系统应用签名和APK签名学习

|  |  |
| --- | --- |
| **Creator :** | 姚少容 |
| **Department :** | SW |
| **Status :** | Draft |
| **Approver :** |  |
| **Version :** | 0.1 |
| **Location :** |  |

Change History:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Status** | **Date** | **Handled by** | **Comments** |
| 0.1 | Draft | 2014-12-22 | Yaoshaorong | Draft version |
| 0.2 | Draft | 2016-4-13 | Yaoshaorong | Draft version |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[Change History: 1](#_Toc407029771)

[1.签名介绍 2](#_Toc407029772)

[1.1 Android 系统有哪些签名 2](#_Toc407029773)

[1.2签名的目的 2](#_Toc407029774)

[2.如何生成签名文件 3](#_Toc407029775)

[2.1利用平台代码生成 3](#_Toc407029776)

[2.2运行脚本 4](#_Toc407029777)

[3.签名系统应用 5](#_Toc407029778)

[3.1集成到系统 5](#_Toc407029779)

[3.1.1 不带密码的签名文件修改 5](#_Toc407029780)

[3.1.2 带密码的签名文件修改 6](#_Toc407029781)

[3.2系统应用签名定义 7](#_Toc407029782)

[4.如何单独给APK签名 8](#_Toc407029783)

[5.MTK平台修改 10](#_Toc407029784)

[5.1生成签名文件方法 10](#_Toc407029785)

[5.2代码修改 10](#_Toc407029786)

# 1.签名介绍

## 1.1 Android 系统有哪些签名

在Android 系统中，所有安装到系统的应用程序都必有一个数字证书，才能在系统上安装运行，此数字证书不需要权威机构来认证，是开发者自己生成。用于标识应用程序的作者和在应用程序之间建立信任关系。

Android 平台通常使用如下四种数字证书来给应用组件签名：

platform: a key for packages that are part of the core platform.

shared: a key for things that are shared in the home/contacts process.

media: a key for packages that are part of the media/download system.

releasekey: the default key to sign with if not otherwise specified

## 1.2签名的目的

a)应用程序升级：

如果应用程序要升级到新的版本，那么必须用同一个证书进行签名。这是由于只有以同一个证书签名，系统才会允许安装升级的应用程序。如果你采用了不同的证书，那么系统会要求你的应用程序采用不同的包名称，在这种情况下相当于安装了一个全新的应用程序。如果想升级应用程序，签名证书要相同，包名称要相同.  
b)应用程序模块化：Android系统可以允许同一个证书签名的多个应用程序在一个进程里运行，系统实际把他们作为一个单个的应用程序，此时就可以把我们的应用程序以模块的方式进行部署，而用户可以独立的升级其中的一个模块.

c)代码或者数据共享：Android提供了基于签名的权限机制，那么一个应用程序就可以为另一个以相同证书签名的应用程序公开自己的功能。以同一个证书对多个应用程序进行签名，利用基于签名的权限检查，你就可以在应用程序间以安全的方式共享代码和数据了。

# 2.如何生成签名文件

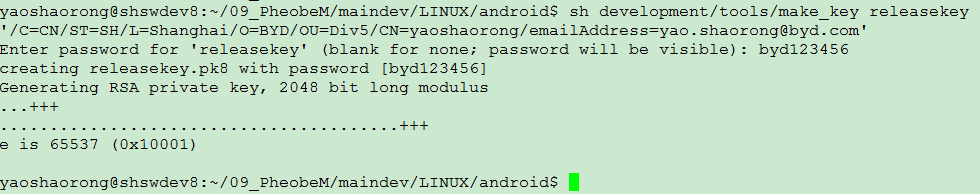
有两种方式生成签名文件，一种是在终端执行命令生成。一种把生成签名的相关内容放到一个文件里面，通过命令执行这个文件即可，生成签名的内容我是从网络上下载。下面是对这两种方式的介绍：

## 2.1利用平台代码生成

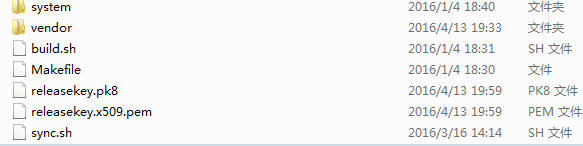
在<project>/ development/ tools目录，有一个make\_key文件，使用make\_key可以生成签名文件。参考build\target\product\security\ README，在根目录执行：



下图是生成releasekey：



过程中需要输入密码，也可以直接回车不输入，最好设置一个有些复杂度的密码，以确保安全性。完成后会在当前目录下会生成一对签名文件如下：



其中 \*.pk8是生成的私钥，而\*.x509.pem是公钥，它们是成对出现。用同样的方法可以生成其他三种key.

生成签名文件中，公司信息参数比较多，这个是可以自己设置的，用于区别其他的签名文件信息，它们的含义如下：

C ---> Country Name (2 letter code)

ST ---> State or Province Name (full name)

L ---> Locality Name (eg, city)

O ---> Organization Name (eg, company)

OU ---> Organizational Unit Name (eg, section)

CN ---> Common Name (eg, your name or your server’s hostname)

emailAddress ---> Contact email address

## 2.2运行脚本

在工程里任意一个目录创建一个mkkey.sh文件，拷贝如下脚本到这个文件，改变AUTH这一行的参数信息，也就是以上公司信息参数.（记得将mkkey.sh这个文件dos2unix转换为linux文档）

#!/bin/sh

AUTH='/C=US/ST=California/L=Mountain View/O=Android/OU=Android/CN=Android/emailAddress=android@android.com'

if [ "$1" == "" ]; then

echo "Create a test certificate key."

echo "Usage: $0 NAME"

echo "Will generate NAME.pk8 and NAME.x509.pem"

echo " $AUTH"

exit

fi

openssl genrsa -3 -out $1.pem 2048

openssl req -new -x509 -key $1.pem -out $1.x509.pem -days 10000 \

-subj "$AUTH"

echo "Please enter the password for this key:"

openssl pkcs8 -in $1.pem -topk8 -outform DER -out $1.pk8 -passout stdin

分别执行如下命令创建这4种key：

sh mkkey.sh platform

sh mkkey.sh media

sh mkkey.sh shared

sh mkkey.sh releasekey

过程中同方法一，输入密码，完成后在当前目录下回生成对应的文件。

（注：不是所有项目都需要生成签名文件，这些签名文件是可以共用的，客户没有特别要求可以用同一套签名，生成一次即可。有的项目客户会提供签名文件，以上步骤可以省略）

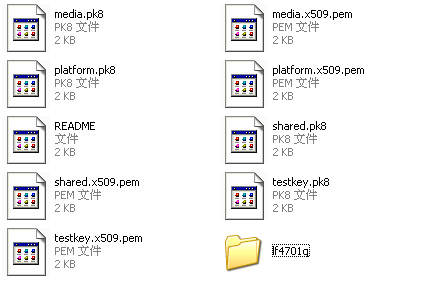
# 3.签名系统应用

## 3.1集成到系统

前面我们生成了系统的签名文件，这些文件就可以应用到系统中替换android默认的签名文件. 这里说明下，既然android系统已经提供签名key，为什么还是我们自己生成？那是因为android代码本身是共享的，这样别的用户也就可以拥有我们的key，这样对系统来说是不安全的，用户可以随意修改代码后替换我们系统的应用. 另外用默认的签名CTS测试也会测不过，所以Android系统默认的签名文件仅供debug使用.

### 3.1.1 不带密码的签名文件修改

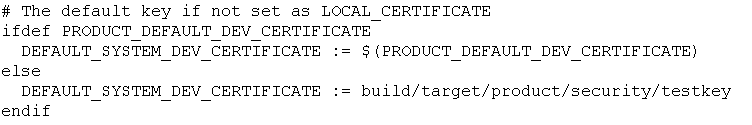
系统默认的签名文件是在如下目录：<project>/build/target/product/security.



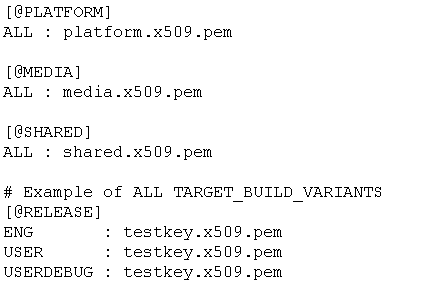
我们可以将生成的签名文件替换这个目录下的文件，这样编译的时候就会使用我们替换的文件. 这个是简便的做法，如果多个项目共用代码，签名文件不同的话，最好还是放在一个单独的项目文件夹。

替换完签名文件后，需要同步修改config.mk和Makefile这两个文件.

<project> \build\core\config.mk

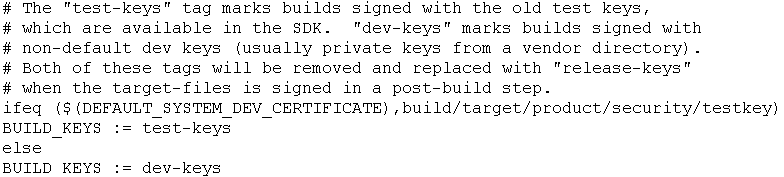


以上代码指出如果PRODUCT\_DEFAULT\_DEV\_CERTIFICATE没有定义，就会使用默认路径下的testkey文件做默认签名。在这里我们可以给PRODUCT\_DEFAULT\_DEV\_CERTIFICATE指定路径，这样就可以使用我们指定目录下的签名了.

另外需要指出，新建的项目目录下是不会放testkey也不需要testkey的，但是在编译过程中，\external\sepolicy\tools\insertkeys.py这个文件会读取\external\sepolicy\keys.conf这个文件下的参数，这些参数分别对应着以上路径下的签名文件，如下： 

其中testkey也被指定，所以如果keys.conf这个文件不改，在编译过程中会提示testkey找不到，编译会中断.

project \build\core\Makefile

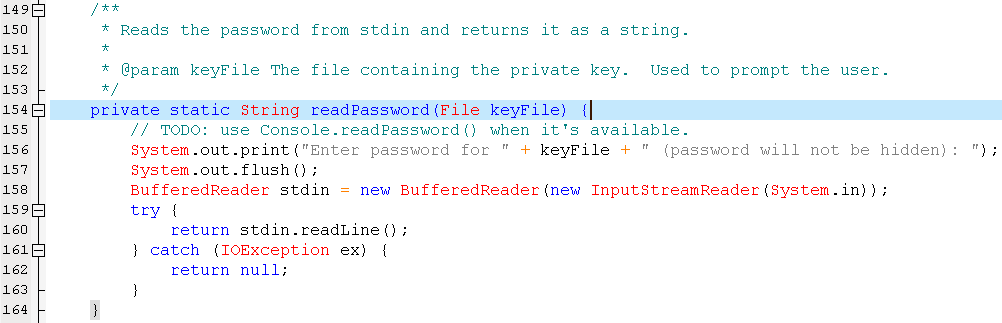


test-keys和dev-keys都是debug时候用的，正式的版本集成了正式的签名文件后需要改成release-keys.

### 3.1.2 带密码的签名文件修改

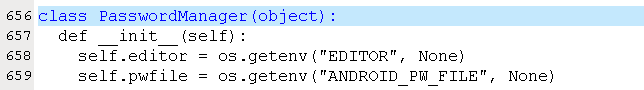
前面我们生成签名文件的时候如果输入了密码，只修改以上就编译，单线程编译过程中会不停的需要输入密码，这会让编译变得非常繁杂. 多线程编译会直接编译报错，追查原因会发现有两个地方要读入密码.

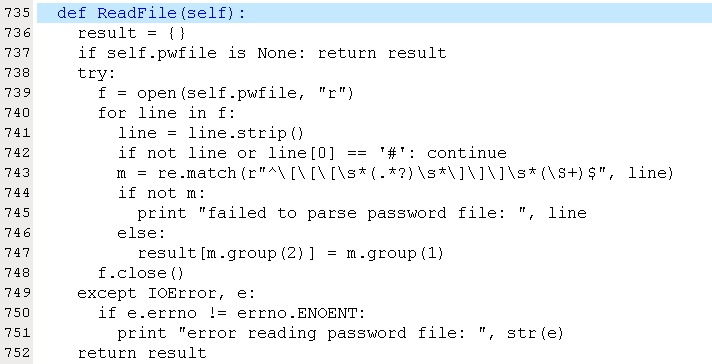
<project>\build\tools\signapk\ SignApk.java



从代码可以看出，这里需要从终端输入密码，而这个密码正是我们key文件的密码。所以我们可以直接写入我们key文件的密码，也可以把密码写在文件中，直接从文件中读入.

<project>\build\tools\releasetools\common.py





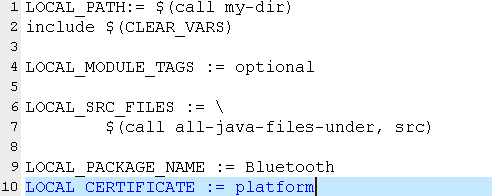
common.py中会读入ANDROID\_PW\_FILE，默认是None，并且密码要符合要求的规则才行.所以带密码的签名文件要把密码写入ANDROID\_PW\_FILE中。可以在设置项目参数的文件中写入.

## 3.2系统应用签名定义

我们都已经知道每个应用都是需要签名的，那怎么知道每个应用都签了什么样的签名呢？

这其实都是有定义，打开应用的Android.mk可以看到有这个一句话：

LOCAL\_CERTIFICATE := platform(or shared or media ) 就是定义签名，releasekey不需要定义，默认就是releasekey.



同样，如果集成三方的时候想要让三方应用签系统的签名，那在mk文件就可以加上这句LOCAL\_CERTIFICATE := xxx。三方应用还可以是LOCAL\_CERTIFICATE := PRESIGNED表示保持三方应用原有的签名.

# 4.如何单独给APK签名

以上是介绍系统应用签名，有时候我们会需要给客户提供的三方签名，那如何给单个apk签我们的系统的签名呢？

首先要安装好开发环境，JDK是必须安装好并把路径设置好。

然后我们需要用到一个工具Auto-sign，是一个压缩包，解压后的文件如下：

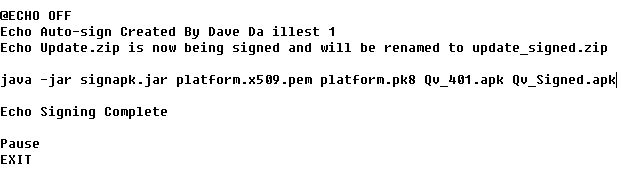


网络上可以下载这个工具，接下来我们分几个步骤介绍：

将系统的签名文件放到这个文件夹中

将需要签名的应用也放到这个文件夹中

用记事本打开Sign.bat并修改这个文件



其实就是一句脚本命令：java -jar signapk.jar platform.x509.pem platform.pk8 Qv\_401.apk Qv\_Signed.apk

Platform是表示你要签的是platform类型，也可以是其他三种类型。

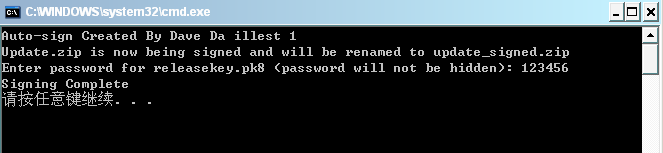
signapk.jar文件如果工程里边有，有工程里边的，网上下载下来代码比较旧.

Qv\_401.apk是你的apk的名字.

Qv\_Signed.apk是签过名后的apk名字，可以自己定义.

修改完后保存.

双击Sign.bat文件，如果有密码需要输入密码.



这时签过名的apk就生成在当前目录了，以上是单独给apk签名的步骤.

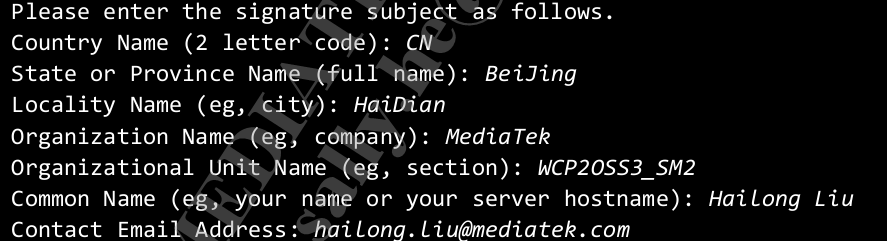
# 5.MTK平台修改

以上文档是基于LF4701Q项目编写，LF4701Q是高通的平台。我们之前有项目是MTK平台，在MTK平台上会有一些差异，MTK做了一些修改实现，所以相对简单一些.

## 5.1生成签名文件方法

在MTK平台中，在根目录执行如下命令并输入相关信息即可生成签名文件

./makeMtk <project> gen-relkey



生成的签名文件在如下目录：<project>/build/target/product/security/<project>/

## 5.2代码修改

打开“MTK\_SIGNATURE\_CUSTOMIZATION”，在如下路径

“mediatek/config/<project>/ProjectConfig.mk” “mediatek/config/common/ProjectConfig.mk” (如果同时有这两个路径，项目的路径会覆盖common), 同时确保 “MTK\_INTERNAL” is disabled.

如果生成没有密码的签名文件，以上修改即可，如果有密码，还要加上3.1.2步骤修改.