SQLCipher for Android Application Integration

This tutorial will cover integrating the binaries of SQLCipher for Android into an Android application. This tutorial assumes the Android SDK is already installed on the local development machine.

Create sample Android application

To create a sample Android application, issue the following commands:

```
% mkdir demo-app
% cd demo-app
% android create project \
  --target android-10 \
  --name demoapp \
  --path . \
  --activity HelloSQLCipherActivity \
  --package com.demo.sqlcipher
```

Obtaining binaries

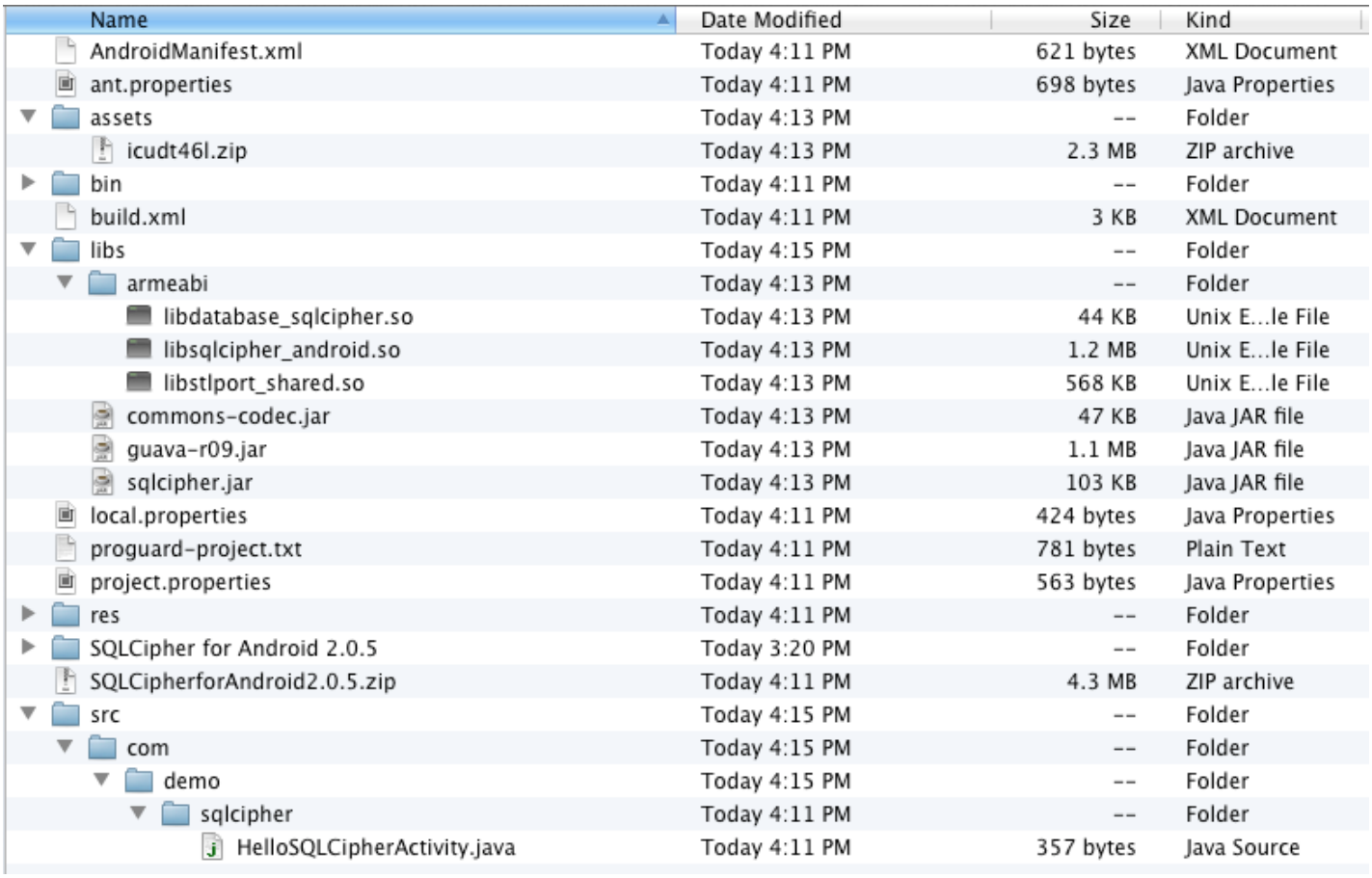
The source code for SQLCipher for Android is publicly maintained on Github.com, the current binary release can be found here (/sqlcipher/open-source). For information on building the source of SQLCipher for Android, please see the instructions below. We will manually download and extract the contents of the compressed tar file into your application root directory with the commands below:

```
% curl -L -o sqlcipher-for-android.zip https://s3.amazonaws.com/sqlcipher/SQLCiphe
r+for+Android+v3.1.0.zip
% unzip sqlcipher-for-android.zip
```

We need to copy the various library and asset files over into the root directory of our application. Execute the following commands:

```
% cp -R SQLCipher\ for\ Android\ v3.1.0/libs/* libs
% cp -R SQLCipher\ for\ Android\ v3.1.0/libs/*assets .
```

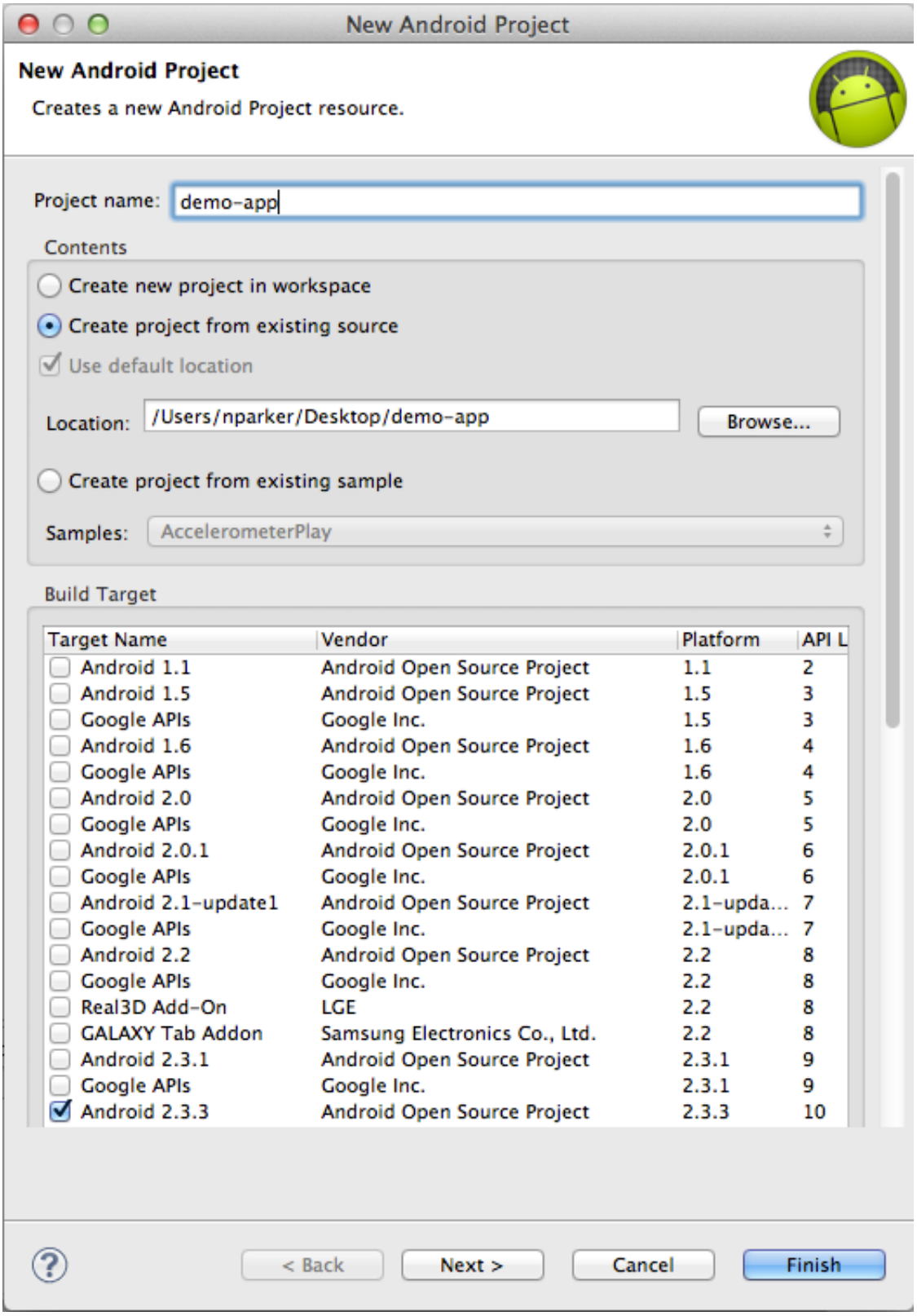
The files for your demo application should look similar to the structure below. Note, `commons-codec.jar` and `guava-r09.jar` are no longer dependencies of the project:



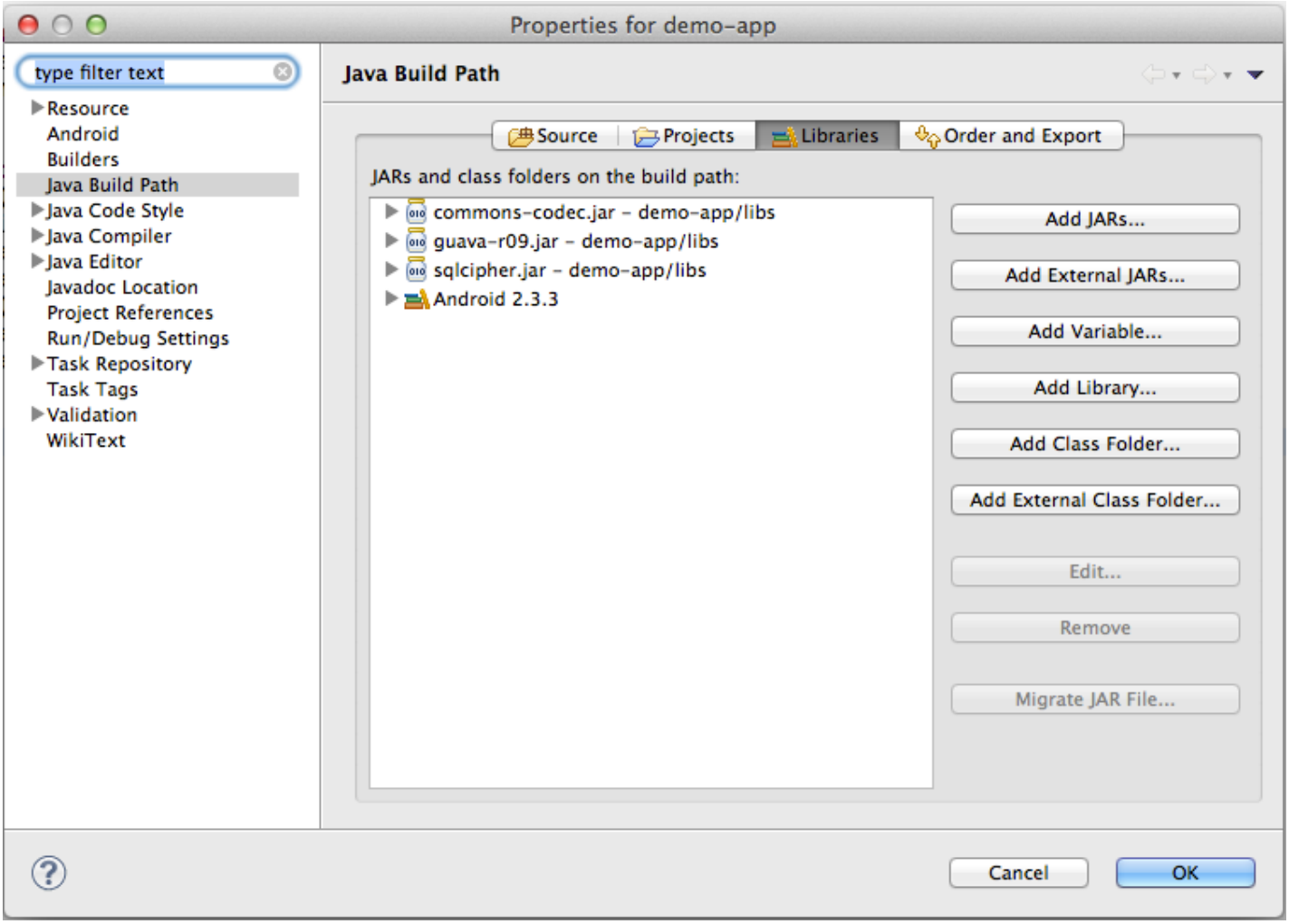
Name	Date Modified	Size	Kind
AndroidManifest.xml	Today 4:11 PM	621 bytes	XML Document
ant.properties	Today 4:11 PM	698 bytes	Java Properties
assets	Today 4:13 PM	--	Folder
icudt46l.zip	Today 4:13 PM	2.3 MB	ZIP archive
bin	Today 4:11 PM	--	Folder
build.xml	Today 4:11 PM	3 KB	XML Document
libs	Today 4:15 PM	--	Folder
armeabi	Today 4:13 PM	--	Folder
libdatabase_sqlcipher.so	Today 4:13 PM	44 KB	Unix E...le File
libsqlcipher_android.so	Today 4:13 PM	1.2 MB	Unix E...le File
libstlport_shared.so	Today 4:13 PM	568 KB	Unix E...le File
commons-codec.jar	Today 4:13 PM	47 KB	Java JAR file
guava-r09.jar	Today 4:13 PM	1.1 MB	Java JAR file
sqlcipher.jar	Today 4:13 PM	103 KB	Java JAR file
local.properties	Today 4:11 PM	424 bytes	Java Properties
proguard-project.txt	Today 4:11 PM	781 bytes	Plain Text
project.properties	Today 4:11 PM	563 bytes	Java Properties
res	Today 4:11 PM	--	Folder
SQLCipher for Android 2.0.5	Today 3:20 PM	--	Folder
SQLCipherforAndroid2.0.5.zip	Today 4:11 PM	4.3 MB	ZIP archive
src	Today 4:15 PM	--	Folder
com	Today 4:15 PM	--	Folder
demo	Today 4:15 PM	--	Folder
sqlcipher	Today 4:11 PM	--	Folder
HelloSQLCipherActivity.java	Today 4:11 PM	357 bytes	Java Source

Integration

Launch Eclipse and choose File -> New -> Android Project from the menu. Give the project a name and choose `Create project from existing source` pointing to your application root directory. It should look similar to this:



Next we need to reference the jar file in our libs directory. Right click on the project node in the Package Explorer and select Build Path -> "Configure Build Path...". Select the Libraries tab and press the "Add JARs..." button. Select `sqlcipher.jar`. After selecting the SQLCipher jar, the screen should appear like this:



Next we will modify the source of the default activity to properly initialize the native libraries for SQLCipher and then create our database file inserting a record. In particular, note the import of `net.sqlcipher.database.SQLiteDatabase` instead of `android.database.sqlite.SQLiteDatabase` as well as the call to `SQLiteDatabase.loadLibs(this)`. The call to `SQLiteDatabase.loadLibs(this)` must occur before any other database operation.

```
package com.demo.sqlcipher;

import java.io.File;
import net.sqlcipher.database.SQLiteDatabase;
import android.app.Activity;
import android.os.Bundle;

public class HelloSQLCipherActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        InitializeSQLCipher();
    }

    private void InitializeSQLCipher() {
        SQLiteDatabase.loadLibs(this);
        File databaseFile = getDatabasePath("demo.db");
        databaseFile.mkdirs();
        databaseFile.delete();
        SQLiteDatabase database = SQLiteDatabase.openOrCreateDatabase(databaseFil
e, "test123", null);
        database.execSQL("create table t1(a, b)");
        database.execSQL("insert into t1(a, b) values(?, ?)", new Object[]{"one fo
r the money",
                                                                    "two for t
he show"});
    }
}
```

The sample application should now be able to run on either an emulator or device.

SQLCipher for Android Build Tutorial

This tutorial will cover building the source code for SQLCipher for Android from scratch. This tutorial targets revision 7 of the Android NDK and is intended for complication under Linux or OSX environments.

System environment requrements

In order to build the source of SQLCipher for Android several tools are required. Several development kits including the Android SDK (http://developer.android.com/sdk/index.html), Android NDK (http://developer.android.com/sdk/ndk/index.html) and the JDK (http://www.oracle.com/technetwork/java/javase/downloads/index.html) need to be present on

the build machine. We will need the `ANDROID_NDK_ROOT` environment variable available on the `PATH`. You will also need Git (http://git-scm.com) to acquire the source tree. To verify your `PATH` is properly configured, execute the following command:

```
% which ndk-build
/Users/nparker/bin/android-ndk/ndk-build
```

If you receive `ndk-build not found` you will need to add the following to your .bashrc or corresponding shell configuration file. Note the `~/bin/android-ndk` path below represents the location on my machine for the root directory of the Android NDK, adjust accordingly:

```
export ANDROID_NDK_ROOT=~/bin/android-ndk
export PATH=$ANDROID_NDK_ROOT:$PATH
```

We will use Make as our base build tool for interacting with the NDK toolchain.

Get the source

The source code for SQLCipher for Android is maintained publicly on Github.com. We will start by cloning the repository locally:

```
% cd ~/code
% git clone git://github.com/sqlcipher/android-database-sqlcipher.git
```

Building

The process of building the code is split into two different phases. The first time this is performed we need to initialize various git submodules that SQLCipher for Android depends on. This is all automated through Make.

```
% cd android-database-sqlcipher
% make init
```

Once git has finished pulling down the various submodules required for compilation, we can begin the build process:

```
% make
```

This process will take some time to complete. Once it has completed, you should have the following files available for integration into your application's libs directory:

```
% tree libs
libs
├── armeabi
│   ├── libdatabase_sqlcipher.so
│   ├── libsqlcipher_android.so
│   └── libstlport_shared.so
└── sqlcipher.jar
```

Localization dependencies

SQLCipher for Android depends on localization data from the ICU project (http://site.icu-project.org). SQLCipher for Android will attempt to use a system provided ICU localization data file called `icudt46l.dat` located in the `/system/usr/icu` directory if available. If that is not found, SQLCipher for Android will attempt to unzip the `icudt46l.zip` file located within the applications asset directory. It is recommended that the `icudt46.zip` file be included with your application for best platform compatibility. If you need to adjust the size of the localization data for your application, a ICU data library customizer is available here (http://apps.icu-project.org/datacustom/ICUData46.html).
