Android 应用打包、签名、验证和安装机制

流程：

1、混淆－－－避免反编译以及减小APK

2、签名－－－应用的唯一标识，**起区分作用的，被恶意的第三方覆盖或替换掉**

3、打包－－－将生成的class文件及相关资源及依赖打包成DEX最终生成APK的安装包

4、验证－－－在安装新的应用时验证新的APK与已存在的APK进行签名比对验证，只有签名一致才可以安装。

签名后的**APK**，在**/META-INF**目录下会生成以下**3**个文件：

1、SHA-1(除META-INF目录外的文件) == MANIFEST.MF中的各SHA-1值；

2、(SHA-1 + Base64)(MANIFEST.MF文件及各子项) == CERT.SF中各值

3、公钥（CERT.SF） == CERT.RSA/DSA对SF文件的签名　CERT.RSA/DSA/EC：保存用私钥计算出CERT.SF文件的数字签名、证书发布机构、有效期、公钥、所有者、签名算法等信息

Android应用签名验证过程中，满足以下条件才能安装应用：

1、SHA-1(除META-INF目录外的文件) == MANIFEST.MF中的各SHA-1值；

2、(SHA-1 + Base64)(MANIFEST.MF文件及各子项) == CERT.SF中各值

3、公钥（CERT.SF） == CERT.RSA/DSA对SF文件的签名

应用签名信息可以通过命令来查看

cd .android

keytool -list -v -keystore 签名文件路径**.keystore**

如：**keytool -list -v -keystore debug.keystore**

密码：默认 **android**

正式发布项目时都需要做混淆处理

1. Android代码混淆是一种应用源代码保护技术，用来防止别人对apk进行逆向分析；
2. 从Android2.3开始，Google在SDK中加入了一个叫**ProGuard**的工具，使用它来进行代码混淆。
3. ProGuard是一个压缩、优化和混淆Java字节码文件的免费工具， 其作用：
   - 删除代码中的注释；
   - 删除代码中没有用到的类、字段、方法和属性；
   - 会把代码中的包名、类名、方法名，变量名等修改为abcd...这种没有意义的名字，使得反编译出来的代码难以阅读，从而达到防止apk被破解和逆向分析的目的；
   - 经过ProGuard混淆后，apk安装包会变小；

4. 在实际项目中，有些java类不能进行混淆，需要配置混淆规则；
5. 在实际项目中，打包apk时一般都需要进行混淆处理；
6. 混淆后会生成mapping.txt文件，该文件需要存档以便用来还原和查看混淆后的出错日志；

配置混淆规则

Eclipse：

1、在项目中找到project.properties文件 开启混淆功能

去掉前面的#号注释　意味着开启了混淆

proguard.config=${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt

2、找到当前项目的proguard-project.txt文件配置混淆规则

1、第三方包不进行混淆，项目依赖的所有第三方包或库不可以混淆

2、所有进行反射调用的类方法以及通过反射解析的泛型类不可以混淆

-dontwarn android.support.**　　# 不要提示警告　　-dontwarn 包名.**

-keep class android.support.** { *; }　# 对指定的类不进行混淆　　-keep class 包名.** { *; }

保留要解析的javabean中的泛型信息

-keepattributes Signature

接收EventBus事件的方法保留不混淆

-keepclassmembers class ** {

public void onEvent*(**);

bean包及其子包下, 所有要通过Gson反射解析的javabean都不能进行混淆（只要用到反射的类都不能进行混淆）

```
-keep class com.itheima.proguard.bean.**
-keep class com.itheima.proguard.bean.** { *;}
```

如果用到了WebView中js与java代码的相互调用，js要调用的java类中的方法不能混淆（回调方法声明在内部类JsInterface中的配置方法）

接收处理JS的方法类

```
-keep class com.itheima.proguard.demo03.WebViewActivity {
        public *;
  }
```

回调方法声明在内部类的配置方法 内部类中有回调或反射的方法

```
-keep class com.itheima.proguard.demo03.WebViewActivity$JsInterface {
      public *;
}
```

3、签名打包

4、混淆后生成的相关文件

Eclipse：这四个文件在项目的proguard目录中自动生成

dump.txt

说明 APK 中所有类文件的内部结构。

mapping.txt (核心重要文件要保存起来)

提供原始与混淆过的类、方法和字段名称之间的转换。

seeds.txt

列出未进行混淆的类和成员。

usage.txt

列出从 APK 移除的代码。

Android Studio 混淆方法：

1、开启混淆 在模块的build.gradle文件中 将混淆开启

```
buildTypes {
    //混淆开启
    release {
        minifyEnabled true      //将此处改为 true  表示开启混淆
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}
```

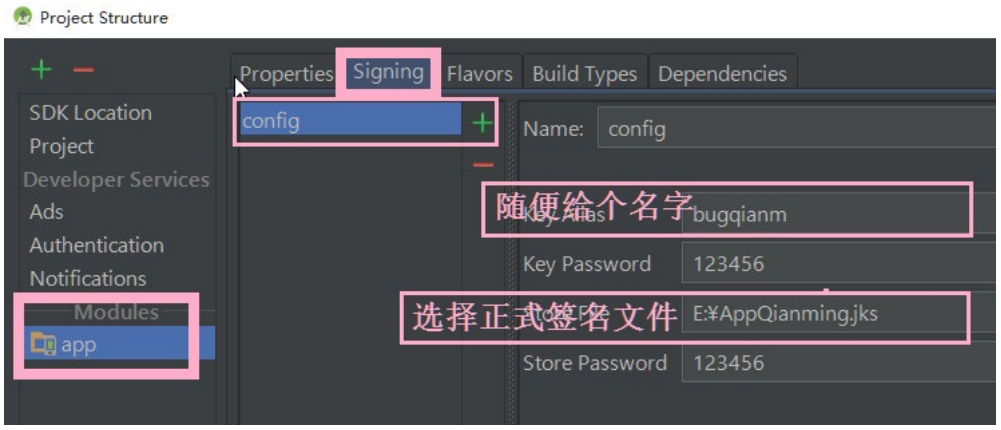
2、在本模块的proguard-rules.pro文件中配置混淆规则，此规则与**Eclipse中的配置方法相同**

3、打包Bulid------->generate signed APK---------->使用或创建签名文件，设置密码---------->确认打包

4、注意问题： debug模式下也可以使用正式的签名模式运行，要使用正式签名模式的方法

Project Structure ------>选择要设置的模块---------->Sinning ---------->选择正式的签名文件，设置密码-------最后设置使用签名

（1）



（2）

操作完以上配置在模块的build.gradle文件中就会多出以下信息，说明配置成功，可以在调试模式下使用正式签名了

```
config {
    keyAlias 'bugqianm'
    keyPassword '123456'
    storeFile file('E:/AppQianming.jks')  //如果项目无法运行，要看一下是否有使用正式签名文件
    storePassword '123456'
}
```

```
buildTypes {
    //调试模式
    debug {
        minifyEnabled true    //开启混淆也不会以混淆运行
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        signingConfig signingConfigs.config    //使用正式签名运行的标志
    }
```