适用情况

使用 J2ME 开发的 MIDlet 应用套件，往往会在运行过程中弹出烦人的“请求用户授权”窗口。这是由于该 MIDlet 程序未被 CA 授权，属于非受信 MIDlet（Untrusted MIDlet）。而当非受信 MIDlet 访问系统敏感 API 时，出于对手机安全性的考虑，设备就需要显式地获得用户许可。

解决该问题的通常做法是购买权威认证机构签署的证书进行签名，使程序成为受信 MIDlet（Trusted MIDlet）。当然，前提是您的手机设备支持该种证书。

但是，权威证书的申请会消耗一定的人力物力，这笔开支对于您兹待签名的 MIDlet 来说也许较大（一个典型的例子是，开发者为自己的手机设计了一款非商业性 MIDlet）。这时，您就可以选择自签名的方法。

## 2. 使用限制

自签名作为一种特殊的解决方案并不被所有手机支持。所以首先您要确定程序运行的手机是否支持自签名。

对于 Nokia 的 Serial 系列来说，自签名不被早期的 Serial 40 和如今的 S60 第三版支持。这对于目前大量的 S60 第三版手机用户来说真是个坏消息。

## 3. 实用步骤

下面，我将对自签名的实用步骤作一详细描述。该步骤于 Nokia N70 上实验获得。Nokia N70 机型属于 S60 第二版。由于实验条件限制，本人并不保证该步骤具有普适性。

### 3.1. 预先准备

以下所列是实施该方法必备的工具：

- 开发 MIDlet 程序的 IDE 工具
- 管理密钥证书的 KeyTool 工具
- 进行套件签名的 JadTool 工具

下面对它们一一作以介绍。

#### 3.1.1. IDE工具

一个好的集成开发环境可以使您的开发更为快捷，相信这篇文章的读者都已拥有一套开发 MIDlet 程序的 IDE。

为了保证方法的通用性，我一般采用最基本的实现方法：通过 DOS 命令行方式直接调用功能函数。这不免对方法的实施增添了复杂度。事实上，一个好的 IDE 工具已经将这些功能集成在一个图形界面中，为我们提供了友好的操作接口。所以您在遵照我所提供的步骤进行实施之前最好检查一下自己的 IDE 是否提供了这些功能。

另外要说明一点，我所用的 IDE 是“Eclipse+Carbide.j”，所以在之后的论述中，我会在适当的时候说明如何用 Carbide.j 插件来实现相同的功能。

#### 3.1.2. KeyTool工具

KeyTool 是 JDK 提供的密钥和证书管理工具。它是 JDK 附带的工具，位于 JDK 安装文件夹下的“bin”文件夹中。在后面的步骤中，我们会使用该工具创建密钥对和自签名证书。

#### 3.1.3. JadTool工具

JadTool 是随 J2ME 工具包提供的一个命令行工具，用于对 MIDlet 套件进行签名。JadTool 只能使用 J2SE 密钥库中的密钥和证书（即用 KeyTool 工具生成的密钥和证书）。

如果您安装了 WTK，则“JadTool.jar”文件位于 WTK 安装文件夹下的

“bin” 文件夹中。

一般情况下，开发 MIDP 程序的 IDE 或插件都会提供 JadTool 工具的，比如我使用的 Carbide.j 插件在其安装目录下的“bin\lib”文件夹下提供了“JadTool.jar”文件。您可以通过文件搜索工具找到它。

## 3.2. 创建安装文件

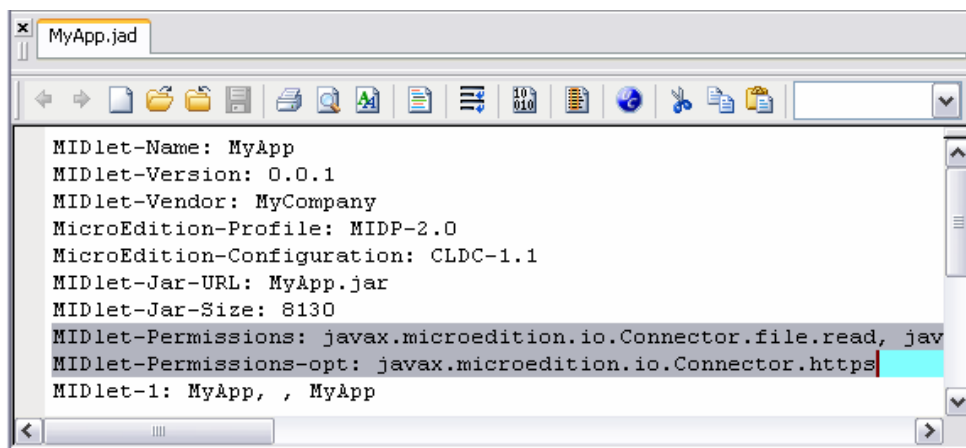
其实，对于如何生成 JAR 和 JAD 文件，大家一定都很熟悉。我这里提到它，主要是为了强调一下权限的设置问题。我们一定要在 JAD 和 JAR 文件中声明程序需要的权限。没有权限声明的签名程序安装后不能访问敏感 API，比不签名还要糟糕。

### 3.2.1. 权限声明标签

有两种标签可以用来声明权限，分别是：

- **MIDlet-Permissions**
- **MIDlet-Permissions-Opt**

这两种标签都可以指定多种权限，权限之间用逗号分割，例如：

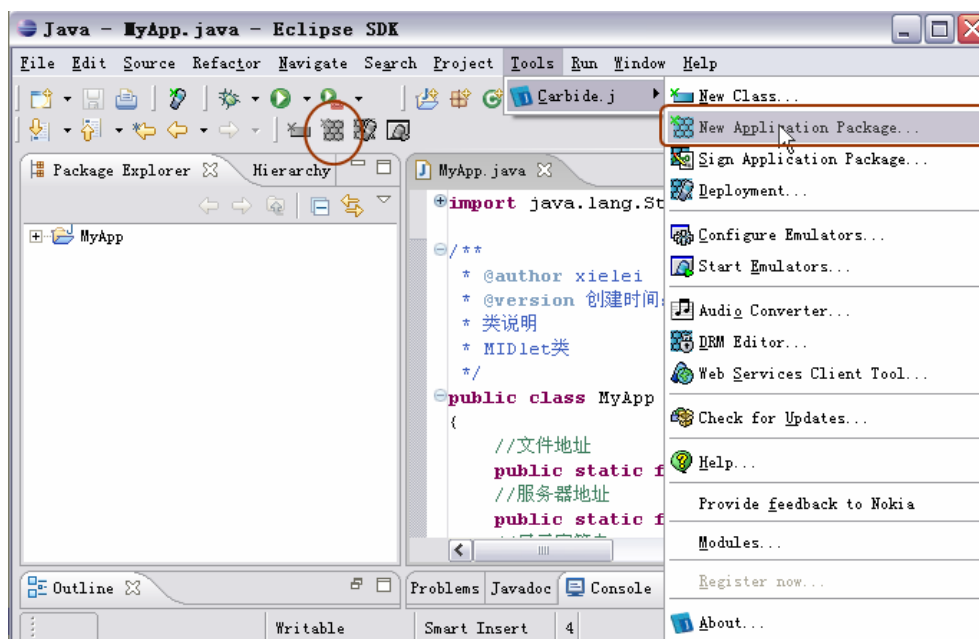


在安装过程中，设备需要检查 MIDlet 要求的权限。如果在 MIDlet-Permissions 中声明了某种权限，设备要么把 MIDlet 安装在保护域中并赋予其相应权限，要么中止安装；如果权限是声明在 MIDlet-Permissions-Opt 标签中，且 MIDlet 套件未被授予对适当保护域的访问权限，安装则可以继续，但套件无法获得其要求的访问权限。所以，如果程序依赖于某种权限，不能在缺少它的情况下运行，那么该权限应该声明在 MIDlet-Permissions 中；如果程序为了完成某些特殊功能而需要使用某种权限，但即便未获得该权限也能正常运行，那么就使用 MIDlet-Permissions-Opt 来声明该权限。

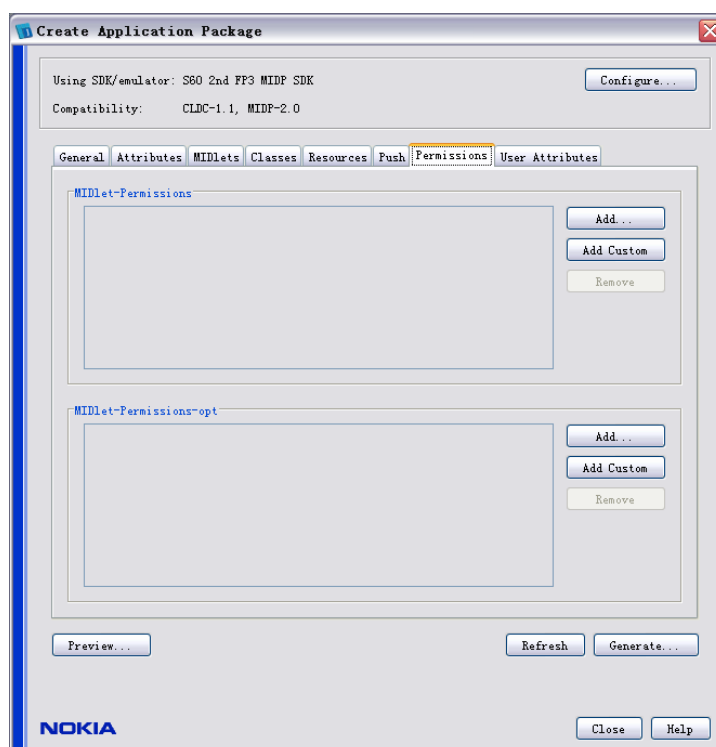
### 3.2.2. 通过Carbide.j进行权限声明

您可以在创建 JAR 和 JAD 文件之前通过 IDE 进行权限设置，下面我以我的 IDE（Eclipse+Carbide.j）为例，详细介绍设置权限声明的步骤：

- 1) 在 Eclipse 视窗中选择菜单“Tools -> Carbide.j -> New Application Package”或点击工具栏相应按钮。打开“Create Application Package”窗口。如图所示：

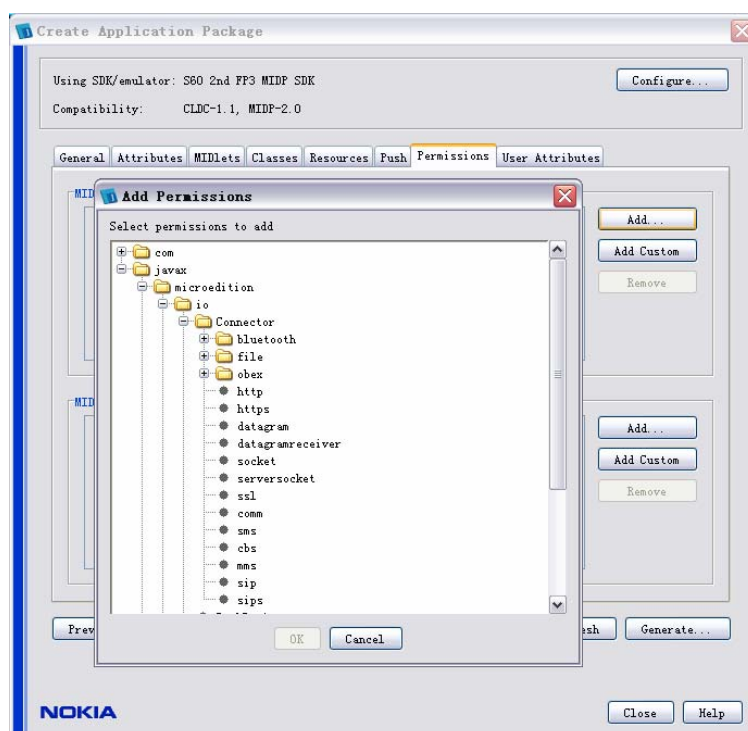


- 2) 在“Create Application Package”窗口中选择“Permissions”选项卡。该选项卡分为上下两部分。上半部分用于添加由 MIDlet-Permissions 标签引导的权限；下半部分用于添加由 MIDlet-Permissions-Opt 标签引导的权限。如图所示：





- 3) 点击每一部分的“Add”按钮，会弹出窗口“Add Permissions”。该窗口提供了树型权限选择列表，以供选择添加权限。如图所示：

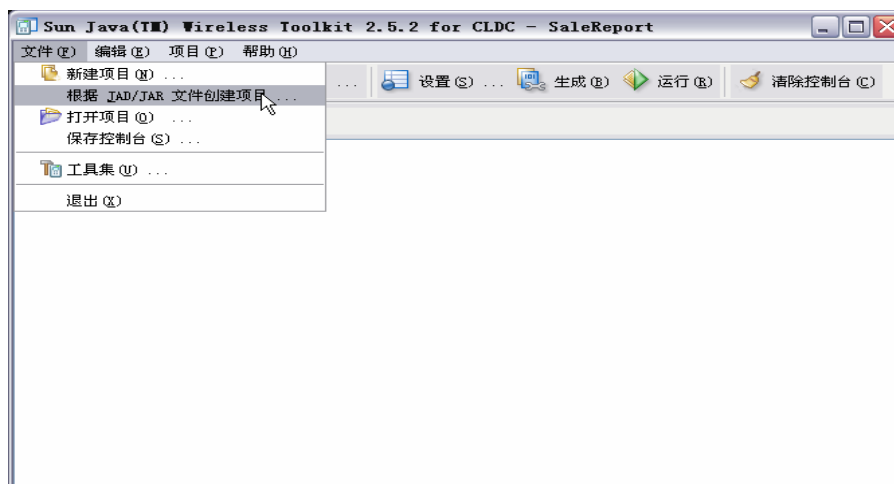


- 4) 在设置好权限后，点击“Create Application Package”窗口中的“Generate”按钮就可以创建出带有权限声明的安装文件了。

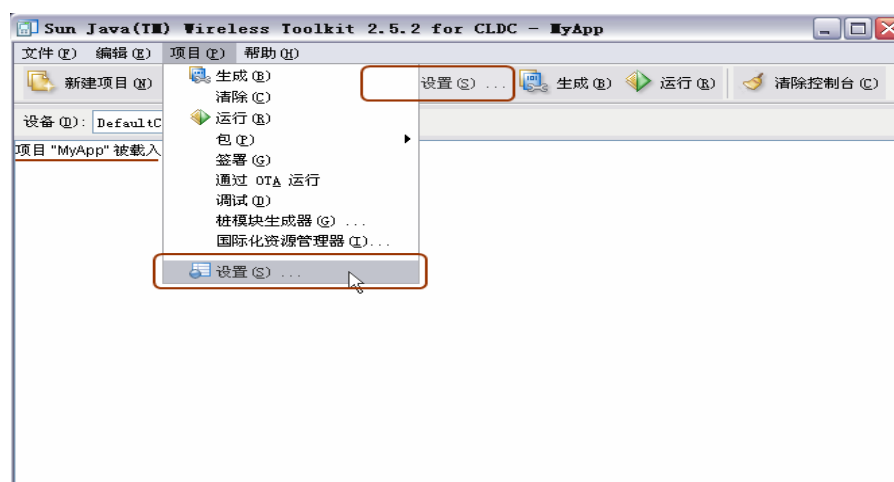
### 3.2.3. 通过WTK进行权限声明

对于未声明权限的 JAR 和 JAD 文件，也可以通过 WTK 进行权限声明的补充：

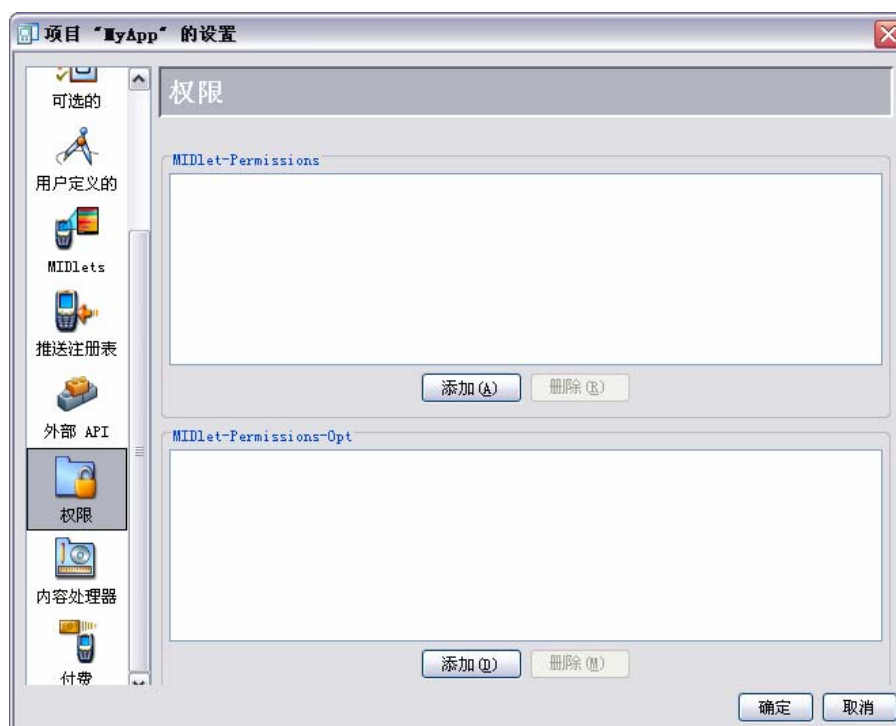
- 1) 在 WTK 视窗中选择菜单“文件->根据 JAD/JAR 文件创建项目”，以通过 JAD/JAR 文件创建项目。如图所示：



- 2) 在弹出的文件选择对话框中，选择要添加权限的 JAD 文件（JAD 文件和 JAR 文件必须在同一个目录下），创建并载入项目。
- 3) 选择菜单“项目->设置”或者点击工具栏中相应按钮，打开“项目设置”对话框。如图所示：



- 4) 在“项目设置”对话框中选择“权限”选项卡。该选项卡的结构和 Eclipse 中的“Permissions”选项卡基本相同，以后的处理方式也基本相同，不再一一赘述。如图所示：



## 3.3. 创建并导出证书

这一步骤使用 KeyTool 工具创建并导出自签名证书（Self-signed Certificate）。

### 3.3.1. 创建密钥

用 KeyTool 工具创建密钥，将会生成相互关联的公钥和私钥，我们称为密钥对（Key Pair）。密钥对生成后，会以密钥项（Key Entry）的结构储存在密钥库中。一个密钥项包含一个私钥和与其关联的证书链（Certificate Chain），证书链的首个证书就是自签名证书，而公钥就包含在自签名证书中。

所以我们可以认为密钥和自签名证书同时产生。

#### 3.3.1.1. 所用DOS命令：

```
keytool -genkey [-alias <alias>] [-keyalg <keyalg>]
               [-keysize <keysize>] [-sigalg <sigalg>]
               [-dname <dname>] [-validity <valDays>]
               [-keypass <keypass>] [-keystore <keystore>]
               [-storepass <storepass>]
```

### 3.3.1.2. 命令说明

- 1) **参数 alias:** 指定别名 (Alias)。别名不区分大小写，在密钥库中具有唯一性，用于标识密钥项。该参数的默认值为 “mykey”。
- 2) **参数 keyalg:** 指定密钥算法 (Key Algorithm)。密钥算法为创建密钥对 (Key Pair) 时所采用的算法，可以为 DSA 或 RSA。该参数的默认值为 “DSA”，一定要将其设置为 “RSA”。
- 3) **参数 keysize:** 指定密钥尺寸 (Key Size)，单位为 “bit”。若参数 keyalg 为 “DSA”，则密钥尺寸被限制在 512bit 和 1024bit 之间，并且要求为 64bit 的倍数。该参数的默认值为 “1024”。一般省略该参数，取其默认值。
- 4) **参数 sigalg:** 指定签名算法 (Signature Algorithm)。签名算法将用于创建自签名证书。若参数 keyalg 为 “DSA”，该参数的默认值为 “SHA1withDSA”；若参数 keyalg 为 “RSA”，该参数的默认值为 “MD5withRSA”。一般省略该参数，取其默认值。
- 5) **参数 dname:** 指定特征名 (Distinguished Names)。特征名遵循 X.500 协议，用实体的基本身份信息来标识实体。KeyTool 的特征名如下表所示：

缩写	全称	意义	示例
CN	Common Name	个人常用名	Xie Lei
OU	Organization Unit	小型组织 (部门或分部) 的名称	R&D
O	Organization Name	大型组织的名称	DealEasy
L	Locality Name	地方 (城市) 名	Shanghai
ST	State Name	州或省份名	Shanghai
C	Country	两个字母的国家代码	CN

在作为参数值时，特征名必须按照以下的格式书写：

```
“CN=cName, OU=orgUnit, O=org, L=city, S=state, C=countryCode”
```

注意不要将引号省略。

该参数没有默认值，如果省略该参数，KeyTool 会在稍后以逐项提示的方式供您指定。所以一般省略该参数。

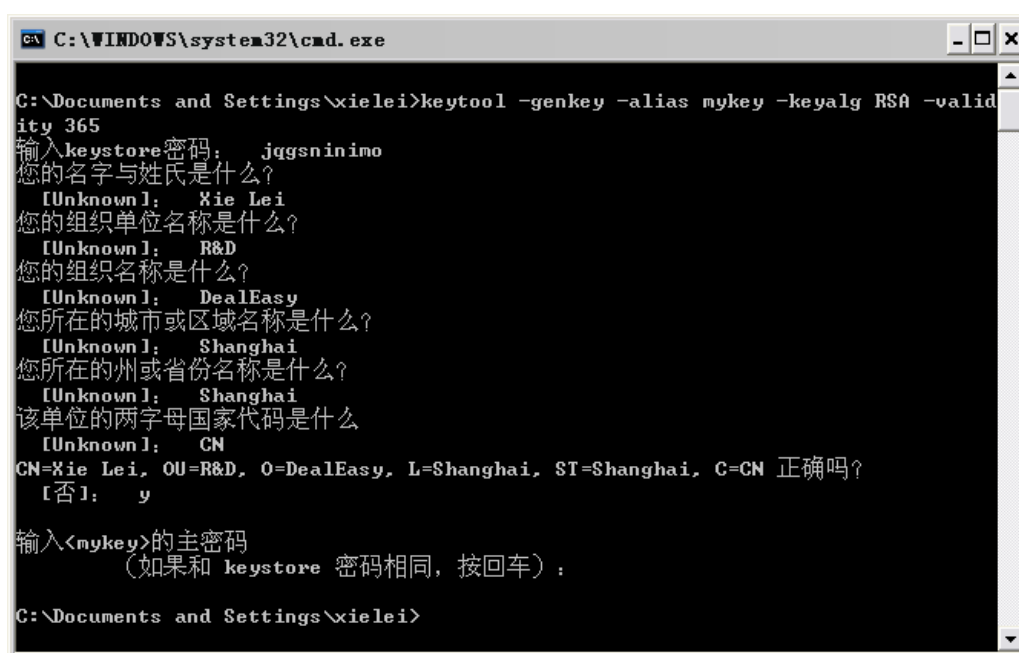
- 6) **参数 validity:** 指定自签名证书的生效期间长度，单位为 “天”。证书

都有一个生效期间 (Validity Period)。KeyTool 将执行该命令的时刻作为证书生效期间的开始时刻，用该参数指定的天数计算出证书生效期间的结束时刻。该参数的默认值为“90”，一般将其设置为“365”。

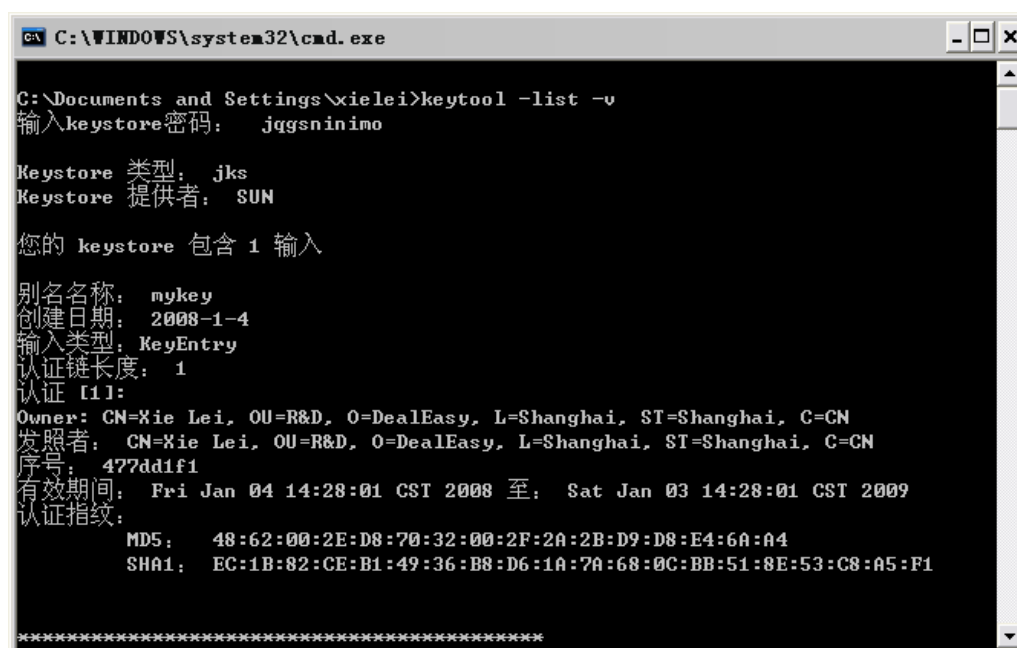
- 7) **参数 keypass:** 指定密钥密码 (Key Password)。该密码用于保护生成密钥对的私钥。该参数无默认值。如果省略该参数，KeyTool 会在稍后提示输入。如果您以回车答复此提示，KeyTool 会将密钥库密码 (Key Store Password) 作为密钥密码。如果没有安全性问题，一般将其设置为密钥库密码。
- 8) **参数 keystore:** 指定所使用密钥库的访问路径。该参数的默认值为“%HOMEDRIVE%%HOMEPATH%/.keystore”。若省略该参数，KeyTool 会将系统当前用户主目录 (一般为 DOS 的默认提示符位置，例如我的主目录为：C:\Documents and Settings\xielei\) 下的“.keystore”文件作为要使用的密钥库文件 (如果不存在就先创建该文件)。一般省略该参数，取其默认值。
- 9) **参数 storepass:** 指定密钥库密码 (Key Store Password)。该密码用于保护密钥库。该参数无默认值。如果省略该参数，KeyTool 会在稍后提示输入。

### 3.3.1.3. 应用示例

下面的截图是我在 MS-DOS 下进行的创建密钥操作：



通过以上的操作，KeyTool 在我的 Windows 主目录下创建了密钥库“.keystore”，并将生成的密钥对和自签名证书存储在密钥库中。您可以通过下面的过程查看生成的密钥：



```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\xielei>keytool -list -v
输入keystore密码:  jggsninimo

Keystore 类型:  jks
Keystore 提供者:  SUN

您的 keystore 包含 1 输入

别名名称:  mykey
创建日期:  2008-1-4
输入类型:  KeyEntry
认证链长度:  1
认证 [1]:
Owner: CN=Xie Lei, OU=R&D, O=DealEasy, L=Shanghai, ST=Shanghai, C=CN
发照者:  CN=Xie Lei, OU=R&D, O=DealEasy, L=Shanghai, ST=Shanghai, C=CN
序号:  477dd1f1
有效期间:  Fri Jan 04 14:28:01 CST 2008 至:  Sat Jan 03 14:28:01 CST 2009
认证指纹:
      MD5:   48:62:00:2E:D8:70:32:00:2F:2A:2B:D9:D8:E4:6A:A4
      SHA1:  EC:1B:82:CE:B1:49:36:B8:D6:1A:7A:68:0C:BB:51:8E:53:C8:A5:F1

*****
```

### 3.3.2. 导出证书

在创建密钥对时，自签名证书就已经产生了。为了在目标手机上安装该证书，我们需要把它从密钥库中导出。

#### 3.3.2.1. 所用DOS命令

```
keytool -export [-alias <alias>]
               [-file <cert_file>]
               [-keystore <keystore>]
               [-storepass <storepass>]
```

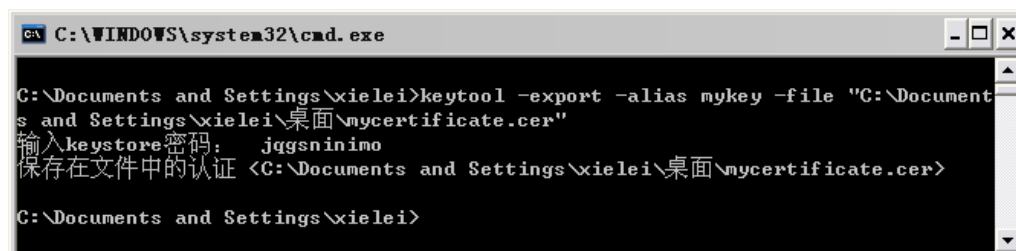
#### 3.3.2.2. 命令说明

- 1) **参数 alias**: 指定所要导出证书存在项的别名。这里补充说明一下。KeyTool 创建的密钥库支持两种数据项，一种是密钥项（上文已经提到，它包括两部分：私钥和证书链），一种是证书项。这两种数据项都统一用别名标识自身。如果该参数指定的是证书项，则该证书被导出；如果该参数指定的是密钥项，则证书链中的第一个证书被导出。该参数的默认值为“mykey”。
- 2) **参数 file**: 用于指定导出证书文件的路径名。如果省略该参数，KeyTool 默认输出端为标准输出（stdout），所以一定要设定该参数。
- 3) **参数 keystore**: 指定所使用密钥库的访问路径。具体参考[3.3.1.2 节](#)。

- 4) 参数 **storepass**: 指定密钥库密码。具体参考[3.3.1.2 节](#)。

### 3.3.2.3. 应用示例

下面的截图是我在 MS-DOS 下进行的导出证书操作:



通过以上的操作, KeyTool 将自签名证书导出到我的 Windows 桌面上。之后, 我就可以将该证书安装到目标手机中了。

## 3.4. 对MIDlet套件签名

这一步骤通过 JadTool 工具, 利用上一步骤生成的自签名证书, 对 MIDlet 套件进行签名。

对 MIDlet 套件进行签名即是将证书信息和 JAR 文件的数字签名信息添加至 JAD 文件的过程。

### 3.4.1. 添加证书信息到JAD文件

我们首先要将证书信息添加到到 JAD 文件中。

#### 3.4.1.1. 所用DOS命令

```
java -jar jadtool.jar -addcert
  -alias <key alias> [-keystore <keystore>]
  [-storepass <password>] [-certnum <number>]
  [-chainnum <number>] [-encoding <encoding>]
  -inputjad <filename> -outputjad <filename>
```

#### 3.4.1.2. 命令说明

- 1) 参数 **alias**: 指定所用证书存在项的别名。该参数必须指定。
- 2) 参数 **keystore**: 指定所使用密钥库的访问路径。帮助文件中称其默认值为 “\$HOME/.keystore”。但在实际操作中, 如果没有设置该参数就不能执行成功。不清楚 “\$HOME” 代表什么地址。
- 3) 参数 **storepass**: 指定密钥库密码。帮助文件中称 “If keystore requires a

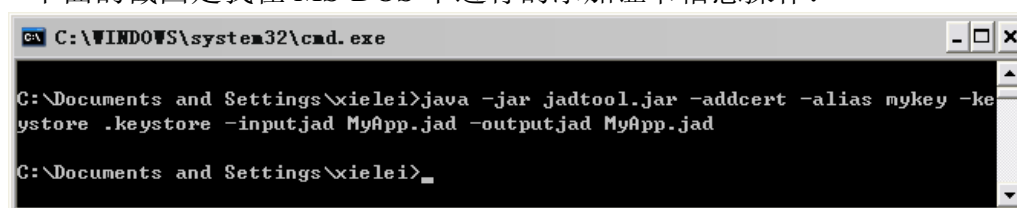


password to access its contents, password must be provided.”，但是奇怪的是不提供该参数依然可以成功执行。

- 4) **参数 certnum:** 指定要替换证书号码。如果需要用新证书替换 JAD 中已包含证书，需要将该参数设置为需替换证书的号码。该参数的默认值为 1。一般省略该参数，取其默认值。
- 5) **参数 chainnum:** 指定证书链的个数。一个 JAD 文件可以包含多个证书链。该参数的默认值为 1。一般省略该参数，取其默认值。
- 6) **参数 encoding:** 指定编码格式。如果参数 inputjad 所引入的 JAD 文件未采用 UTF-8 编码，则该参数必须设置。JadTool 用该参数所指定的编码格式读写由参数 inputjad 和 outputjad 所指定的 JAD 文件。该参数的默认值为“UTF-8”。一般省略该参数，取其默认值（大多数 JAD 文件都用 UTF-8 格式编码）。
- 7) **参数 inputjad:** 指定需要添加证书信息的 JAD 文件。该参数必须指定。
- 8) **参数 outputjad:** 指定处理后 JAD 的存储文件。该参数必须指定。一般将该参数与参数 inputjad 设为相同，以直接改变原 JAD 文件。

### 3.4.1.3. 应用示例

下面的截图是我在 MS-DOS 下进行的添加证书信息操作：



```
C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\xielei>java -jar jadtool.jar -addcert -alias mykey -keystore .keystore -inputjad MyApp.jad -outputjad MyApp.jad

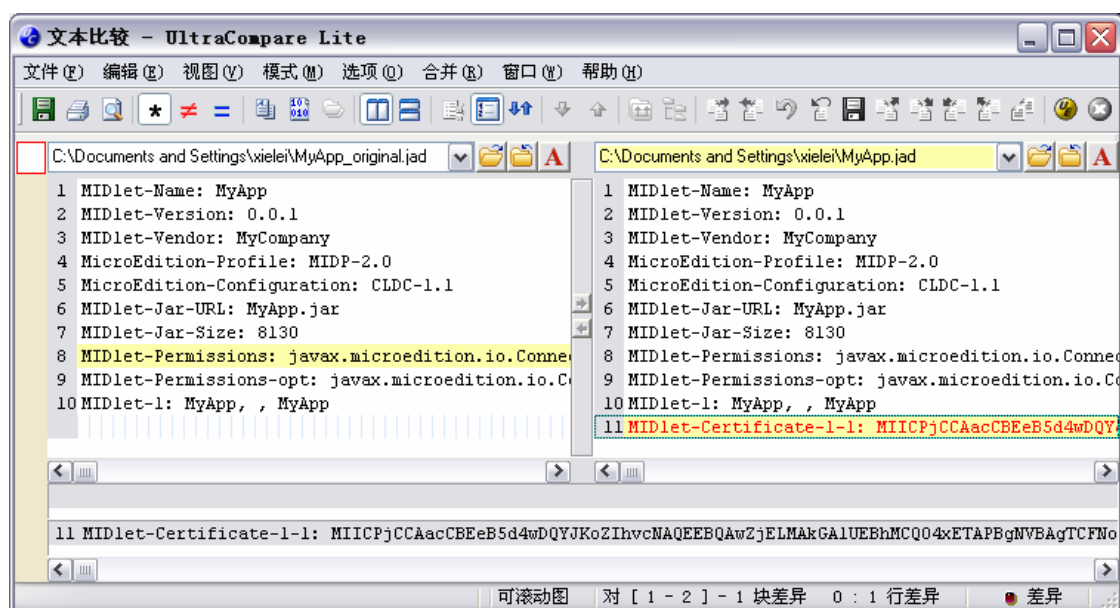
C:\Documents and Settings\xielei>
```

通过以上的操作，JadTool 为 MyApp.jad 文件添加了证书信息。证书信息用这样的属性名引导：

MIDlet-Certificate-m-n

其中，“m”表示证书链号码，“n”表示证书号码。如下所示：





其中左侧为原始的 JAD 文件，右侧为添加了证书的 JAD 文件。您可以看到右侧的 JAD 文件多了第 11 行，该行的属性名为“MIDlet-Certificate-1-1”，引导了证书信息。

### 3.4.2. 添加签名信息到JAD文件

在将证书信息添加到 JAD 文件之后，我们就要对 JAR 文件签名，并将签名信息添加到 JAD 文件中。所以在该步骤，我们需要同时对 JAR 文件和 JAD 文件进行操作。

#### 3.4.2.1. 所用DOS命令

```
java -jar jadtool.jar -addjarsig
[-jarfile <filename>] -keypass <password>
-alias <key alias> -storepass <password>
[-keystore <none|keystore>] [-encoding <encoding>]
-inputjad <filename> -outputjad <filename>
```

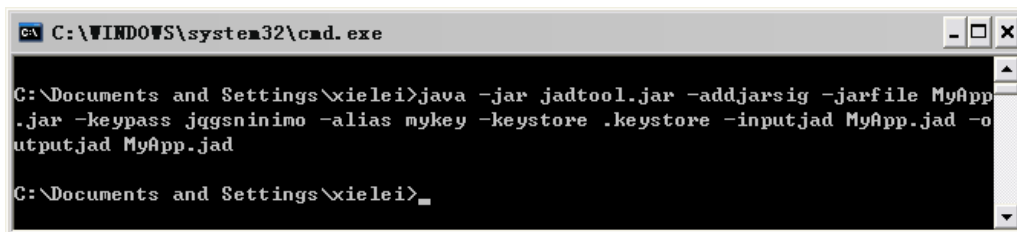
#### 3.4.2.2. 命令说明

- 1) 参数 **jarfile**: 指定需要签名的 JAR 文件。如果省略该参数，JadTool 将参考参数 inputjad 所指定的 JAD 文件中的相关信息。JAD 文件中名为“MIDlet-Jar-URL”的属性说明了该 JAD 所描述的 JAR 的 URL。一般要通过该参数指定 JAR 文件的路径名。
- 2) 参数 **keypass**: 指定所用密钥的密码。该参数不能省略。如果创建密钥时未设置密码，密钥密码会被默认置为密钥库密码。

- 3) 参数 **alias**: 指定所用密钥的别名。该参数不能省略。
- 4) 参数 **storepass**: 指定所用密钥库的密码。具体参考[3.4.1.2 节](#)。
- 5) 参数 **keystore**: 指定所用密钥库的访问路径。具体参考[3.4.1.2 节](#)。
- 6) 参数 **encoding**: 指定编码格式。具体参考[3.4.1.2 节](#)。
- 7) 参数 **inputjad**: 指定需要添加签名信息的 JAD 文件。该参数必须指定。
- 8) 参数 **outputjad**: 指定处理后 JAD 的存储文件。具体参考[3.4.1.2 节](#)。

### 3.4.2.3. 应用示例

下面的截图是我在 MS-DOS 下进行的添加签名信息操作:



```
C:\WINDOWS\system32\cmd.exe

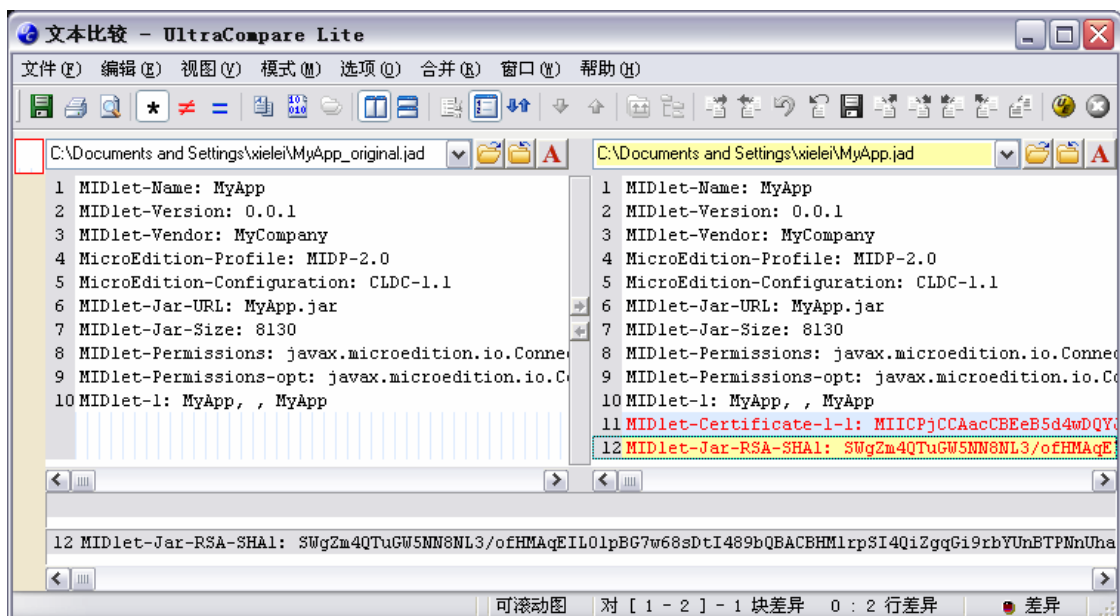
C:\Documents and Settings\xielei>java -jar jadtool.jar -addjarsig -jarfile MyApp.jar -keypass jggsninimo -alias mykey -keystore .keystore -inputjad MyApp.jad -outputjad MyApp.jad

C:\Documents and Settings\xielei>
```

通过以上的操作, JadTool 使用用户指定的私钥为 MIDlet 套件进行了签名, 并将签名信息添加到 MyApp.jad 文件中。签名信息用这样的属性名引导:

MIDlet-Jar-RSA-SHA1

其中“RSA”表示密钥所采用的算法;“SHA1”表示签名所采用的算法。如下所示:



其中左侧为原始的 JAD 文件, 右侧为添加了签名信息的 JAD 文件。您可以看到右侧的 JAD 文件多了第 12 行, 该行的属性名为“MIDlet-Jar-RSA-SHA1”, 引导了签名信息。

## 3.5. 安装到目标手机

通过以上的步骤，我们获得了自签名证书和已签名的 MIDlet 套件：

- mycertificate.cer
- MyApp.jad
- MyApp.jar

接下来，转移战场，将这些文件拷贝到手机上（通过数据线、红外、蓝牙等连接手段），以进行之后的安装工作。

我将这三个文件拷贝到目标手机中存储卡上的 Document 文件夹下，如图所示：



之后，我们就可以在手机上安装自签名证书和 MIDlet 套件了。

### 3.5.1. 安装自签名证书

- 1) 通过“功能表->工具->文件管理”，打开证书“mycertificate.cer”文件，进入“存储证书”界面。如图所示：



- 2) 点击左键，选择“储存”命令，弹出提示“新证书可能不安全”。如图所示：



- 3) 仍选择“储存”命令，弹出编辑窗口，让您输入证书栏目。编辑框内默认的内容为创建证书时填写的 CN (Common Name)。如图所示（其左下角的命令标识原为“确定”，使用截图工具导致其改变）：



- 4) 选择“确定”命令，弹出选择窗口，让您选择受信应用程序。其中罗列了四种应用程序。由于我们的证书用于安装应用程序，所以我们仅选择“应用程序安装”选项就可以了。当然全部选择也是可以的。如果不选择“应用程序安装”选项，证书对于应用程序的安装是没有效果的（S60 第三版之所以无法使用自签名证书，主要的原因就是不再提供“应用程序安装”选项）。如图所示：



- 5) 选择“确认”命令，安装完成。

### 3.5.2. 核实证书安装情况

下面，让我们核实一下证书的安装情况：

- 1) 打开“功能表->工具->设置->安全性设置->证书管理”，在证书列表中找到刚刚安装的证书。如图所示：



- 2) 选择“选项”命令，弹出选择列表。选择“证书详情”可查看证书内容；选择“删除”可删除证书。如图所示：



- 3) 选择“信任设置”命令，也可以对证书进行“受信应用程序”的设置。如果您未在证书的安装过程中设置正确，那就可以在这里重新设置了。如图所示：



### 3.5.3. 安装已签名MIDlet套件

在安装已签名 MIDlet 套件时，一定要用 JAD 文件安装，而不要选择 JRE 文件安装。因为证书和签名信息都是存储在 JAD 文件中的，如果用 JRE 文件直接安装，就相当于进行未签名 MIDlet 套件安装。

- 1) 通过“功能表->工具->文件管理”，打开 JAD 文件“MyApp.jad”，弹出询问对话框。如图所示：



- 2) 选择“是”命令继续安装，会弹出“选项”对话框，在这里您可以查看证书和程序的详细信息。如图所示：



- 3) 选择“查看证书”选项，会弹出证书的详细内容。可以看到这正是我们制作的证书。如图所示：



- 4) 选择“确认”命令后返回到“选项”对话框，选择“继续”命令，弹出选择窗口，让您选择安装位置。如图所示：



- 5) 选定位置后，点击“选择”按钮，安装完成。一般情况下，您可以在“功能表->我的助理”中运行您的程序。



### 3.5.4. 程序设置

之后我们要对已安装的程序进行设置，以使程序在运行过程中不再弹出“请求用户授权”窗口。

- 1) 打开“功能表->工具->程序管理”，在已安装程序列表中选择我们的程序“MyApp”。如图所示：



- 2) 点击“选项”按钮，在弹出的命令列表中选择“套件设置”命令。如图所示：





3) 在随后弹出的设置列表中，共列出七种设置选项，您可以依据需求进行适当的安全限制。如图所示：



3.5.5. 已签名与未签名套件之比较

未签名套件不包含证书信息，在查看证书时，会有“证书不存在”的提示。

未签名套件在安装过程中会弹出“应用程序不受信任”的提示。如图所示：



而两者最主要的区别体现在安装后套件的设置选项上。未签名套件选项少、受限大。下表清楚地反映了这一点：

功能	签名情况	安全限制选项（●：包含；○：不包含）			
		不允许	每次询问	第一次询问	总是允许
网络接入	未签名	●	●	●	○
	已签名	●	●	●	●
信息	未签名	●	●	○	○
	已签名	●	●	○	○
程序自动启动	未签名	●	●	●	○
	已签名	●	●	●	●

连接功能	未签名	●	●	●	●
	已签名	●	●	●	●
多媒体	未签名	●	●	●	○
	已签名	●	●	●	●
读取用户数据	未签名	●	●	○	○
	已签名	●	●	●	●
编辑用户数据	未签名	●	●	○	○
	已签名	●	●	●	●

从上表可以看出，一般情况下，未签名程序的安全限制选项缺少“总是允许”这一项。因此，用户就无法完全取消掉烦人的“请求用户授权”窗口。

我们在安装好已签名程序后，对于一些功能不需要确认提示的话，我们就可以将该功能的安全限制设置为“总是允许”。

## 4. 其它签名方法

上面使用 DOS 命令行方式实现了 MIDlet 套件的签名，操作颇为繁琐。其实一些开发 MIDlet 的图形化工具都对此提供了简便的实现接口。下面我以 Carbide.j 和 WTK 为例，说明签名的过程。两者的签名过程是大致相同的，所以我想使用其它工具也应该相差不远。

### 4.1. 预先准备

在操作之前，您首先要用 KeyTool 工具创建好密钥对，用 IDE 生成 MIDlet 套件，并且保证 MIDlet 套件的 JAD 文件和 JAR 文件处在一个目录之下。

下面我来说明一下，为什么需要作这两点准备。

实际上，Carbide.j 和 WTK 都具有独立的密钥库，并且提供有密钥生成功能，您完全可以使用它们方便地生成密钥对。问题是它们并没有提供导出自签名证书的功能。

为了获得自签名证书，您要首先找到它们创建的密钥库文件（一般以“.ks”作为扩展名），之后使用 KeyTool 工具，指定该文件为目标文件库进行证书导出工作。

这样做实际上并不复杂，您完全可以使用图形化工具进行创建密钥对、套件签名的工作，之后再利用 KeyTool 工具导出该密钥对所对应的自签名证书。这样做唯一麻烦的事情是，您要找到所使用工具创建的密钥库在哪里。还有一点要注意的是，在导出证书时，KeyTool 工具会要求您输入密钥库密码。而一般情况下，图形化工具创建的密钥库并没有设置密码。所以，这时您直接回车就可以了。

这里我为了集中说明签名的过程，所以默认您已经用 KeyTool 工具创建好密钥对。那么下面我要做的就是将该密钥对载入到图形化工具的密钥库中。

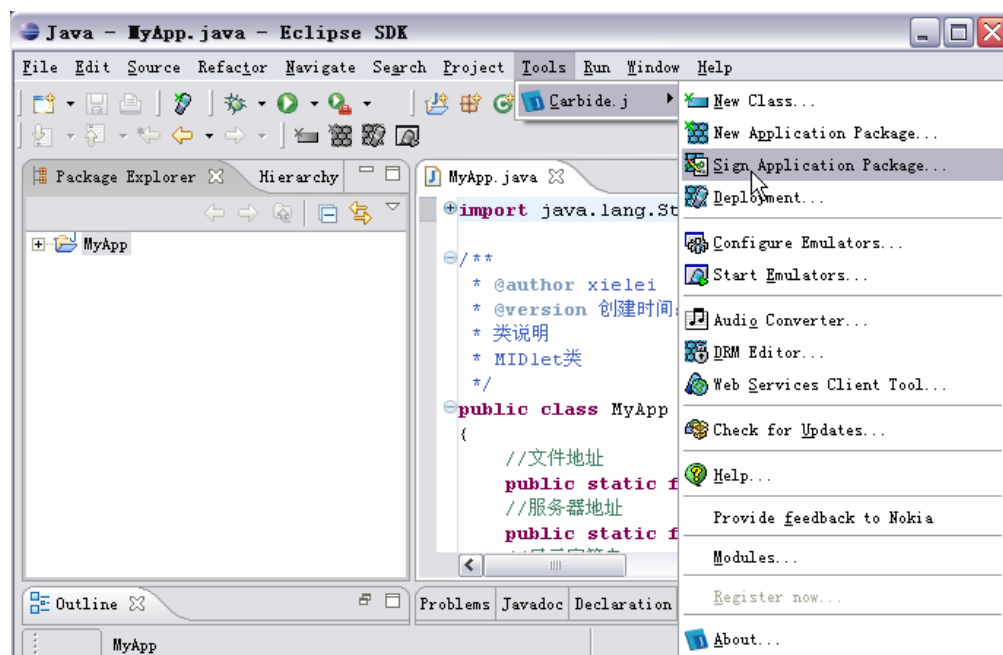
另一个问题是，JAD 文件和 JAR 文件为什么要放在一个目录下。

当做签名工作时，需要同时对 JAD 和 JAR 文件进行处理。而我所用的这两种工具都默认它们存在于一个文件夹中，只要求您指定 JAD 文件的位置。如果它们不在同一个目录下，签名就会失败。

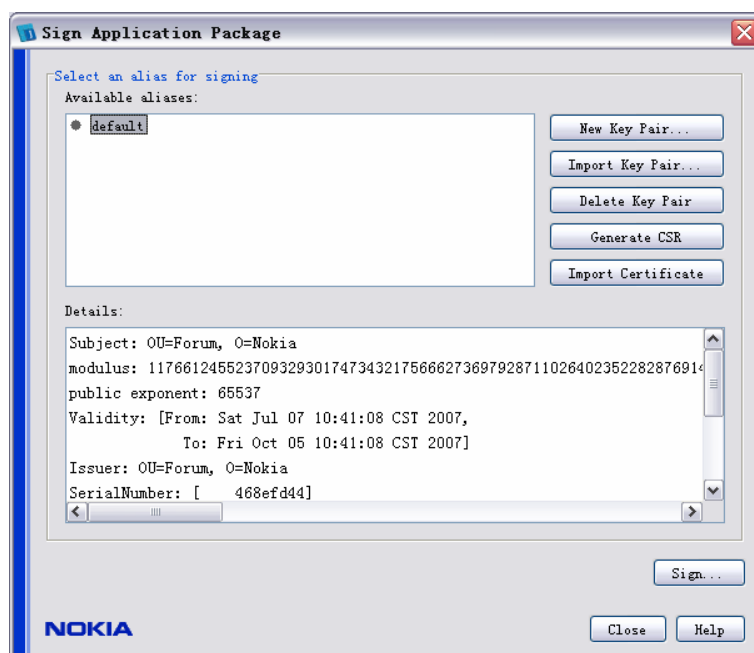
### 4.2. 使用Carbide.j签名

我使用的 IDE 是 Eclipse+Carbide.j。其中 Carbide.j 插件提供了签名功能。下面我来介绍一下用它进行套件签名的步骤：

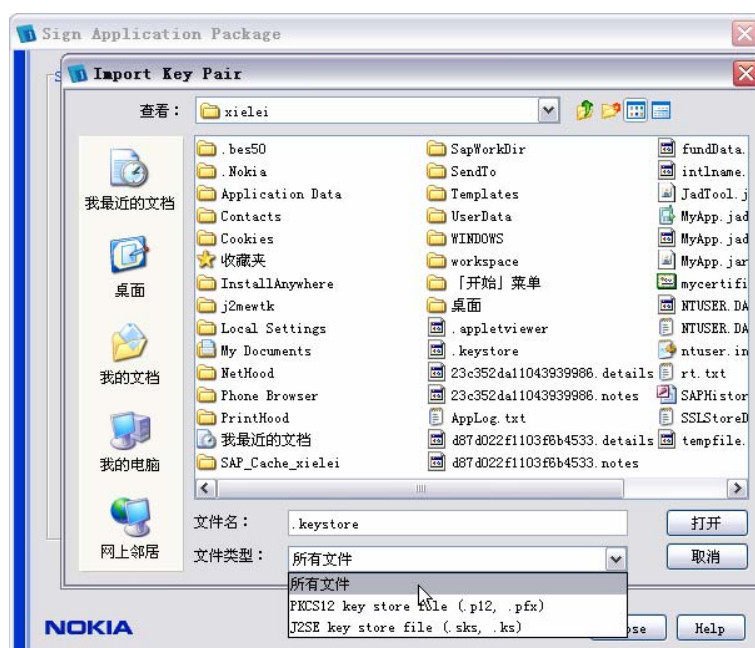
- 1) 打开 Eclipse，选择 “Tools -> Carbide.j -> Sign Application Package” 菜单。如图所示：



- 2) 弹出 “Sign Application Package” 窗口。该窗口就是 Carbide.j 所提供签名功能的主窗口了。该窗口左上方为有效别名列表，显示当前可以使用的密钥项或证书项的别名；右上方提供的五个按钮分别提供新建密钥对、载入密钥对、删除密钥对、创建 CSR 文件和导入证书的功能；下方的显示文本框会显示有效别名列表中被选中别名的详细信息；右下角的 “Sign” 按钮提供对 MIDlet 套件的签名功能；其下的两个标准按钮就不用说明了。如图所示：



- 3) 点击“Import Key Pair”按钮，弹出文件选择窗口。您需要选择要载入密钥对所在的密钥库文件。文件选择窗口默认的文件类型为“.ks”，而用 KeyTool 生成的密钥库文件的默认名是“.keystore”，所以一般要将

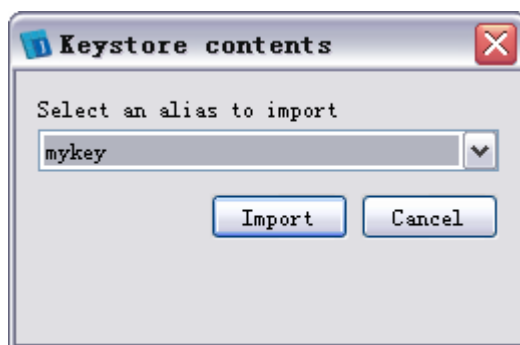


其设为“所有文件”。如图所示：

- 4) 通过文件选择窗口选中密钥库文件，点击“打开”按钮后，会弹出“Enter password”窗口，要求您输入密钥库密码。如图所示：



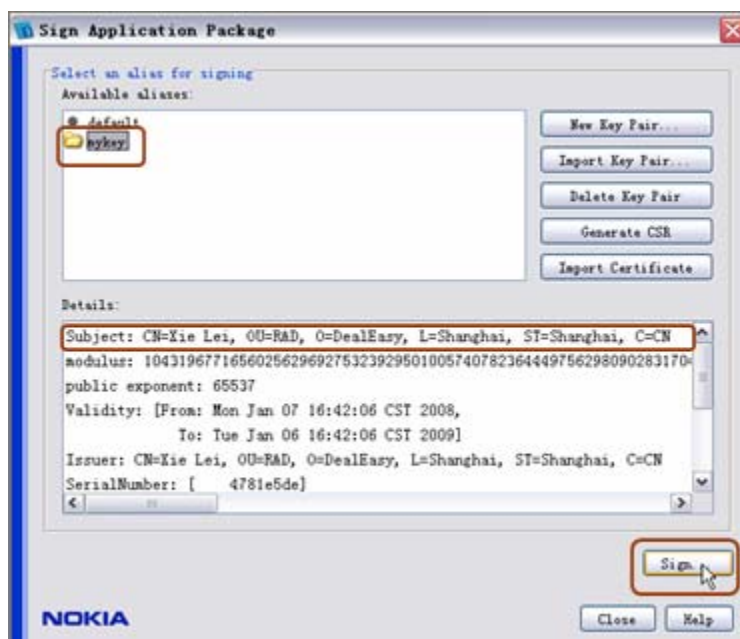
- 5) 输入密钥库密码后, 点击“OK”按钮, 会弹出“Keystore contents”窗口。在这里选择您要载入的密钥对别名。如图所示:



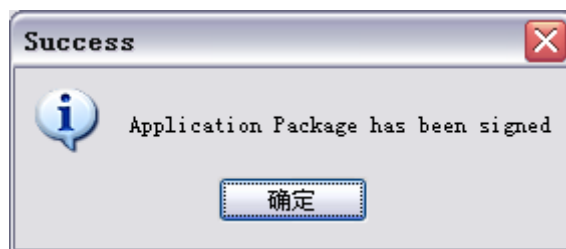
- 6) 选中您要载入的密钥对, 点击“Import”按钮, 会再次弹出“Enter password”窗口。在这里您需要输入该密钥对的密码。如图所示:



- 7) 输入密钥对密码后, 点击“OK”按钮, 加入密钥对成功。
- 8) 在“Sign Application Package”窗口的“Available aliases”列表框中, 选中刚刚载入的密钥对, 点击右下方的“Sign”按钮进行套件签名。如图所示:



- 9) 之后弹出文件选择窗口，选择需要签名的 JAD 文件进行签名。如果签名成功，会弹出下面的成功提示窗口：



- 10) 如果您所选择的 JAD 文件没有和其描述的 JAR 文件同在一个目录下，则会弹出下面的错误提示窗口：

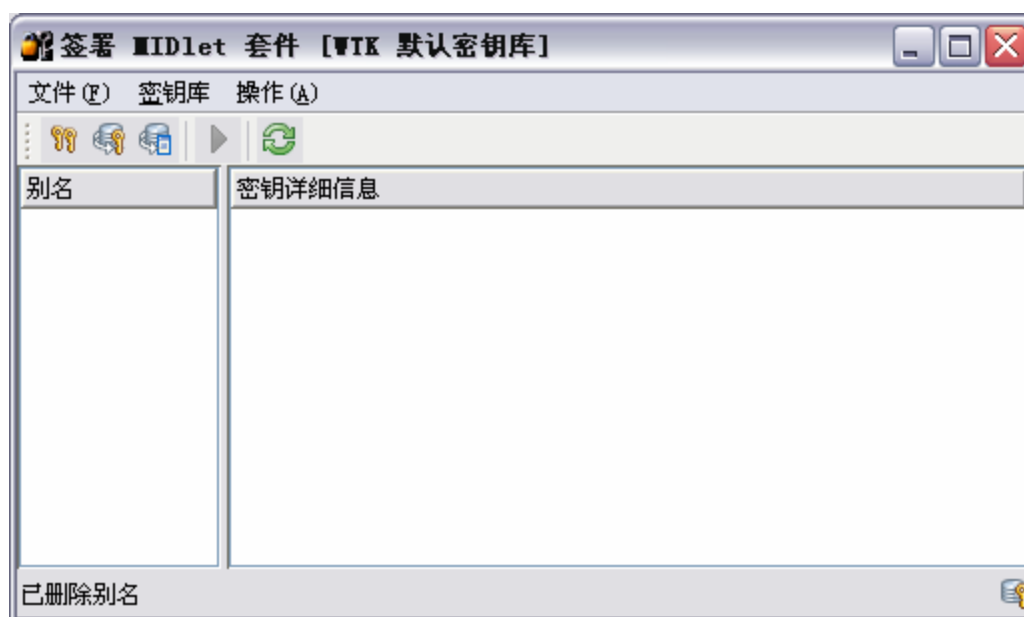


### 4.3. 使用WTK签名

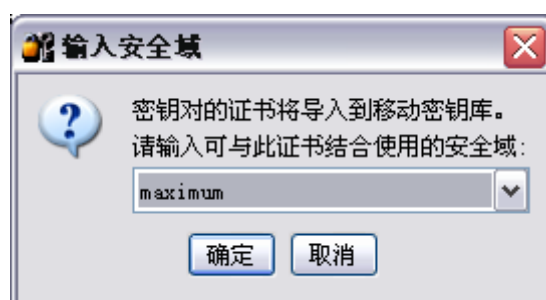
WTK 包含一个工具集 (Utilities)。在工具集中有一个“签署 MIDlet”的工具。我们就是用它对 MIDlet 套件签名的。

由于WTK的签名过程和Carbide.j的签名过程基本相同，下面的步骤描述会比较简单，不明之处请参考[4.2 节](#)。

- 1) 直接打开工具集或者通过 WTK 的“文件->工具集”菜单打开工具集，在工具集的工具列表中选择工具“签署 MIDlet”，打开“签署 MIDlet”工具的主界面。如果您将要签名的 MIDlet 套件创建为 WTK 的项目，也可以通过选择“项目->签署”菜单进行签名。如图所示：



- 2) 选择菜单“密钥库->导入密钥对”，在弹出的文件选择窗口中选择打开需要载入的密钥对所在的密钥库文件。
- 3) 在随后出现的“密码输入”窗口中输入所打开密钥库的密码，点击“确定”按钮。
- 4) 在随后出现的“别名选择”窗口中选择要导入密钥对的别名，点击“确定”按钮。
- 5) 随后会弹出“输入安全域”窗口，让您选择与证书结合使用的安全域。这是 WTK 为了支持其附属的仿真器而添加的步骤，与我们的签署没有直接关系。我们可选择“maximum”安全域。如图所示：

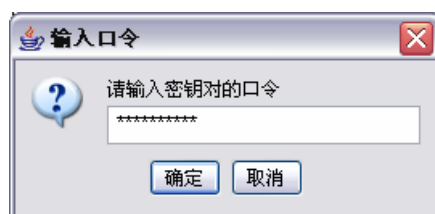


- 6) 点击“确定”按钮，导入密钥对成功。
- 7) 选择菜单“操作->签署 MIDlet 套件”，进行签署。如图所示：





- 8) 在弹出的文件选择窗口中选择打开所要签署的 JAD 文件（保证 JAD 和 JAR 文件在同一个目录下）。
- 9) 接下来会弹出“输入口令”窗口，要求输入密钥对口令。如图所示：



- 10) 正确输入密钥对口令，点击“确定”按钮，弹出成功提示窗口。不过初读提示语句实在令人困惑。如图所示：

