# Signing an Android Application

**1.** Create folder (*’release’* is used here) at a convenient location to keep the apk and keystore files to sign (Desktop is ideal).

**2.** An unsigned package of the application needs to be exported from the IDE. Open the project on eclipse and right-click on the project and select Android-Tools->Export Unsigned Package.

**3.** Copy debug.keystore to the above created folder. In windows, debug.keystore can be found inside ***.android*** directory found under *Users\YourLoginName*

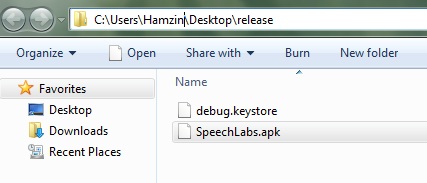


Figure1: The folder after exporting the unsigned package and copying the debug.keystore

**Tip 1:** Few things to know about the debug.keystore. The key alias and the password of the debug keystore will be used to certify during the next step.

* Keystore name: "debug.keystore"
* Keystore password: "android"
* Key alias: "androiddebugkey"
* Key password: "android"

**4.** JDK provides a tool called jarsigner which can be used to create a signed application from the debug keystore. Open command prompt and go to folder that was created in step1 and enter the following command. Type ‘android’ for password when it prompts.

jarsigner -verbose -keystore debug.keystore SpeechLabs.apk androiddebugkey

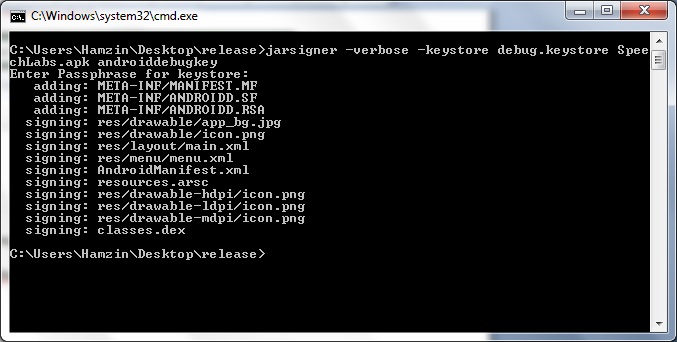


Figure 2: Running the **jarsigner** on ***debug.keystore***

**5.** Once the above step is done, following command can be used to confirm it.

jarsigner -verify SpeechLabs.apk

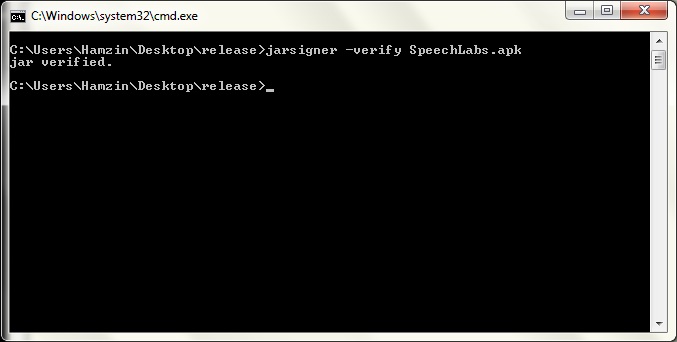


Figure 3: Verifying whether the application is signed properly

**Note:** It is important to understand at this point that this application needs to be certified again using a private (your own) keystore in order to make this available for public downloads via android market. Again jarsigner tool can be used for this purpose.

**6.** The following command can be run to create the private keystore which is to be named ‘release.keystore’ and will have an alias ‘release’. When prompted a desired password can be entered and confirmed. Here ‘android’ will be used as the password for the release keystore as well.

keytool -genkey -v -keystore release.keystore -alias release -keyalg RSA -validity 10000

**7.** After the password is setup, the tool will continue to prompt some additional information. Fill out and type ‘**yes**’ at the end of it to generate the new (i.e. the private keystore) keystore. Again **press** **enter** at the next prompt When prompts for “RETURN if same as keystore password ”.

At the end of this step a new filed named release.keystore should be created.

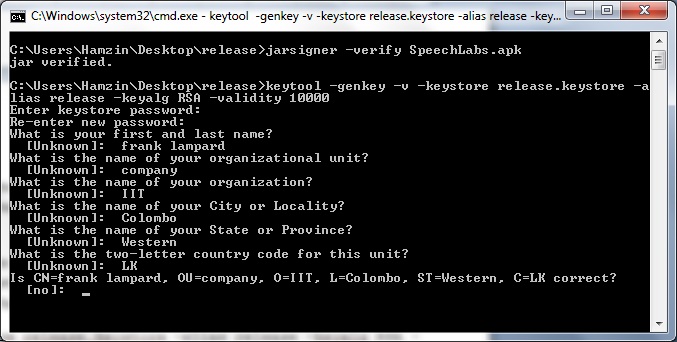


Figure 4: Setting up the private keystore

**8.** The final step is to **sign the application** again **using the private keystore**. This will be similar to the command used in **step 4**. Yet, this time it will be signed with the private keystore and enter ‘android’ when the tool prompts for password.

jarsigner -verbose -keystore release.keystore SpeechLabs.apk release

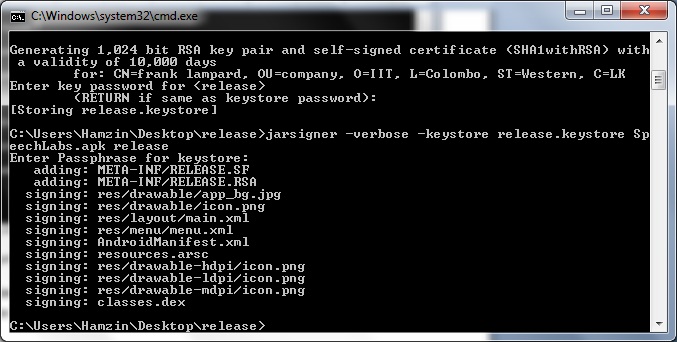


Figure 5: Signing the application using the private keystore

Now the application can be made available for public use and it can now be uploaded to the android market. :-)